Some Remarks on Variable Independence, Closure, and Orthographic Dimension in Constraint Databases

Leonid Libkin
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
E-mail: libkin@bell-labs.com

Abstract

The notion of variable independence was introduced by Chomicki, Goldin, and Kuper in their PODS'96 paper as a means of adding a limited form of aggregation to constraint query languages while retaining the closure property. Later, Grumbach, Rigoux and Segoufin showed in their ICDT'99 paper that variable independence and a related notion of orthographic dimension are useful tools for optimizing constraint queries.

However, several results in those papers are incorrect as stated. As the notions of variable independence and orthographic dimension appear to be important for implementing constraint database prototypes, I explain in this short note the problems with the above mentioned papers and outline a solution for aggregate closure.

1 Constraint databases

Constraint Databases were introduced by Kanellakis, Kuper and Revesz [KKR90] in their seminal PODS'1990 paper, with the goal of modeling infinite database objects. Such objects arise in a variety of applications, for example, in Geographical Information Systems.

To define a particular constraint model, one needs a context structure. Two context structures most often considered for spatial applications are $\langle \mathbb{R}, +, -, 0, 1, < \rangle$ (*linear constraints*) and $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ (*polynomial constraints*). The role of a tuple is then played by conjunctions of atomic formulae over these structures. Examples of tuples over $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ are

$$\begin{aligned} t_1 &\equiv (x = 1) \wedge (y^2 + z^2 = 1), \\ t_2 &\equiv 2xy + 3yz - xz = 0, \end{aligned}$$

and examples of tuples over $\langle \mathbb{R}, +, -, 0, 1, < \rangle$ are

$$\begin{aligned} t_3 &\equiv (2x - y = 1) \wedge x + y > 0, \\ t_4 &\equiv (0 < x < 1) \wedge (0 < y < 1) \wedge (0 < z < 1) \\ &\quad \wedge (x = y) \wedge (y = z). \end{aligned}$$

A constraint relation then is a finite set of tuples. The semantics of a tuple is the set it defines in $\mathbb{R}^n$; for example, the semantics of t_4 is the diagonal in the open unit cube in $\mathbb{R}^3$. The semantics of a relation is the union of all the sets defined by its tuples. That is, the constraint relation can be thought of as a DNF formula

$$\bigvee_{i=1}^{n} t_i, \quad \text{where} \quad S = \{t_1, \ldots, t_n\}$$

and the semantics of the relation is the set of tuples $\vec{x}$ satisfying this formula, Thus, the semantics of a constraint relation over $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ is a set definable as a Boolean combination of sets of the form $\{\vec{x} \mid p(\vec{x}) > 0\}$, where p is a polynomial. Such sets are known as *semi-algebraic*. The semantics of a constraint relation over $\langle \mathbb{R}, +, -, 0, 1, < \rangle$ is a set definable as a Boolean combination of sets of the form $\{\vec{x} \mid \sum a_i x_i > b\}$. Such sets are known as *semi-linear*. We will thus speak of semi-algebraic and semi-linear databases.

Constraint query languages then are just the relational calculus (first-order logic) extended with the appropriate class of constraints. We use the notation $\mathrm{FO} + \textsc{Lin}$ and $\mathrm{FO} + \textsc{Poly}$ for the extension with linear and polynomial constraints. An example of a $\mathrm{FO} + \textsc{Poly}$ query is $\varphi(x)$ defined as

$$\exists u, v, r \ (\forall y, z \ (S(x, y, z) \leftrightarrow (y - u)^2 + (z - v)^2 = r^2)).$$

Given a set $S \subseteq \mathbb{R}^3$, this query finds all x such that the set $\{y, z \mid (x, y, z) \in S\}$ is a circle. If S is interpreted as $\{t_1\}$, then $\varphi(x)$ is true iff $x = 1$.

The key property of languages like $\mathrm{FO} + \textsc{Lin}$ and $\mathrm{FO} + \textsc{Poly}$ is that they are *closed*: For every semi-linear database D and a $\mathrm{FO} + \textsc{Lin}$ query Q, $Q(D)$

is semi-linear and can be effectively found. Similarly, for every semi-algebraic database D and a $\mathrm{FO}+\mathrm{POLY}$ query Q, $Q(D)$ can be effectively found and is semi-algebraic, see [KKR90]. This is a restatement of the well known fact in logic that the context structures $\langle \mathbb{R}, +, -, 0, 1, < \rangle$ and $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ admit effective quantifier elimination.

