The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases ⋆

Manolis Koubarakis

Dept. of Computation
UMIST
P.O. Box 88
Manchester M60 1QD
U.K.
manolis@sna.co.umist.ac.uk

Abstract

In previous work we have developed the scheme of indefinite $\mathcal{L}$-constraint databases where $\mathcal{L}$, the parameter, is a first-order constraint language. This scheme extends the constraint database proposal of Kanellakis, Kuper and Revesz to include indefinite (or uncertain) information in the style of Imielinski and Lipski. In this paper we study the complexity of query evaluation in an important instance of this abstract scheme: indefinite temporal constraint databases. Our results indicate that the data/combined complexity of query evaluation does not change when we move from queries in relational calculus over relational databases, to queries in relational calculus with temporal constraints over temporal constraint databases. This fact remains true even when we consider query evaluation in relational databases with indefinite information vs. query evaluation in indefinite temporal constraint databases. In the course of our work, we provide precise bounds on the complexity of decision/quantifier elimination for a subtheory of Presburger arithmetic and a subtheory of real addition with order. The bounds for the latter theory are original and of independent interest.

1 Introduction

An important requirement of advanced temporal applications (e.g., planning and scheduling, medical information systems, geographical information systems and natural language processing systems) is the ability to deal with *in-*

⋆ To appear in Theoretical Computer Science, Special Issue on Uncertainty in Databases and Deductive Systems, Editor: L.V.S. Lakshmanan.

definite (or *uncertain*) temporal information. In [28,30–32] we argued that the combination of *relational databases* and *temporal constraints* offers a powerful framework which addresses this requirement. We have developed the scheme of *indefinite $\mathcal{L}$-constraint databases* where $\mathcal{L}$, the parameter, is a first-order *constraint language.* This parameterized model extends the scheme of [26,27] to include indefinite information in the style of [23,20]. The resulting model allows the representation of *definite, indefinite, finite* and *infinite* information in a single unifying framework.

In this paper we study the complexity of query evaluation in $\mathcal{L}$-constraint databases and indefinite $\mathcal{L}$-constraint databases where $\mathcal{L}$ is dePCL or diPCL. dePCL is a first-order language of *temporal constraints* of the form $x - y \leq c$ where x, y are variables and c is a rational numeral. The intended structure $\mathbf{Q}$ for dePCL interprets the symbol $-$ as subtraction over the set of rationals, the symbol $\leq$ as the relation "less than or equal" over the rationals, and rational numerals as "themselves". diPCL is a first-order language which allows the same kind of atomic temporal constraints but now discrete time is assumed. The intended structure $\mathbf{Z}$ for diPCL interprets constraints over the set of integers. We will collectively refer to diPCL-constraint databases and dePCL-constraint databases as *temporal constraint databases (TCDB)*. Similarly, we refer to indefinite diPCL-constraint databases and indefinite dePCL-constraint databases as *indefinite temporal constraint databases (ITCDB).* The models based on dePCL are essentially the ones studied in [28].

The results of our complexity analysis are summarized in the tables of Figure 1. In these tables RC+TC stands for relational calculus with temporal constraints while $\exists_k$ QL stands for queries in language QL which are in prenex normal form with k alternations of quantifiers beginning with an existential one. $\Diamond$+RC+TC (resp. $\Box$+RC+TC) stands for "possibility" (resp. "certainty") queries in modal relational calculus with temporal constraints. Finally, an entry $\mathcal{C}$, where $\mathcal{C}$ is a complexity class, means that the corresponding query answering problem is complete for class $\mathcal{C}$.

Figure 1 together with the results of [6,51,52,1] shows that the data/combined complexity of query evaluation *does not change* when we move from queries in relational calculus over relational databases, to queries in relational calculus with temporal constraints over TCDB. This fact remains true even if we consider indefinite relational databases vs. ITCDB. The most important contribution of our work is the derivation of the upper bounds in the above tables; the lower bounds follow easily from published work.

Our analysis complements the results of [42,10] and extends the results of [26,27,50]. [42,43,10] have studied Datalog with integer gap-order constraints (a subset of the atomic constraints allowed by diPCL). [26,27] have studied $\mathcal{L}$-constraint databases where $\mathcal{L}$ is the language of rational order with constants

Complexity type	Query language	
	RC+TC	$\exists_k$ RC+TC
Data complexity	LOGSPACE	—
Combined complexity	PSPACE	Σ_k^p

Complexity type	Query language		
	$\Diamond$+RC+TC	$\Box$+RC+TC	$\exists_k$ $\Box$+RC+TC
Data complexity	NP	co-NP	—
Combined complexity	PSPACE	PSPACE	Π_{k+1}^p

Fig. 1. Query evaluation in TCDB (top) and ITCDB (bottom)

(a sublanguage of dePCL). [50] has considered positive existential queries over indefinite $\mathcal{L}$-constraint databases where $\mathcal{L}$ is the language of rational order (a sublanguage of dePCL) or discrete order (a sublanguage of diPCL).

In the course of our work, we study the decision/quantifier-elimination problem for the theories of structures $\mathbf{Q}$ and $\mathbf{Z}$. We show that these problem can be solved in PSPACE (Theorems 7.1, 7.5 and 7.6). In particular, the theory of the structure $\mathbf{Q}$ can be decided in DSPACE(n^3) where n is the length of the input sentence (Theorem 7.1). Since the theory of structure $\mathbf{Q}$ is a subtheory of real addition with order, this result is of independent interest and adds to the literature on the complexity of logical theories [18,16,40,17,15,4,5,26,27]. The results discussed here were originally presented in less detail in [29,33].

This paper is organized as follows. The next section presents some examples of constraint languages and defines the relevant abstract concepts. In Section 3 we introduce the concepts of variable and quantifier elimination. In Section 4 we present the scheme of indefinite $\mathcal{L}$-constraint databases while in Section 5 we discuss query languages for this scheme. In Section 6 we present a quantifier elimination method for evaluating expressions of modal relational calculus with $\mathcal{L}$-constraints over $\mathcal{L}$-constraint databases. Section 7 discusses the problems of decision and quantifier elimination for the temporal theories of structures $\mathbf{Q}$ and $\mathbf{Z}$. Section 8 uses the results of Section 7 to analyze the complexity of query evaluation in TCDB and ITCDB. Finally, Section 9 discusses related work and future research.

2 Constraint Languages

We consider (many-sorted) first-order languages, structures and theories [14]. Every language $\mathcal{L}$ will be interpreted over a *fixed* structure, called the *intended structure*, which will usually be denoted by $\mathbf{M}_{\mathcal{L}}$. If $\mathbf{M}$ is a structure then $Th(\mathbf{M})$ will denote the theory of $\mathbf{M}$ i.e., the set of sentences which are true in $\mathbf{M}$. For every language $\mathcal{L}$, we will distinguish a class of quantifier free formulas called *$\mathcal{L}$-constraints*. The atomic formulas of $\mathcal{L}$ will be included in the class of $\mathcal{L}$-constraints. There will also be two distinguished $\mathcal{L}$-constraints *true* and *false* with obvious semantics. Similar assumptions have been made in [37] in the contex of the CLP scheme. A set of $\mathcal{L}$-constraints will be the algebraic counterpart of the logical conjunction of its members. Thus we will freely mix the terms "set of $\mathcal{L}$-constraints" and "conjunction of $\mathcal{L}$-constraints". We will assume that the reader is familiar with the notions of *solution, consistency* and *equivalence* of sets of constraints [37].

Let us now give some examples of constraint languages.

The language *ECL*. The language *ECL* (*E*quality *C*onstraint *L*anguage) with predicate symbols $=$, $\neq$ and an infinite number of constants has been defined in [26]. The intended structure for this language interprets $=$ as equality, $\neq$ as non-equality and constants as "themselves". An *ECL-constraint* is an ECL formula of the form $x_1 = x_2$ or $x_1 \neq x_2$ where x_1, x_2 are variables or constants. ECL has been used by [26] for the development of an extended relational model based on ECL-constraints.

We now present two languages for expressing *temporal constraints.*

The language *dePCL*. The language *dePCL* (*de*nse *P*oint *C*onstraint *L*anguage) allows us to make statements about points in dense time. dePCL is a first-order language with equality and the following set of non-logical symbols: the set of rational numerals, function symbol $-$ of arity 2 and predicate symbol $<$ of arity 2. The *terms* and *atomic formulas* of dePCL are defined as follows. Constants and variables are terms. If t_1 and t_2 are variables or constants then $t_1 - t_2$ is a term. An *atomic formula* of dePCL is a formula of the form $t \sim c$ or $c \sim t$ where $\sim$ is $<$ or $=$, t is a term and c is a constant.

The intended structure for dePCL is $\mathbf{Q}$. $\mathbf{Q}$ interprets each rational numeral by its corresponding rational number, function symbol $-$ by the subtraction operation over the rationals and $<$ by the relation "less than". The theory $Th(\mathbf{Q})$ is a subtheory of $Th(\mathcal{R}, +, <)$, the theory of real addition with order [39].

A *dePCL-constraint* is a dePCL formula of the form $t \sim c$ where t is a term, c is a constant and $\sim$ is $=$, $<$, $>$, $\leq$ or $\geq$. For example, the formulas $p_1 < p_2$, $p_3 - p_4 \geq 15$, $p_3 = 5/4$ are dePCL-constraints.

The language ***diPCL.*** As in the previous example, we can define the language *diPCL* (*di*screte *P*oint *C*onstraint *L*anguage) which allows us to make statements about points in discrete time. The definition of diPCL is almost identical to the definition of dePCL. The only difference is that points are identified with the integers, and are interpreted accordingly by the structure $\mathbf{Z}$. The theory $Th(\mathbf{Z})$ is a subtheory of $Th(\mathcal{Z}, +, <)$, the theory of integer addition with order (or Presburger arithmetic) [39]. *diPCL- constraints* are defined similarly with dePCL-constraints. For example, the formulas $p_1 \leq p_2$, $p_3 - p_4 < 5$, $p_3 = 8$ are diPCL-constraints.

We will also consider the many-sorted languages ECL+dePCL and ECL+diPCL. The language ECL+dePCL is the union of ECL and dePCL. The sorts of ECL+dePCL are $\mathcal{D}$ (for the infinite set of constants of ECL) and $\mathcal{Q}$ (for the rational numerals of dePCL). The symbols of ECL+dePCL are interpreted by the many-sorted structure which is the union of the intended structures for ECL and dePCL. ECL+diPCL is defined similarly (the sort of integer numerals will be denoted by $\mathcal{Z}$).

3 Variable and Quantifier Elimination

We now define two operations that will be very crucial for the rest of our developments: variable elimination and quantifier elimination. *Variable elimination* is an algebraic operation; its logical counterpart is *quantifier elimination.* In the rest of the paper we will alternate between the two notions and use the one which is more appropriate in each case. We will always assume that we have to deal with formulas of finite length.

Notation 3.1 The vector of symbols $(o_1, \ldots, o_n)$ will be denoted by $\overline{o}$. The natural number n will be called the *size* of $\overline{o}$ and will be denoted by $|\overline{o}|$. This notation will be used for vectors of variables but also for vectors of domain elements. Variables will be denoted by x, y, z, t etc. and vectors of variables by $\overline{x}, \overline{y}, \overline{z}, \overline{t}$ etc. If $\overline{x}$ and $\overline{y}$ are vectors of variables then $\overline{x} \setminus \overline{y}$ will denote the vector obtained from $\overline{x}$ by deleting the variables in $\overline{y}$. If $\overline{x}$ and $\overline{y}$ are vectors of variables and every variable in $\overline{x}$ is also contained in $\overline{y}$ then we will write $\overline{x} \subseteq \overline{y}$. If $\overline{x}$ is a vector of variables then $\overline{x}^0$ will be a vector of constants of the same size. The notation $\overline{x}^0 \setminus \overline{y}^0$ is similarly defined for vectors of constants.

Definition 3.1 Let $\mathcal{L}$ be a many-sorted first-order language. The class of $\mathcal{L}$-constraints *admits variable elimination* iff for every boolean combination ϕ of $\mathcal{L}$-constraints in variables $\overline{x}$, and every vector of variables $\overline{z} \subseteq \overline{x}$, there exists a disjunction ϕ' of conjunctions of $\mathcal{L}$-constraints in variables $\overline{x} \setminus \overline{z}$ such that

(i) If $\overline{x}^0$ is a solution of ϕ then $\overline{x}^0 \setminus \overline{z}^0$ is a solution of ϕ'.
(ii) If $(\overline{x} \setminus \overline{z})^0$ is a solution of ϕ' then this solution can be extended to a solution $\overline{x}^0$ of ϕ.

Some people might find the above definition overly strong. But requiring ϕ' to be just a boolean combination of $\mathcal{L}$-constraints would turn out to be unsatisfactory for the database models discussed in Section 4. The reason is very simple: when we eliminate variables, we would have to deal with negations of $\mathcal{L}$-constraints. Similar arguments and definitions appear in [48].

The following definition will be useful in the forthcoming sections.

Definition 3.2 Let $\mathcal{L}$ be a many-sorted first-order language. The class of $\mathcal{L}$-constraints is *weakly closed under negation* if the negation of every $\mathcal{L}$-constraint is equivalent to a disjunction of $\mathcal{L}$-constraints.

The following proposition, whose proof can easily be done by induction [32], shows that if the class of $\mathcal{L}$-constraints is weakly closed under negation then we can determine whether it also admits variable elimination by considering only conjunctions of $\mathcal{L}$-constraints.

Proposition 3.1 *Let $\mathcal{L}$ be a many-sorted first-order language. The class of $\mathcal{L}$-constraints admits variable elimination if*

- *it is weakly closed under negation, and*
- *for every conjunction θ of $\mathcal{L}$-constraints in variables $\overline{x}$ and every vector of variables $\overline{z} \subseteq \overline{x}$, there exists a disjunction θ' of conjunctions of $\mathcal{L}$-constraints in variables $\overline{x} \setminus \overline{z}$ such that*

(i) If $\overline{x}^0$ is a solution of θ then $\overline{x}^0 \setminus \overline{z}^0$ is a solution of θ'.
(ii) If $(\overline{x} \setminus \overline{z})^0$ is a solution of θ' then this solution can be extended to a solution $\overline{x}^0$ of θ.

The importance of this proposition is that it provides us with the following algorithm for eliminating variables $\overline{z}$ from a boolean combination ϕ of $\mathcal{L}$-constraints in variables $\overline{x}$:

(i) Transform ϕ into a formula where negation applies only to $\mathcal{L}$-constraints (by applying De Morgan's laws and laws for negation).
(ii) Substitute every negated $\mathcal{L}$-constraint by its equivalent disjunction of $\mathcal{L}$-constraints.

(iii) Transform ϕ into disjunctive normal form $\theta_1 \vee \cdots \vee \theta_m$ where each θ_i is a conjunction of $\mathcal{L}$-constraints.
(iv) Perform variable elimination in each θ_i i.e., substitute each disjunct θ_i of ϕ by the equivalent disjunction of conjunctions of $\mathcal{L}$-constraints in variables $\overline{x} \setminus \overline{z}$.

For this algorithm to be effective, we must know how to perform steps 2 and 4. For most languages of interest step 2 will be obvious. Step 4 will usually be more involved. For the languages defined in section 2 variable elimination is straightforward. In the case of ECL a variable elimination algorithm appears in most logic textbooks [14,7]. In the case of diPCL and dePCL, variable elimination can be performed using Fourier's algorithm which was originally devised for linear inequality constraints over the real numbers [44]. Fourier's algorithm can be summarized as follows. Any weak linear inequality involving a variable x can be written in the form $x \leq r_u$ or $x \geq r_l$ i.e., it gives an upper or a lower bound on x. Thus if we are given two linear inequalities, one of the form $x \leq r_u$ and the other of the form $x \geq r_l$, we can eliminate x and obtain the inequality $r_l \leq r_u$. Obviously, $r_l \leq r_u$ is a logical consequence of the given inequalities. In addition, any solution of $r_l \leq r_u$ can be extended to a solution of the given inequalities (simply by choosing for x any value between the values of r_l and r_u). Following this observation, Fourier's elimination algorithm forms all pairs $x \leq r_u$ and $x \geq r_l$, eliminates x and returns the resulting constraints. The generalization of this algorithm to strict linear inequalities is obvious.

The following is an example of variable elimination [30].

Example 3.1 Eliminating variable x_1 from the set of dePCL inequalities

$$\{x_3 \leq x_1,\ x_5 \leq x_1,\ x_1 - x_2 \leq 2,\ x_4 \leq x_5\}$$

gives $\{x_3 - x_2 \leq 2,\ x_5 - x_2 \leq 2,\ x_4 \leq x_5\}$.

In the rest of this paper we will only be interested in constraints which admit variable elimination and are weakly closed under negation. Many interesting classes of constraints fall under this category. The following proposition shows that this is also the case for the constraint classes defined in th section 2.