2 Adding aggregation

What if one wants to form new queries about volumes of outputs of already defined queries? It is known [Ku93, CGK96] that the closure property is immediately lost. For example, given a semi-linear set $A = \{x, y, z \mid x, y, z > 0, 0 < x < y < z\}$, for each z the area of $A_z = \{x, y \mid (x, y, z) \in A\}$ is $z^2/2$. Another example is that of a semi-algebraic set $A' = \{x, y, z \mid x, y, z > 0, y < z, x \cdot (1 + y^2) < 1\}$. Then the volume of A'_z is $\arctan(z)$, which is not semi-algebraic.

In general, an *aggregate function* maps definable sets (semi-linear or semi-algebraic) to real numbers. Here we concentrate on the main spatial aggregate: volume. We use the notation $\mathrm{VOL}(X)$ for the volume of a set $X \subset \mathbb{R}^n$. As one is normally interested in volumes of bounded sets, we also use the notation $\mathrm{VOL}_I(X)$ for $\mathrm{VOL}(X \cap [0,1]^n)$ (any other bounding box can be chosen without loss of generality).

The first approach to getting a closed language with aggregation proposed in the literature [CGK96] was based on *variable independence*. We say that in a constraint tuple t, variables x and y are *independent* if there is no atomic formula in t that mentions both x and y. These two variables are independent in a constraint relation S if there is a representation $\{t_1, \ldots, t_n\}$ of S such that x and y are independent in each t_i. Note that variables x and y can be independent in S, but not necessarily in every representation: for example, if $t' \equiv (x > 0) \wedge (x \leq y)$, $t'' \equiv (x > 0) \wedge (x \geq y)$ and $t''' = (x > 0)$, then in $S = \{t', t''\}$ variables x and y are independent, although they occur together in some constraints. This is because S is equivalent to $\{t'''\}$.

Suppose we are given a constraint relation S which can be considered as a DNF formula (as explained above). Assume that the free variables of this DNF formula are $\vec{x}$, $\vec{y}$. Then we say that $\vec{x}$ and $\vec{y}$ are independent in S if every x, y coming from $\vec{x}, \vec{y}$ respectively, are independent in S. In this case, S can be given by a DNF formula of the form

$$\bigvee_{i=1}^{n} \alpha_i(\vec{x}) \wedge \beta_i(\vec{y}).$$

If we have a $\mathrm{FO}+\mathrm{LIN}$ ($\mathrm{FO}+\mathrm{POLY}$) query $\varphi(\vec{x}, \vec{y})$ and a semi-linear (semi-algebraic) database D, we say that $\vec{x}$ and $\vec{y}$ are *independent in* φ *on* D if they are independent on the output of φ on D, which is a semi-linear (semi-algebraic) set.

The definition of adding aggregation to language like $\mathrm{FO}+\mathrm{POLY}$ given in [CGK96] is a bit ambiguous, as the text leaves two possible interpretations for aggregate operators, depending on whether these are defined over constraint relations, or their interpretations which are (potentially) infinite subsets of the domain. In what follows, we use the volume aggregate VOL as an example.

Under the first interpretation, the aggregates are applied to a constraint relation. Suppose $\varphi(\vec{x}, \vec{y})$ is an already defined query, where $\vec{x}$ has n variables and $\vec{y}$ has m variables. We then define a new query

$$\psi(\vec{x}, z) \quad \equiv \quad \mathrm{VOL}\ \vec{y}(\varphi(\vec{x}, \vec{y}))$$

with free variables $\vec{x}$ and z (where z is a new variable). To define the semantics, fix a database D. Let $\vec{a}$ be an interpretation $\vec{x}$. Let $S \subseteq \mathbb{R}^{n+m}$ be the result of φ on D. Assume that it is a definable set, that is, it is represented as set of constraint tuples $\{t_1(\vec{x}, \vec{y}), \ldots, t_n(\vec{x}, \vec{y})\}$. For each t_i, define a set

$$A_i(\vec{a}) \;=\; \{\vec{b} \mid \exists t_j (t_j[\vec{x}] = t_i[\vec{x}] \wedge t_j(\vec{a}, \vec{b}))\}.$$