Proposition 3.2 *The classes of ECL-, diPCL-, and dePCL-constraints admit variable elimination and are weakly closed under negation.*

Let us now consider elimination of quantifiers.

Definition 3.3 Let Th be a theory in some language $\mathcal{L}$. Th *admits elimination of quantifiers* iff for every formula ϕ there is a disjunction ϕ' of conjunctions of $\mathcal{L}$-constraints such that $Th \models \phi \equiv \phi'$.

Similarly to variable elimination this definition is stronger than the traditional one where ϕ' is simply required to be quantifier-free [14]. We require ϕ' to be in the above form because we do not want to deal with negations of $\mathcal{L}$-constraints.

Proposition 3.3 *The languages ECL, diPCL and dePCL admit quantifier elimination.*

Proof: Let us assume that ϕ is a formula of ECL, diPCL or dePCL. The *standard algorithm* to eliminate quantifiers proceeds by computing the PNF form of ϕ and then eliminating quantifiers starting from the innermost one. Eliminating quantifier (Qx) from $(Qx)\theta$ where θ is quantifier-free can be done by transforming θ in DNF and then *eliminating variable* x from each disjunct. The result now follows from proposition 3.2. ■

We have now presented all the concepts necessary for the understanding of constraints and constraint languages. In the next section we show how to integrate constraints and relational databases for the representation of infinite and indefinite information.

4 Indefinite Constraint Databases

In [26,27], Kanellakis, Kuper and Revesz proposed to integrate constraints and relations for the representation of infinite information in a finite way. In this section we extend the scheme of [26] to account for indefinite information in the style of [23,20]. For the rest of this section, let $\mathcal{L}$ be a many-sorted language and $\mathbf{M}_{\mathcal{L}}$ be the *intended* $\mathcal{L}$*-structure.* Let us also assume that the class of $\mathcal{L}$-constraints admits variable elimination and is weakly closed under negation.

For each sort $s \in sorts(\mathcal{L})$, let U_s be a countably infinite set of *attributes* of sort s. The set of all attributes, denoted by $\mathcal{U}$, is $\bigcup_{s \in sorts(\mathcal{L})} U_s$. The sort of attribute A will be denoted by $sort(A)$. With each $A \in \mathcal{U}$ we associate a set of values $dom(A) = dom(s, \mathbf{M}_{\mathcal{L}})$ called the *domain* of A.[1] A *relation scheme* R is a finite subset of $\mathcal{U}$.

We will first define $\mathbf{M}_{\mathcal{L}}$-relations which are unrestricted (i.e., finite or infinite) standard relations. $\mathbf{M}_{\mathcal{L}}$-relations are a theoretical device for giving semantics to indefinite $\mathcal{L}$-constraint relations.

Definition 4.1 Let R be a relation scheme. An $\mathbf{M}_{\mathcal{L}}$*-relational tuple* t over scheme R is a mapping from R to $\bigcup_{s \in sorts(\mathcal{L})} dom(s, \mathbf{M}_{\mathcal{L}})$ such that $t(A) \in$

[1] If s is a sort and M is a structure then $dom(s, \mathbf{M})$ denotes the domain of s in structure M.

$dom(sort(A), \mathbf{M}_{\mathcal{L}})$. An $\mathbf{M}_{\mathcal{L}}$*-relation* r over scheme R is an *unrestricted* set of $\mathbf{M}_{\mathcal{L}}$-relational tuples over R.

For every $s \in sorts(\mathcal{L})$, we now assume the existence of two disjoint countably infinite sets of *variables*: the set of *u-variables* $UVAR^s_{\mathcal{L}}$ and the set of *e-variables* $EVAR^s_{\mathcal{L}}$. Let $UVAR_{\mathcal{L}}$ and $EVAR_{\mathcal{L}}$ denote $\bigcup_{s \in sorts(\mathcal{L})} UVAR^s_{\mathcal{L}}$ and $\bigcup_{s \in sorts(\mathcal{L})} EVAR^s_{\mathcal{L}}$ respectively. The intersection of the sets $UVAR_{\mathcal{L}}$ and $EVAR_{\mathcal{L}}$ with the domains of attributes is empty.

Notation 4.1 U-variables will be denoted by letters of the English alphabet, usually x, y, z, t, possibly subscripted. E-variables will be denoted by letters of the Greek alphabet, usually $\omega, \lambda, \zeta, \nu$, possibly subscripted.

Definition 4.2 Let R be a relation scheme. An *indefinite $\mathcal{L}$-constraint tuple* t over scheme R is a mapping from $R \cup \{CON\}$ to $UVAR_{\mathcal{L}} \cup WFF(\mathcal{L})$ such that (i) $t(A) \in UVAR^{sort(A)}_{\mathcal{L}}$ for each $A \in R$, (ii) $t(A_i)$ is different than $t(A_j)$ for all distinct $A_i,\ A_j \in R$, (iii) $t(CON)$ is a conjunction of $\mathcal{L}$-constraints and (iv) the free variables of $t(CON)$ are included in $\{t(A):\ A \in R\} \cup EVAR_{\mathcal{L}}$. $t(CON)$ is called the *local condition* of the tuple t while $t(R)$ is called the *proper part* of t.

Definition 4.3 Let R be a relation scheme. An *indefinite $\mathcal{L}$-constraint relation* over scheme R is a *finite* set of indefinite $\mathcal{L}$-constraint tuples over R. Each indefinite $\mathcal{L}$-constraint relation r is associated with a boolean combination of $\mathcal{L}$-constraints $G(r)$, called the *global condition* of r.

Similarly we can define database schemes, $\mathbf{M}_{\mathcal{L}}$-relational databases and indefinite $\mathcal{L}$-constraint databases [32]. Database schemes and databases will usually be denoted by $\widetilde{R}$ and $\widetilde{r}$ respectively.

The above definitions extend the model of [26] by introducing *e-variables* which have the semantics of marked nulls of [23]. As in [20], the possible values of the e-variables can be constrained by a *global condition*.

Example 4.1 BOOKED is an indefinite ECL+dePCL-constraint relation giving the times that rooms are booked. The first tuple says that room WP212 is booked from 1:00 to 7:00. For room WP219 the information is indefinite: it is booked from 1:00 until some time between 5:00 and 8:00. This indefinite information is captured by the *e-variable* ω and its global condition $5 \leq \omega \leq 8$. *E-variables* can be understood as being existentially quantified and their scope is the entire database. They represent values that exist but are not known precisely [23,20]. All we know about these values is captured by the global condition. *U-variables* (e.g., x_1, x_2, t_1, t_2) can be understood as being universally quantified and their scope is the tuple in which they appear [26].

BOOKED

Room	Time	CON
x_1	t_1	$x_1 = WP212,\ 1 \leq t_1 < 7$
x_2	t_2	$x_2 = WP219,\ 1 \leq t_2 < \omega$

$$G(BOOKED):\ 5 \leq \omega \leq 8$$

Example 4.2 Let us consider a planning database used by a medical laboratory for keeping track of patient appointments for the year 1996. The set of integers $\mathcal{Z}$ will be our time line. The year 1996 is assumed to start at time 0 and every interval $[i, i+1)$ represents a day (for $i \in \mathcal{Z}$ and $i \geq 0$). Time intervals will be represented by their endpoints. They will always be assumed to be of the form $[B, E)$ where B and E are the endpoints. The following indefinite ECL+diPCL-constraint relation APPOINTMENT gives a possible state of the database. [2]

APPOINTMENT

Patient	Treatment	Begin	End	CON
Smith	Chemotherapy1	ω_1	ω_2	*true*
Smith	Chemotherapy2	ω_3	ω_4	*true*
Smith	Radiation	ω_5	ω_6	*true*

G(APPOINTMENT) :

$$\omega_1 \geq 0,\ \omega_2 \geq 0,\ \omega_3 \geq 0,\ \omega_4 \geq 0,\ \omega_5 \geq 0,\ \omega_6 \geq 0,$$

$$\omega_2 = \omega_1 + 1,\ \omega_4 = \omega_3 + 1,\ \omega_6 = \omega_5 + 2$$

$$\omega_2 \leq 91,\ \omega_3 \geq 91,\ \omega_4 \leq 182,$$

$$\omega_3 - \omega_2 \geq 60,\ \omega_5 - \omega_4 \geq 20,\ \omega_6 \leq 213$$

The above database represents the following information:

(i) There are three scheduled appointments for patient Smith. This is represented by three tuples in relation APPOINTMENT.

[2] In this example we do not follow strictly our definition of an $\mathcal{L}$-constraint tuple. The example can easily be made to conform with Definition 4.2 by introducing u-variables and adding equality constraints in the local condition of each tuple.

(ii) Chemotherapy appointments must be scheduled for a single day. Radiation appointments must be scheduled for two consecutive days. This information is represented by constraints $\omega_2 = \omega_1 + 1$, $\omega_4 = \omega_3 + 1$, and $\omega_6 = \omega_5 + 2$.

(iii) The first chemotherapy appointment for Smith should take place in the first three months of 1996 (i.e., days 0-91). This information is represented by constraint $\omega_2 \leq 91$.

(iv) The second chemotherapy appointment for Smith should take place in the second three months of 1996 (i.e., days 92-182). This information is represented by constraints $\omega_3 \geq 91$ and $\omega_4 \leq 182$.

(v) The first chemotherapy appointment for Smith must precede the second by at least two months (60 days). This information is represented by constraint $\omega_3 - \omega_2 \geq 60$.

(vi) The radiation appointment for Smith should follow the second chemotherapy appointment by at least 20 days. Also, it should take place before the end of July (i.e., day 213). This information is represented by constraints $\omega_5 - \omega_4 \geq 20$ and $\omega_6 \leq 213$.

This example shows that the formalism of this paper needs to be extended with intervals and calendars so that it can be used more effectively in real-life applications. The extension with intervals has already been described in [30].

4.1 Semantics

Let us first define two special kinds of valuations. An *e-valuation* in $\mathbf{M}_\mathcal{L}$ is a valuation whose domain is restricted to the set $EVAR_\mathcal{L}$. Similarly, a *u-valuation* in $\mathbf{M}_\mathcal{L}$ is a valuation whose domain is restricted to the set $UVAR_\mathcal{L}$. The symbols $Val^e_{\mathbf{M}_\mathcal{L}}$ and $Val^u_{\mathbf{M}_\mathcal{L}}$ will denote the set of e-valuations and u-valuations in $\mathbf{M}_\mathcal{L}$ respectively. The result of applying an e-valuation v to an indefinite $\mathcal{L}$-constraint relation r over R will be denoted by $v(r)$. $v(r)$ is an $\mathcal{L}$-constraint relation over R obtained from r by substituting each e-variable ω of r by the constant symbol whose denotation in structure $\mathbf{M}_\mathcal{L}$ is $v(\omega)$. The result of applying a u-valuation of $\mathbf{M}_\mathcal{L}$ to the proper part of a tuple can be defined as follows. If t is an $\mathcal{L}$-constraint tuple on scheme R and u is a u-valuation in $\mathbf{M}_\mathcal{L}$ then $u(t)$ is an $\mathbf{M}_\mathcal{L}$-tuple over R such that for each $A \in R$, $u(t)(A) = u(t(A))$.

The semantics of an $\mathcal{L}$-constraint relation is given by the function *points* [26]. *points* takes as argument an $\mathcal{L}$-constraint relation r over R and returns the $\mathbf{M}_\mathcal{L}$-relation over R which is finitely represented by r:

$$points(r) = \{u(t) : \ t \in r, \ u \in Val^u_{\mathbf{M}_\mathcal{L}} \text{ and } \mathbf{M}_\mathcal{L} \models t(CON)[u]\}.$$

The *semantics* of an indefinite $\mathcal{L}$-constraint relation r over scheme R is defined to be the following set of $\mathbf{M}_{\mathcal{L}}$-relations:

$$sem(r) = \{points(v(r)) : \text{ there exists } v \in Val^{e}_{\mathbf{M}_{\mathcal{L}}} \text{ s.t. } \mathbf{M}_{\mathcal{L}} \models G(r)[v]\}.$$

The function *rep* will also be useful in the rest of this paper. If r is an indefinite $\mathcal{L}$-constraint relation over scheme R then *rep* gives the set of $\mathcal{L}$-constraint relations represented by r:

$$rep(r) = \{v(r) : \text{ there exists } v \in Val^{e}_{\mathbf{M}_{\mathcal{L}}} \text{ s.t. } \mathbf{M}_{\mathcal{L}} \models G(r)[v]\}$$

The functions *points*, *sem* and *rep* can be extended to databases in the obvious way. [3]

5 Querying Indefinite $\mathcal{L}$-Constraint Databases

[26] proposed *relational calculus with $\mathcal{L}$-constraints* as a declarative query language for $\mathcal{L}$-constraint databases. In this section we propose *modal relational calculus with $\mathcal{L}$-constraints* as a declarative query language for indefinite $\mathcal{L}$-constraint databases. Similar modal query languages have been investigated in [36,35,41]. We have also defined an algebraic language, the *modal $\mathcal{L}$-constraint algebra*, for querying indefinite $\mathcal{L}$-constraint databases. The interested reader can consult [30,31] for more details.

Definition 5.1 Let $\widetilde{R}$ be a database scheme and $R(C_1, \ldots, C_m)$ be a relation scheme. An expression over $\widetilde{R}$ in *modal relational calculus with $\mathcal{L}$-constraints* is

$$\{R(C_1, \ldots, C_m), x_1/s_1, \ldots, x_m/s_m \;:\; OP\ \phi(x_1, \ldots, x_m)\}$$

where $s_i \in sorts(\mathcal{L})$ is the sort of C_i, OP is an *optional* modal operator $\Diamond$ or $\Box$, ϕ is a well-formed formula of relational calculus with $\mathcal{L}$-constraints and $x_1, \ldots, x_m$ are the only free variables of ϕ. If an expression does not contain a modal operator then it will be called *pure*, otherwise it will be called *modal.*

[3] The above definitions imply that indefinite $\mathcal{L}$-constraint relations are interpreted in a *closed-world* fashion. They are assumed to represent all facts relevant to an application domain. However the exact value of any attribute of these facts may not be known precisely.

We will now define the *value* of expressions in modal relational calculus.

Definition 5.2 Let f be the pure expression

$$\{R(C_1,\ldots,C_m), x_1/s_1,\ldots,x_m/s_m \ :\ \phi(x_1,\ldots,x_m)\}$$

over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then the *value* of f on the set of $\mathbf{M}_{\mathcal{L}}$-relational databases $sem(\widetilde{r})$, whose finite representation is $\widetilde{r}$, is the following set of $\mathbf{M}_{\mathcal{L}}$-relations:

$$f(sem(\widetilde{r})) = \{\ \{(a_1,\ldots,a_m) \in dom(s_1)\times\cdots\times dom(s_m) : \\ (\mathbf{M}_{\mathcal{L}}, Dom, \widetilde{r}') \models \phi(a_1,\ldots,a_m)\}\ : \widetilde{r}' \in sem(\widetilde{r})\}$$

The question left open by the above definition is whether we can guarantee *closure* as required by the constraint query language principles laid out in [26]. In other words, given a pure expression f of modal relational calculus with $\mathcal{L}$-constraints, and an indefinite $\mathcal{L}$-constraint database $\widetilde{r}$, is it possible to find an indefinite $\mathcal{L}$-constraint relation which finitely represents $f(sem(\widetilde{r}))$? In [31,30] we have answered this question affirmatively by showing how to translate calculus queries into algebraic queries. In Section 6, we show an alternative way of performing query evaluation based on quantifier elimination.

Example 5.1 The query "Find all rooms that are booked at 6:00" over the database of Example 4.1 can be expressed as

$$\{BOOKED\_AT\_6(Room), x/\mathcal{D} : BOOKED(x,6)\}.$$

The answer to this query is given by the following relation:

BOOKED_AT_6

Room	CON
x_1	$x_1 = WP212$
x_2	$x_2 = WP219,\ \omega > 6$

This answer is *conditional.* Room WP212 is booked on time 6. However, room WP219 is booked on time 6 *only under the condition* that ω is greater than 6. In Section 6 we will show how to evaluate calculus queries and compute a finite representation of the answer.