Here $t[\vec{x}]$ denotes the set $\{\vec{c} \mid \exists \vec{y} t(\vec{c}, \vec{y})\}$. Then the output of ψ on D is defined as

$$\{(\vec{a}, c) \mid \exists i : \vec{a} \in t_i[\vec{x}] \wedge c = \mathrm{VOL}(A_i(\vec{a}))\}.$$

The second interpretation of aggregates is somewhat less desirable, as it takes us out of the realm of constraint databases, and operates on unrestricted relations (which are just arbitrary subsets of $\mathbb{R}^n$). Under this interpretation, the definition looks identical to the one above, except that the tuples in a relation are just the usual relational tuples. That is, a relation $S \subseteq \mathbb{R}^n$ is then the set of tuples

$$x_1 = a_1 \wedge x_2 = a_2 \wedge \ldots \wedge x_n = a_n$$

where $(a_1, \ldots, a_n)$ ranges over all (perhaps infinitely many) n-tuples in S.

The resulting language need not be closed, under either interpretation, as follows from the examples given earlier. Then [CGK96] defined a "safe" language by only allowing the application of the VOL operator when $\vec{x}$ and $\vec{y}$ are independent in φ on D. Their claim then was:

Theorem 1 (CGK, PODS'96) *Languages obtained by extending* FO + LIN *and* FO + POLY *with a safe application of* VOL *are closed (i.e., they only define semi-linear and semi-algebraic sets, resp.).* □

However, as stated, this theorem is incorrect. Namely,

Proposition 1 *Languages obtained by extending* FO + POLY *with a safe application of* VOL *are not closed, under either interpretation of definitions in [CGK96].*

Proof. Take S consisting of a single tuple $t_1 \equiv (x = 1) \wedge (y^2 + z^2 = 1)$, in which x is independent from y, z, and apply

$$(x = 1) \wedge [\text{VOL }(y, z)(S(x, y, z))](x, v) \; .$$

This results (regardless of interpretation) in a relation with one tuple $(1, \pi)$ which is not semi-algebraic as π is not definable over $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ (it only defines algebraic numbers). □

Proposition 2 *The language obtained by extending* FO + LIN *with a safe application of* VOL *is not closed, under the constraint interpretation of definitions in [CGK96].*

Proof. Consider a semi-linear relation $S(x, y, u, v)$. That is, S is interpreted as a subset of $\mathbb{R}^4$. Let $\tau_1(x, y, u, v), \ldots, \tau_k(x, y, u, v)$ be formulae listing all order types of x, y, u, v, with τ_1 being $x < y < u < v$. Let t_1 be the constraint tuple $x, y, u, v \in (0, 1) \wedge \tau_1(x, y, u, v)$. ($x \in (0, 1)$ is the conjunction $0 < x \wedge x < 1$.) We put t_1 into S. Next, for every order type τ_i (including τ_1), we do the following. If τ_i is consistent with $y \leq x$, we put $x, y, u, v \in (0, 1) \wedge \tau_i$ into S. Otherwise, we create three tuples:

$$\begin{aligned} t_i' &\equiv x, y, u, v \in (0, 1) \wedge \tau_i \wedge x, y < 1/2 \\ t_i'' &\equiv x, y, u, v \in (0, 1) \wedge \tau_i \wedge x, y > 1/2 \\ t_i''' &\equiv x, y, u, v \in (0, 1) \wedge \tau_i \wedge x < 1/2 \wedge y > 1/2 \end{aligned}$$

and put them in S.

Note that since $t_i' \vee t_i'' \vee t_i'''$ is equivalent to $x, y, u, v \in (0, 1) \wedge \tau_i$, the semantics of S is $(0, 1)^4$ and thus all four variables x, y, u, v are independent from each other. We now form the query

$$\psi(x, y, z) \;\; \equiv \;\; \text{VOL }(u, v) \; (S(x, y, u, v))$$

Consider any tuple t_j which involves a conjunct consistent with $y < x$. Since we have a full listing of order types consistent with $y < x$, for each $a < b$ we have $A_j(a, b) = \{u, v \mid u, v \in (0, 1)\}$. Thus, such a tuple contributes the set $\{x, y, z \mid x, y \in (0, 1) \wedge y < x \wedge z = 1\}$ to the output of ψ. Similarly, for every tuple involving an order type consistent with $y = x$, we get $\{x, y, z \mid y = x \in (0, 1) \wedge z = 1\}$ in the output, and every tuple of the form t_i' contributes $\{x, y, z \mid 0 < x < y < 1/2, z = 1\}$. Similar statements are true for tuples of the form t_i'' and t_i'''. Since we listed every order type in conjunction with $z = 1$, we thus see that tuples other than t_1 contribute the set $\{x, y, z \mid x, y \in (0, 1), z = 1\}$ to the output. This is, of course, a semi-linear set.