Definition 5.3 Let f be the modal expression

$$\{R(C_1,\ldots,C_m), x_1/s_1,\ldots,x_m/s_m \ :\ \Box\, \phi(x_1,\ldots,x_m)\}$$

over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then the *value* of f on the set of $\mathbf{M}_{\mathcal{L}}$-relational databases $sem(\widetilde{r})$, whose finite representation is $\widetilde{r}$, is the following *singleton* set of $\mathbf{M}_{\mathcal{L}}$-relations:

$$\begin{array}{c} f(sem(\widetilde{r})) = \{\ \{(a_1, \ldots, a_m) \in dom(s_1) \times \cdots \times dom(s_m) : \\ \text{for every } \mathbf{M}_{\mathcal{L}}\text{-relational database } \widetilde{r}' \in sem(\widetilde{r}) \\ (\mathbf{M}_{\mathcal{L}}, Dom, \widetilde{r}') \models \phi(a_1, \ldots, a_m)\}\ \} \end{array}$$

The value of a $\diamond$-expression is defined in the same way but now the quantification over $\mathbf{M}_{\mathcal{L}}$-relational databases in $sem(\widetilde{r})$ is existential. Section 6 and [31,30] demonstrate that expressions of modal relational calculus with $\mathcal{L}$-constraints can also be evaluated in closed form. In summary, for every expression f (pure or modal) in modal relational calculus with $\mathcal{L}$-constraints and indefinite $\mathcal{L}$-constraint database $\widetilde{r}$, it is possible to find an indefinite $\mathcal{L}$-constraint relation which finitely represents $f(\widetilde{r})$.

Example 5.2 The query "Find all rooms that are possibly booked at 6:00" over the database of Example 4.1 can be expressed as

$$\{POSS\_BOOKED\_AT\_6(Room), x/\mathcal{D} : \diamond BOOKED(x, 6)\}.$$

If this query is evaluated using the method of Section 6, the answer will be the following relation:

POSS_BOOKED_AT_6

Room	CON
x_1	$x_1 = WP212$
x_2	$x_2 = WP219$

The above answer is *unconditional*. It is possible that both rooms WP212 and WP219 are booked on time 6.

Example 5.3 Let us now consider the database of Example 4.2 and the query "Find all appointments for Smith that should necessarily take place in the first six months of 1996". This query can be expressed as follows:

$$\begin{array}{c} \{APP\_SMITH(Patient, Treatment), x, y/\mathcal{D} : \\ \Box(APPOINTMENT(x, y, t_1, t_2)\ \wedge\ t_1 \geq 0\ \wedge\ t_2 \leq 182)\}. \end{array}$$

The answer to this query is the following relation:

APP_SMITH

Patient	Treatment	CON
Smith	Chemotherapy1	*true*
Smith	Chemotherapy2	*true*

Queries in modal relational calculus with $\mathcal{L}$-constraints allow $\Diamond$ and $\Box$ as top-level connectives only. It is not difficult to relax this requirement to allow for a full-fledged modal query language with constraints (but we will not explore this route in this paper). The interested reader is referred to [36,35,41] for similar efforts.

6 Evaluating Calculus Queries Over $\mathcal{L}$-Constraint Databases

In [31] we have shown that expressions of modal relational calculus with $\mathcal{L}$-constraints have equivalent expressions in modal $\mathcal{L}$-constraint algebra. Thus we can evaluate a calculus expression by evaluating an equivalent algebraic expression. As we have shown in [30,31], algebraic query evaluation can be done bottom-up and the answer is obtained in closed form. Therefore calculus expressions can also be evaluated bottom-up in closed form on indefinite $\mathcal{L}$-constraint databases. Our results (as well as the analogous theorems of [25,30,38]) provide a translation of calculus expressions into algebraic expressions. This translation can be the first step in optimizing the evaluation of expressions in relational calculus with $\mathcal{L}$-constraints.

Query evaluation over indefinite $\mathcal{L}$-constraint databases can also be viewed as quantifier elimination in the theory $Th(\mathbf{M}_{\mathcal{L}})$. Quantifier elimination is always possible in our framework since $Th(\mathbf{M}_{\mathcal{L}})$ admits quantifier elimination. This idea was originally presented in [26] in the less general scheme of $\mathcal{L}$-constraint databases. The following theorem is from [26].

Notation 6.1 We will use $\overline{x}, \overline{y}, \overline{z}, \ldots$ to represent vectors of variables of $\mathcal{L}$, $\overline{a}, \overline{b}, \overline{c}, \ldots$ to represent vectors of domain elements, $\overline{X}, \overline{Y}, \overline{Z}, \ldots$ to represent vectors of attributes of relations, $\overline{s}, \overline{s}', \ldots$ to represent vectors of sorts and $\overline{\omega}$ to represent vectors of e-variables. The size of a vector $\overline{x}$ will be denoted by $|\overline{x}|$.

Theorem 6.1 *Let $\tilde{r}$ be an $\mathcal{L}$-constraint database over $\tilde{R}$ and f be the expression $\{R(\overline{X}), \overline{x}/\overline{s} : \phi(\overline{x})\}$ over $\tilde{R}$ in relational calculus with $\mathcal{L}$-constraints. Then $f(points(\tilde{r})) = \{ \ \overline{a} : \ \mathbf{M}_{\mathcal{L}} \models \psi[\overline{x} \leftarrow \overline{a}] \ \}$ where ψ is the formula of $\mathcal{L}$*

corresponding to ϕ and $\tilde{r}$.

The formula of $\mathcal{L}$ corresponding to ϕ and $\tilde{r}$ can be obtained from ϕ by substituting each occurence of a database predicate $R(\overline{x})$ by the disjunction of conjunctions of $\mathcal{L}$-constraints which is equivalent to the relation over scheme R.

Let us now consider indefinite $\mathcal{L}$-constraint databases.

Theorem 6.2 *Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database over $\tilde{R}$. Let $G(\overline{\omega})$ be the global condition of $\tilde{r}$ where $\overline{\omega}$ is a vector of e-variables of sort $\overline{s}'$. Let f be the expression $\{R(\overline{X}), \overline{x}/\overline{s} :\ OP\ \phi(\overline{x})\}$ over $\tilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. If OP is $\Diamond$ then*

$$f(sem(\tilde{r})) = \{\ \{\ \overline{a} :\ \mathbf{M}_{\mathcal{L}} \models (\exists \overline{z}/\overline{s}')(G(\overline{z})\ \wedge\ \psi'(\overline{x},\overline{z}))[\overline{x} \leftarrow \overline{a}]\ \}\ \}.$$

If OP is $\Box$ then $f(sem(\tilde{r})) = \{\ \{\ \overline{a} :\ \mathbf{M}_{\mathcal{L}} \models (\forall \overline{z}/\overline{s}')(G(\overline{z})\ \supset\ \psi'(\overline{x},\overline{z}))[\overline{x} \leftarrow \overline{a}]\ \}\ \}$. In the previous expressions $G(\overline{z})$ is the formula of $\mathcal{L}$ obtained from $G(\overline{\omega})$ by substituting $\overline{z}$ for $\overline{\omega}$, and $\psi'(\overline{x},\overline{z})$ is the formula of $\mathcal{L}$ which is obtained from the formula corresponding to ϕ and $\tilde{r}$ by substituting $\overline{z}$ for $\overline{\omega}$.

The theorem implies that queries in modal relational calculus with $\mathcal{L}$-constraints over $\mathcal{L}$-constraint databases can be evaluated in closed form by eliminating quantifiers from a formula of $\mathcal{L}$. The resulting formula in DNF can be turned into an indefinite $\mathcal{L}$-constraint relation which is the answer to the query f.

Example 6.1 The query "Find all rooms that are possibly booked between 4:00 and 5:00" over the database of Example 4.1 can be expressed as follows:

$$\{BOOKED\_4TO5(RN), x/\mathcal{D} : \Diamond(\exists t/\mathcal{Q})(BOOKED(x,t)\ \wedge\ 4 \leq t \leq 5)\}$$

This query can be evaluated by eliminating quantifiers from the following ECL+dePCL formula:

$$\begin{gathered}(\exists \omega/\mathcal{Q})(5 \leq \omega \leq 8\ \wedge\ (\exists t/\mathcal{Q})((x = WP212\ \wedge\ 1 \leq t\ \wedge\ t \leq 2) \vee \\ (x = WP219\ \wedge\ 1 \leq t\ \wedge\ t \leq \omega))\ \wedge\ 4 \leq t\ \wedge\ t \leq 5).\end{gathered}$$

The result is $x = WP212\ \vee\ x = WP219$.

The above theorems will be used in Section 8 to analyze the complexity of query evaluation in TCDB and ITCDB.

7 Quantifier Elimination in Theories of Temporal Constraints

In this section we study the problems of decision and quantifier elimination for theories $Th(\mathbf{Q})$ and $Th(\mathbf{Z})$. Our techniques will be similar to the ones described in [16,17]. The main point of these techniques is that "given a particular theory, one gives an elimination of quantifiers procedure, analyzes it to see how *large* constants can grow, and then uses this analysis (...) to limit quantifiers to range over finite sets instead of an infinite domain" [16].