Next, consider t_1. The construction of S ensures that there is no other tuple t_j, $j \neq 1$, such that $t_1[x, y] = t_j[x, y]$. Thus, this tuple contributes the following set to the output:

$$\{x, y, z \mid 0 < x < y < 1 \wedge z = \text{VOL}(A_1(x, y))\}$$

where $A_1(x, y)$ is $\{u, v \mid x < y < u < v < 1\}$. That is, $z = \frac{(1-y)^2}{2}$. Putting it all together, the output of ψ on S is

$$\begin{aligned} & \{x, y, z \mid x, y \in (0, 1), z = 1\} \\ \cup \;\; & \{x, y, z \mid 0 < x < y < 1, 2z = (1 - y)^2\} \end{aligned}$$

which is not semi-linear. □

Remark The problem with FO + POLY is not limited to nondefinable constants; one can find an example, along the lines of the one we gave for FO + LIN, that defines transcendental functions with "safe" aggregation. We also remark that in the example here we could have used VOL_I everywhere instead of VOL.

What about the unrestricted interpretation of volume aggregates added to FO + LIN? It turns out that this does give us a closed language (although the proof and the definition in [CGK96] have a gap, and the computability issues are not addressed). We shall explain this shortly, after dealing with variable independence.

3 Testing variable independence

As we mentioned before, the concept of variable independence turned out to be useful in constraint query optimization. Suppose we are given a constraint relation $S \subseteq \mathbb{R}^n$ defined by a formula $\varphi(\vec{x}_1, \ldots, x_n)$. The *orthographic dimension* [GRS99] of S is the maximum size of a subset of variables $\vec{y}$ that is independent from its complement.

One phenomenon observed in [GRS99] is that while many algorithms on constraint databases run in time $O(N^{f(n)})$ in general, where N is the input size, n is its dimension, and f is some function, under the assumption that the orthographic dimension of S is $d < n$, the running time reduces to $O(N^{f(d)})$. In particular, many algorithms become quite efficient when $d = 2$. To understand this phenomenon, assume that we want to find a projection $\exists x\varphi$. In general, quantifier-elimination algorithms would have the dimension in the exponent. However, if S is represented as $\bigvee \alpha_i(\vec{y}) \wedge \beta_i(\vec{z})$ and x comes from $\vec{y}$, then $\exists x\varphi$ is equivalent to $\bigvee(\exists x\alpha_i(\vec{y})) \wedge \beta_i(\vec{z})$ and thus it is the length of $\vec{y}$ that gets into the exponent.

The problem is then to test for orthographic dimension, or more generally, to test for variable independence. The problem naturally arose in [CGK96] (under the assumption that conditions for safe aggregation need to be tested) where an algorithm was proposed for the semi-linear case. As it was only briefly sketched, [GRS99] attempted to give a nice presentation of the algorithm. The criterion they presented was the following.

For a semi-linear set $S \subseteq \mathbb{R}^n$ given by $\varphi(x_1, \ldots, x_n)$, we write $Ind(x_i, x_j)$ if, for every assignment of values a_k to x_k, $k \neq i, j$, the set $\{x_i, x_j \mid \varphi(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, a_{j-1}, x_i, a_{j+1}, \ldots, a_n)\}$ is a rectangular subset of $\mathbb{R}^2$. By a rectangular subset of $\mathbb{R}^2$ we mean a set of the form $\bigcup_{l=1}^{m} I_l \times J_l$ where I_ls and J_ls are intervals on $\mathbb{R}$ (which could be degenerate and unbounded).