We will first develop a decision procedure for theory $Th(\mathbf{Q})$. Then we will use this procedure for developing a quantifier elimination algorithm for arbitrary dePCL formulas. Although the algorithms for $Th(\mathbf{Q})$ and $Th(\mathbf{Z})$ are similar, the complexity analysis for $Th(\mathbf{Q})$ is trickier. The results of this section will be used in Section 8 for studying the complexity of query evaluation in TCDB and ITCDB.

7.1 A Decision Procedure for Theory $Th(\mathbf{Q})$

At first we show that we can confine our attention to formulas of dePCL with *only integer* constants (i.e., no rationals). A similar result appears in [3] in the context of checking emptiness of the language of a timed automaton.

Definition 7.1 Let ϕ be a formula of dePCL and $r \in \mathcal{Q}$. Then ϕ_r will denote the formula which is obtained from ϕ by replacing each rational constant c by $r \cdot c$. If $\overline{\tau} \in \mathcal{Q}^n$ then $r \cdot \overline{\tau}$ will denote $(r \cdot \tau_1, \ \ldots, \ r \cdot \tau_n)$.

Lemma 7.1 *Let ϕ be a formula of dePCL with free variables $\overline{t}$. If $r \in \mathcal{Q}$, $r > 0$ and $\overline{\tau} \in \mathcal{Q}^n$ then*

$$\mathbf{Q} \models \phi[\overline{t} \leftarrow \overline{\tau}] \ \text{ iff } \ \mathbf{Q} \models \phi_r[\overline{t} \leftarrow r \cdot \overline{\tau}].$$

Proof: Use induction on the structure of ϕ. ■

The following definitions will be used often in the subsequent discussion.

Definition 7.2 Let i be an integer. We will say that i is *limited* by the positive integer L, denoted by $i \preceq L$, iff $|i| \leq L$.

Definition 7.3 Let r be a rational. We will say that r is *limited* by fraction A/B, denoted by denoted by $i \preceq A/B$, iff its numerator is limited by A and its denominator *is* B.

Example 7.1 The rationals limited by $2/3$ are $-2/3, -1/3, 0, 1/3$ and $2/3$.

The following lemma will be used for estimating the rational constants obtained after any number of quantifier eliminations.

Lemma 7.2 *Let $r_1, \ldots, r_n$ be rationals limited by $A_1/B, \ldots, A_n/B$. Then $r_1 + \cdots + r_n$ is limited by $\frac{A_1+\cdots+A_n}{B}$.*

Definition 7.4 A quantifier Qz is *limited* by the fraction A/B, denoted by $Qz \preceq A/B$, if, instead of ranging over all rational numbers, it ranges over all rationals limited by A/B.

Definition 7.5 If ϕ is a dePCL formula then $maxabs(\phi)$ will denote the maximum absolute value of the integers which appear in ϕ as numerators or denominators. If ϕ involves only integer constants then $maxabs(\phi)$ will denote the maximum absolute value of the integers which appear in ϕ.

Lemma 7.3 *Let ϕ be the following dePCL sentence*

$$(Q_1 t_1)\cdots(Q_K t_K)\psi(t_1,\ldots,t_K).$$

Let us also assume that only integer constants appear in ϕ. Then $\mathbf{Q} \models \phi$ iff $\mathbf{Q} \models \phi'$ where ϕ' is the same as ϕ except that $(Q_i t_i)$ is substituted by

$$(Q_i \preceq \tfrac{2^{K+3i-2}(maxabs(\phi)+1)}{2^i})$$

for all $i = 1, \ldots, K$.

Proof: We will use induction on the order of appearance of the quantifier in ϕ. We assume that ϕ is in prefix normal form (PNF). Note that the proof still goes through when this assumption is dropped.

Base case, $i = 1$. Let us eliminate quantifiers $(Q_2 t_2), \ldots, (Q_K t_K)$ from ϕ using the procedure of Proposition 3.3. It is easy to see that the integer constants in the resulting formula $(Q_1 t_1)\phi'(t_1)$ are limited by $2^{K-1}(maxabs(\phi)+1)$. These integers partition $\mathcal{Q}$ into the following set S of $4 \cdot 2^{K-1}(maxabs(\phi)+1)+3$ distinct intervals:

$$\begin{gathered} S = \{\ (-\infty, 2^{K-1}(maxabs(\phi)+1)),\ (2^{K-1}(maxabs(\phi)+1), \infty)\ \} \cup \\ \{\ [i,i]:\ -2^{K-1}(maxabs(\phi)+1) \le i \le 2^{K-1}(maxabs(\phi)+1)\ \} \cup \\ \{\ (i,i+1):\ -2^{K-1}(maxabs(\phi)+1) \le i \le 2^{K-1}(maxabs(\phi)+1)-1\ \} \end{gathered}$$

The truth value of $\phi'(t_1)$ remains the same for all t_1 in the same interval of S. Therefore we can determine the truth value of $(Qt_1)\phi'(t_1)$ in $\mathbf{Q}$ by determining the truth value of $\phi'(t_1)$ while t_1 ranges over a finite set of representatives, one for each interval of S. The set

$$S_{rep} = \{\ (t_1+t_2)/2 : t_1, t_2 \in \mathcal{Z} \text{ and } t_1, t_2 \preceq 2^{K-1}(maxabs(\phi)+1)\ \} \cup$$

$$\{\ -2^{K-1}(maxabs(\phi)+1)-1,\ 2^{K-1}(maxabs(\phi)+1)+1\ \}$$

is an appropriate set of such representatives. Lemma 7.2 implies that the elements of S_{rep} are rationals limited by $2^{K+1}(maxabs(\phi)+1)/2$.

Inductive step. Let us now assume that the lemma holds for quantifiers $(Q_1t_1),\ldots,(Q_it_i)$. We will show that the lemma holds for $(Q_{i+1}t_{i+1})$ as well. From the induction hypothesis we have $\mathbf{Q}\models\phi$ if and only if $\mathbf{Q}\models\phi_1$ where ϕ_1 is

$$(Q_1t_1\preceq\frac{2^{K+1}(maxabs(\phi)+1)}{2})\cdots(Q_it_i\preceq\frac{2^{K+3i-2}(maxabs(\phi)+1)}{2^i})$$
$$(Q_{i+1}t_{i+1})\cdots(Q_Kt_K)\psi(t_1,\ldots,t_K).$$

Let us eliminate quantifiers $(Q_{i+2}t_{i+2}),\cdots,(Q_Kt_K)$ from ϕ_1 using the procedure of Proposition 3.3 to arrive at the following formula ϕ_2:

$$(Q_1t_1\preceq\frac{2^{K+1}(maxabs(\phi)+1)}{2})\cdots(Q_it_i\preceq\frac{2^{K+3i-2}(maxabs(\phi)+1)}{2^i})$$
$$(Q_{i+1}t_{i+1})\psi_1(t_1,\ldots,t_{i+1}).$$

The truth value of ϕ_2 depends on the truth values of all formulas

$$(Q_{i+1}t_{i+1})\psi_1(\tau_1,\ldots,\tau_i,t_{i+1})$$

where

$$\tau_j\preceq\frac{2^{K+3j-2}(maxabs(\phi)+1)}{2^j},\quad j=1,\ldots,i.$$

The atomic formulas of $(Q_{i+1}t_{i+1})\psi_1(\tau_1,\ldots,\tau_i,t_{i+1})$ are of the form $t_{i+1}\sim r$ or $t_{i+1}\sim\tau_j+r$ where $1\le j\le i$, $\sim$ is $<,\le$ or $=$, τ_j is a rational limited as above and r is an integer limited by $2^{K-(i+1)}(maxabs(\phi)+1)$.

Let us now observe that every rational limited by $2^{K+3j-2}(maxabs(\phi)+1)/2^j$, where $1\le j\le i-1$, is included in the set of rationals limited by

$$2^{K+3i-2}(maxabs(\phi)+1)/2^i.$$

Therefore τ_j+r is a rational limited by $2^{K+3i-1}(maxabs(\phi)+1)/2^i$.

Using an argument similar to the one given for the base case, we can see that the quantifier $(Q_{i+1}t_{i+1})$ can be limited to range over rationals that are the average of two rationals limited by $2^{K+3i-1}(maxabs(\phi)+1)/2^i$, or are one smaller or one larger than all such averages. As a result, $(Q_{i+1}t_{i+1})$ can be limited by $2^{K+3(i+1)-2}(maxabs(\phi)+1)/2^{i+1}$ (from Lemma 7.2). The result follows. ■

Remark 7.1 The models of computation used in the rest of this paper are deterministic and non-deterministic Turing machines with a read-only input tape, a fixed number of read/write work tapes and a write-only output tape where the head can never move left. The *time* of a computation is its length. The *space* of a computation is the number of cells visited on the *work tapes*. Precise definitions of these notions and the associated time and space complexity classes can be found in the standard literature [24].