Proposition 3 (GRS, ICDT'99) *For a semi-linear set $S \subseteq \mathbb{R}^n$ given by $\varphi(x_1, \ldots, x_n)$, variables x_i and x_j are independent iff $Ind(x_i, x_j)$ holds.* $\square$

It is decidable if a semi-linear set in $\mathbb{R}^2$ is rectangular, and thus [GRS99] concluded from this criterion that variable independence is decidable. However,

Proposition 4 *There exists a semi-linear set $S \subset \mathbb{R}^3$ given by $\varphi(x, y, z)$ such that all $Ind(x, y)$, $Ind(x, z)$ and $Ind(y, z)$ hold, but no pair of variables is independent.*

Proof. Let S be $\{x, y, z \mid 0 \leq x, y, z \leq 1 \wedge x = y = z\}$. Fix any value $c \in [0, 1]$ for z; then $\{x, y \mid (x, y, c) \in S\} = \{x, y \mid x = y = c\}$ which is certainly rectangular. Thus, $Ind(x, y)$ (and the other two) hold. However, it is easy to see that x and y are not independent. $\square$

Note that in the proof above S is one-dimensional. One can get a 3-dimensional counterexample by taking $[0, 1]^3 - S$.

Thus, the decidability results of [CGK96, GRS99] are not valid. It would be a pity to lose decidability of variable independence, as the rest of [GRS99] contains a number of nice results that, although not affected by the above problem, assume that the data comes in the right format (that is, it is of a low orthographic dimension). Fortunately, the decidability is not only recovered, but can even be stated for a larger class of semi-algebraic sets. The proof is a bit involved and will appear elsewhere; however, I can announce the result here.

Proposition 5 (see [Li99]) *Given a semi-algebraic set $S \subseteq \mathbb{R}^n$, its orthographic dimension is computable, and for every two variables x_i and x_j it is decidable if they are independent.* $\square$

4 Closed languages and variable independence

Now that we regained the decidability of variable independence, the question arises whether it can be used to obtain a closed language for constraint databases with the volume aggregate. We now answer this question.

First-order constraint query languages with *independence-restricted volume aggregation* are syntactically just FO + Lin and FO + Poly with the addition of the volume operator; that is, for each query $\varphi(\vec{x}, \vec{y})$, we form a new query

$$\psi(\vec{x}, z) \quad \equiv \quad \textsc{Vol}\ \vec{y}(\varphi(\vec{x}, \vec{y}))$$

where z is a new variable.

To define the semantics, we introduce the notation $\varphi(\vec{a}, D) = \{\vec{b} \mid D \models \varphi(\vec{a}, \vec{b})\}$ for any instantiation $\vec{a}$ for $\vec{x}$ and any database instance D, and $Ind_{\varphi}^{D}(\vec{x}, \vec{y})$ for $\vec{x}$ and $\vec{y}$ being independent in φ on D (that is, being independent in the output of φ on D). Then $D \models \psi(\vec{a}, v)$ iff

$$\begin{cases} v = 0, & Ind_{\varphi}^{D}(\vec{x}, \vec{y}) \text{ fails, or} \\ v = 0, & Ind_{\varphi}^{D}(\vec{x}, \vec{y}) \text{ holds and} \\ & \varphi(\vec{a}, D) \text{ is of infinite measure, or} \\ v = \textsc{Vol}(\varphi(\vec{a}, D)), & Ind_{\varphi}^{D}(\vec{x}, \vec{y}) \text{ holds.} \end{cases}$$

We denote the resulting languages by $\mathrm{FO} + \textsc{Lin} + \textsc{Vol}_{\mathsf{ind}}$ and $\mathrm{FO} + \textsc{Poly} + \textsc{Vol}_{\mathsf{ind}}$ respectively.

Our hope, as before, is to achieve closure by allowing the volume aggregate to be applied to a part of the input that is "independent" from the rest. A typical application, as indicated in [CGK96], is databases with *cadastral* information, that is, information about land ownership and boundaries. In such databases spatial attributes are typically independent from the thematic and temporal ones, and thus one hopes to be able to formulate volume queries about the spatial part.

However, even with $\mathrm{FO} + \mathrm{POLY}$ this does not work. In fact, the example in the proof of Proposition 1 shows that $\mathrm{FO} + \mathrm{POLY} + \mathrm{VOL}_{\mathsf{ind}}$ is not closed.

Still, we have better luck with $\mathrm{FO} + \mathrm{LIN}$.