If $i > 1$ then $\log i$ will denote $\lceil \log_2 i \rceil$. By convention we assume that $\log 1 = 1$.

Formulas will be represented by the standard binary encoding. Integers are written in binary. Rationals are written as fractions of two integers. Variables are chosen from v_0, v_1, v_{10} etc. (i.e., subscripts are written in binary). If ϕ is a formula then $|\phi|$ denotes the length of ϕ.

We are now ready to prove the basic result of this section.

Theorem 7.1 *Let ϕ be a dePCL sentence. The problem of deciding whether $\mathbf{Q} \models \phi$ is PSPACE-complete.*

Proof: *Lower bound.* PSPACE-hardness follows from a straightfoward reduction from QBF [47].

Upper bound. An algorithm for this problem can proceed as follows. First, we transform ϕ into formula ϕ' which involves only integer constants. This can be done by multiplying every fraction p/q of ϕ by the product of all denominators of fractions in ϕ. Therefore every integer in ϕ' will be limited by $maxabs(\phi)^{|\phi|+1}$. We can conclude that

$$|\phi'| \leq |\phi|\,(|\phi|+1)\log(maxabs(\phi)) \leq 2\,|\phi|^2\log(maxabs(\phi)) \leq 2\,|\phi|^3$$

and $maxabs(\phi') \leq maxabs(\phi)^{|\phi|+1}$.

Then we can use the recursive algorithm DEPC-EVAL shown in Figure 2 to decide ϕ'. This algorithm will make use of Lemma 7.3 to limit quantifiers over finite sets of rationals. Every such rational has a numerator limited by $2^{4K-2}(maxabs(\phi')+1)$ and a denominator limited by 2^K where K is the number of quantifiers in ϕ'. Storing the numerator of each one of these rationals requires

$$4K-2+\log(maxabs(\phi')+1) \leq 4K-1+\log(maxabs(\phi)^{|\phi|+1}) \leq$$

Algorithm $\text{DEPC-EVAL}(\phi)$

Input: A dePCL sentence ϕ.
Output: 1 iff $\mathbf{Q} \models \phi$. Otherwise 0.

Method:

(i) If ϕ is an atomic formula of dePCL (with no variables) and $\mathbf{Q} \models \phi$ then return 1, else return 0.
(ii) If ϕ is $\phi_1 \vee \phi_2$ and $\text{DEPC-EVAL}(\phi_1)$ returns 1 then return 1, else return the result of $\text{DEPC-EVAL}(\phi_2)$.
(iii) If ϕ is $\neg\phi'$ and the result of $\text{DEPC-EVAL}(\phi')$ is 1 then return 0, else return 1.
(iv) If ϕ is $(\exists t)\phi'(t)$ then for every rational $r \preceq 2^{4K-2}(maxabs(\phi)+1)/2^K$, call $\text{DEPC-EVAL}(\phi'(r))$. If the result of one of these calls is 1 then return 1 else return 0.

Fig. 2. The algorithm DEPC-EVAL

$$4K - 1 + (|\phi| + 1)\log(maxabs(\phi)) \leq 6\,|\phi|^2$$

bits while storing the denominator requires $O(|\phi|)$ bits. Thus DEPC-EVAL needs $O(|\phi|^3)$ space for keeping track of the current assignment to the quantified variables of ϕ'. No more space is required for bookkeeping. Therefore the total space requirement of DEPC-EVAL is $O(|\phi|^3)$. ■

Remark 7.2 In [29] the space complexity of deciding $Th(\mathbf{Q})$ was given as $O(|\phi|^2 \log|\phi|)$ because we were working under the assumption that any integer constant in a formula ϕ of dePCL is assumed to have size $O(\log|\phi|)$ (Remark 4.1 of [29]). This assumption is now only used in Section 8 when we consider data complexity.

It might be interesting to compare the above result with the following theorems. The first one considers a theory which is less expressive than $Th(\mathbf{Q})$. The second deals with the full first-order theory of real addition with order.

Theorem 7.2 *[15,26] Deciding a sentence of length n in the first order theory of rational order can be done in deterministic space $O(n\log n)$.*

Theorem 7.3 *[4,5] The problem of deciding a sentence of length n in the theory $Th(\mathcal{R},+,<)$ is complete for the class $\bigcup_{k>0} TA[2^{n^k}, n]$.*

The next theorem follows easily from the above discussion. It is also a consequence of the fact that deciding a formula of the same form in the first-order theory of real addition with order is also Σ_k^p-complete [45].

Theorem 7.4 *Let ϕ be $\exists_k$ sentence of dePCL. The problem of deciding whether $\mathbf{Q} \models \phi$ is Σ_k^p-complete.*

A corresponding Π_k^p bound can be established for $\forall_k$ sentences of dePCL.

7.2 Quantifier Elimination in $Th(Q)$

We will now present a quantifier elimination algorithm for open formulas of dePCL. The following lemma will allow us to concentrate on dePCL formulas with only integer constants.

Lemma 7.4 *Let ϕ be a formula of dePCL and ϕ' be a quantifier-free formula equivalent to ϕ. If $r \in \mathcal{Q}$ and $r > 0$ then $\phi' \equiv \psi_{r^{-1}}$ where ψ is a quantifier-free formula equivalent to ϕ_r.*

Proof: Let $\overline{x}$ be the vector of variables in ϕ'. The symbol $\mathcal{Q}^{|\overline{x}|}$ will denote the cartesian product $\mathcal{Q} \times \cdots \times \mathcal{Q}$ ($|\overline{x}|$ times). If $\overline{\chi} \in \mathcal{Q}^{|\overline{x}|}$ then the following equivalences prove the lemma (with help from Lemma 7.1):

$\mathbf{Q} \models \phi'[\overline{x} \leftarrow \overline{\chi}]$ iff $\mathbf{Q} \models \phi[\overline{x} \leftarrow \overline{\chi}]$ iff
$\mathbf{Q} \models \phi_r[\overline{x} \leftarrow r \cdot \overline{\chi}]$ iff $\mathbf{Q} \models \psi[\overline{x} \leftarrow r \cdot \overline{\chi}]$ iff
$\mathbf{Q} \models \psi_{r^{-1}}[\overline{x} \leftarrow \overline{\chi}]$.

■

Now assume we are given a formula ϕ of dePCL. We can find a quantifier-free formula equivalent to ϕ as follows. First we transform ϕ into a formula ϕ_r which has only integer constants by multiplying every fraction by an appropriate integer r. Then we find a quantifier-free formula ψ equivalent to ϕ_r. Finally, we compute $\psi_{r^{-1}}$ which is a quantifier-free formula equivalent to ϕ.

Let us then assume that $\phi(\overline{t})$ is a formula of dePCL which involves only integer constants, and $\overline{t}$ are all the free variables of ϕ. A quantifier free formula ϕ' equivalent to ϕ can be found in the following way. At first, we estimate how large integer constants can grow in the constraints of the answer. Then we use this information to construct a finite partition of the space $\mathcal{Q}^{|\overline{t}|}$ into *regions* with the following properties:

(i) Every region can be represented by a conjunction of dePCL-constraints with only integer constants.

(ii) The truth value of the sentence $\phi(\overline{\tau})$ is the same for all points $\overline{\tau}$ in the same region.

Therefore we can check whether *all* points in a region satisfy $\phi(\overline{t})$ by picking a *single* point $\overline{\tau}$ in the region and checking whether $\mathbf{Q} \models \phi(\overline{\tau})$ is true. The latter check can be done using the above algorithm dePC-Eval. Every conjunction of constraints representing a region for which this check succeeds, becomes a disjunct in the DNF form of ϕ'. Similar techniques have been used in [26].

The technical tools which we will introduce immediately come from the temporal constraint literature [11]. Similar tools have also been used under various names in [42,9] for studying the complexity of query evaluation for Datalog with integer gap-order constraints. Let C be a set (i.e., conjunction) of dePCL or diPCL inequalities in variables $x_1, \ldots, x_n$. The *binary inequality constraint network* (BICN) associated with C is a labeled directed graph $G = (V, E)$ where $V = \{1, \ldots, n\}$. Node i represents variable x_i and edge (i, j) represents the binary constraints involving x_i and x_j. Let us assume that the constraints on $x_j - x_i$ are $x_j - x_i \leq d_{ij}$ and $x_j - x_i \geq -d_{ji}$ where $-d_{ji} \leq d_{ij}$. Then the corresponding BICN N will have an edge $i \rightarrow j$ labeled by the *convex* interval $N_{ij} = [-d_{ji}, d_{ij}]$. Unary constraints are represented with the introduction of a special variable $x_0 = 0$. The notions of solution set, consistency and minimality are defined as usual [12]. If a BICN has $n+1$ nodes (including the 0-th one which corresponds to variable $x_0 = 0$), we will say that it is of *size* n. We will also use a function $Constraints(N)$ which gives the set of binary constraints represented by BICN N. The formal definition of $Constraints$ is omitted.