Theorem 2 *The language* $\mathrm{FO} + \mathrm{LIN} + \mathrm{VOL}_{\mathsf{ind}}$ *is closed and its queries are effectively computable.*

Proof sketch. Suppose we are given $\varphi(\vec{x}, \vec{y})$ and a database D, and assume that $Ind^D_\varphi(\vec{x}, \vec{y})$ holds (which can be tested for). Then the output of φ on D is given in the form $\bigvee_{i=1}^k (\alpha_i(\vec{x}) \wedge \beta_i(\vec{y}))$. We assume without loss of generality that $k > 0$. Let σ be a mapping from $\{1, \ldots, k\}$ to $\{0, 1\}$. For each such σ, define

$$A_\sigma(\vec{x}) = \bigwedge_{i:\sigma(i)=1} \alpha_i(\vec{x}) \quad \wedge \quad \bigwedge_{i:\sigma(i)=0} \neg\alpha_i(\vec{x}),$$

$$B_\sigma(\vec{y}) = \bigvee_{i:\sigma(i)=1} \beta_i(\vec{y}).$$

Each $A_\sigma(\vec{x})$ and $B_\sigma(\vec{y})$ define semi-linear sets, and $A_\sigma(\vec{x})$s are mutually exclusive; moreover, the original set can be represented as $\bigvee_\sigma (A_\sigma \wedge B_\sigma)$.

If a semi-linear set is of finite measure, then its volume is a rational number (see, for example, [BN83]) and hence definable over $\langle \mathbb{R}, +, -, 0, 1, < \rangle$. Moreover, one can effectively test if a semi-linear set is of finite measure, by testing for the existence of a bounding box outside of which the set has lower dimension. Let then q_σ be $\mathrm{VOL}(\{\vec{b} \mid B_\sigma(\vec{b})\})$ if it is defined, and 0 otherwise. Then the output of $\mathrm{VOL}\ \vec{y}(\varphi)$ is the semi-linear set

$$\bigvee_\sigma A_\sigma(\vec{x}) \wedge (v = q_\sigma)$$

where σ ranges over all mappings from $\{1, \ldots, k\}$ to $\{0, 1\}$. $\square$

5 Conclusion

The main conclusions are:

- Variable independence is decidable for relevant spatial domains, and
- There exist closed languages with volume aggregates based on variable independence.

Both results were claimed in [CGK96] (the former only for linear constraints); the decidability was also claimed in [GRS99]. As we showed, both claimed were incorrect as stated, although both could be recovered. As the concept of orthographic dimension and variable independence seems to be promising for optimizing constraint queries [GRS98, GRS99], it is good to have decidability as then one can make use of the techniques developed in [GRS99].

As for the aggregation closure, we showed that the approach based on variable independence does indeed give us a closed language; however, the restriction seems too drastic for most applications. Perhaps a more promising approach is that of [BL99] which extends $\mathrm{FO} + \mathrm{POLY}$ with a safe application of standard relational aggregation, and shows that volumes of semi-linear sets are then definable, without any variable independence restrictions. Also note an interesting commonality between the two approaches: a restriction to semi-linear sets is needed in both cases to obtain closure.

Acknowledgement I thank Michael Benedikt and Gabi Kuper for their comments.

References

[BL99] M. Benedikt and L. Libkin. Exact and approximate aggregation in constraint query languages. In *PODS'99*, pages 102–113.

[BN83] H. Bieri and W. Nef. A sweep-plane algorithm for computing the volume of polyhedra represented in Boolean form. *Linear Algebra and Its Applications* 52/53 (1983), 69–97.

[CGK96] J. Chomicki, D. Goldin and G. Kuper. Variable independence and aggregation closure. In *PODS'96*, pages 40–48.

[GRS98] S. Grumbach, P. Rigaux, L. Segoufin. The DEDALE system for complex spatial queries. In *SIGMOD'98*, pages 213–224.

[GRS99] S. Grumbach, P. Rigaux, L. Segoufin. On the orthographic dimension of constraint databases. In *ICDT'99*, pages 199–216.

[KKR90] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *JCSS*, 51 (1995), 26–52. Extended abstract in *PODS'90*, pages 299–313.

[Ku93] G. Kuper. Aggregation in constraint databases. In *PPCP'93*, 166–173.

[Li99] L. Libkin. Variable independence in theories with quantifier elimination. Bell Labs Technical Memo, 1999.