An alternative graph representation will also be useful. Let C be a set of dePCL or diPCL inequalities in variables $x_1, \ldots, x_n$. The *distance graph* associated with C is a directed labelled graph $G = (V, E)$ where $V = \{1, \ldots, n\}$. Node i represents variable x_i and edge (i, j) represents the binary constraints involving x_i and x_j. If there is a constraint $x_j - x_i \leq d_{ij}$ in C then edge $i \rightarrow j$ of the associated distance graph will be labelled by d_{ij}. The concept of *minimal distance graph* can be defined similarly with the concept of minimal network. Given the minimal network associated with a set of weak inequality constraints, it is trivial to construct the associated minimal distance graph (and vice versa). The set of constraints $Constraints(G)$ represented by a distance graph G is defined as for BICN.

Let us now define a special kind of BICN.

Definition 7.6 A *rational BICN with bounds from the set* $Z \subseteq \mathcal{Z}$ is a consistent BICN with all edges labeled with $[c, c], (c, d), (c, \infty)$ or $(-\infty, c)$ where $c < d$ and $c, d \in Z$.

Let us now define the concept of a formula corresponding to a rational BICN N.

Definition 7.7 Let N be a rational BICN of size n. The formula with free variables $\overline{x} = (x_1, \ldots, x_n)$ corresponding to N is the dePCL formula

$$\bigwedge_{\theta \in Constraints(M)} \theta$$

where M is the minimal network equivalent to N. This formula will be denoted by $\Phi(N)$.

Notation 7.1 If ϕ is a dePCL formula with K quantifiers and vector of free variables $\overline{t}$ then Z_ϕ will denote the set of integers

$$\{i : \ i \in \mathcal{Z} \text{ and } |i| \leq |\overline{t}|\, 2^K maxabs(\phi)\}.$$

The following lemmas help to establish the main result. Lemma 7.5 tells us how to partition the answer space into regions. Subsequent lemmas give properties of rational BICN.

Lemma 7.5 *Let $\phi(\overline{t})$ be a dePCL formula with only integer constants (i.e., no rationals). Then there is a quantifier-free formula $\phi'(\overline{t})$ equivalent to ϕ with the following properties:*

(i) ϕ' is in DNF
(ii) Every disjunct of ϕ' is the dePCL formula corresponding to a rational BICN N of size $|\overline{t}|$ with bounds from the set Z_ϕ.

Proof: Let $\phi_1(\overline{t})$ be the quantifier-free formula

$$\psi_1(\overline{t}) \ \vee \ \cdots \ \vee \ \psi_m(\overline{t})$$

equivalent to ϕ obtained by the algorithm of Proposition 3.3. Let us also assume that the only relation symbol appearing in ψ_1 is $\leq$. It is easy to see that integer constants in ϕ_1 are limited by $2^K maxabs(\phi)$.

Let $N_1, \ldots, N_{m_1}$, $(m_1 \leq m)$ be the minimal BICN corresponding to the consistent ψ_i's $(i = 1, \ldots, m)$. Let us create the formula

$$\phi_2(\overline{t}) \equiv \psi'_1(\overline{t}) \ \vee \ \cdots \ \vee \ \psi'_{m_1}(\overline{t})$$

where

$$\psi'_i = \bigwedge_{\eta \in Constraints(N_i)} \eta, \quad \text{for all } i = 1, \ldots, m_1.$$

Every integer r, such that $t_k - t_l \leq r$ (or $t_k - t_l = r$) is a conjunct in some ψ'_i, is the length of the shortest (*simple*) path connecting nodes k and l in the minimal distance graph for N_i [12]. The number of edges in a simple path is $|\bar{t}|$ and the label of each edge is limited by $2^K maxabs(\phi)$. Therefore every integer in ϕ_2 is limited by $|\bar{t}|\, 2^K maxabs(\phi)$.

We can now obtain a formula equivalent to ϕ_2 in the desired form as follows:

(i) From every ψ'_i construct a conjuction of disjunctions ψ''_i in the following way.
   (a) For every pair of variables t_k, t_l such that $t_k - t_l = r_1$ is in ψ'_i, add $t_k - t_l = r_1$ to ψ''_i.
   (b) For every pair of variables t_k, t_l such that $t_k - t_l \leq r_1$ and $t_k - t_l \geq r_2$ is in ψ'_i, add

$$\bigvee_{r_2 \leq r \leq r_1} t_k - t_l = r$$

   to ψ''_i.
   (c) For every pair of variables such that only $t_k - t_l \leq r_1$ is in ψ'_i, add conjunct

$$t_k - t_l < -\Lambda \ \vee \bigvee_{-\Lambda \leq r \leq r_1} t_k - t_l = r$$

   to ψ''_i where $\Lambda = |\bar{t}|\, 2^K maxabs(\phi)$.
   (d) For every pair of variables such that only $t_k - t_l \geq r_2$ is in ψ'_i, add conjunct

$$\bigvee_{r_2 \leq r \leq \Lambda} t_k - t_l = r \ \vee \ t_k - t_l > \Lambda$$

   to ψ''_i where Λ is as above.
   (e) For every pair of variables $t_k,\ t_l$ such that ψ'_i contains no constraint involving t_k and t_l, add conjunct

$$t_k - t_l < -\Lambda \ \vee \bigvee_{-\Lambda \leq r \leq \Lambda} t_k - t_l = r \ \vee \ t_k - t_l > \Lambda$$

   to ψ''_i where Λ is as above.

(ii) Transform every $\psi''_i(\bar{t})$ in DNF. Let

$$\theta_1(\bar{t}) \ \vee \ \cdots \ \vee \ \theta_n(\bar{t})$$

   be the resulting formula.

(iii) Let $M_1, \ldots, M_{n_1}$, $(n_1 \leq n)$ be the minimal BICN corresponding to consistent θ_i's $(i = 1, \ldots, n)$. The wanted formula $\phi'(\bar{t})$ is

$$\theta'_1(\bar{t}) \ \vee \ \cdots \ \vee \ \theta'_{n_1}(\bar{t})$$

where

$$\theta_i' = \bigwedge_{\eta \in Constraints(M_i)} \eta, \quad \text{for all } i = 1, \ldots, n_1.$$

■

Lemma 7.6 *Let $\overline{\tau}$ be an element of $\mathcal{Q}^n$. For any set $Z \subseteq \mathcal{Z}$, there exists a unique rational BICN N with bounds from Z such that $\mathbf{Q} \models \Phi(N)(\overline{\tau})$.*

Proof: Existence is obvious. For the uniqueness part, we simply observe that $N_1 \neq N_2$ implies that $\Phi(N_1) \ \wedge \ \Phi(N_2)$ is inconsistent. ■

The following lemma allows us to verify the truth of a dePCL formula over a region of the answer space by verifying its truth over a *point* in this region.

Lemma 7.7 *Let $\phi(\overline{t})$ be a dePCL formula with only integer constants (i.e., no rationals). If N is a rational BICN with bounds from the set Z_ϕ then $\mathbf{Q} \models \Phi(N) \ \supset \ \phi$ iff $\mathbf{Q} \models \phi(\overline{\tau})$ for an arbitrary $\overline{\tau}$ such that $\mathbf{Q} \models \Phi(N)(\overline{\tau})$.*

Proof: The "only if" part is trivial so we consider the "if" part. Let us assume that there is $\overline{\tau} \in \mathcal{Q}^{|\overline{t}|}$ such that $\mathbf{Q} \models \Phi(N)(\overline{\tau})$ and $\mathbf{Q} \models \psi(\overline{\tau})$. Let

$$\phi'(\overline{t}) \equiv \theta_1'(\overline{t})) \ \vee \ \cdots \ \vee \ \theta_n'(\overline{t})$$

be the dePCL formula equivalent to ϕ computed as in Lemma 7.5. Then there exists a single disjunct $\theta_i(\overline{t})$ of ϕ' such that $\mathbf{Q} \models \theta_i(\overline{\tau})$. But then θ_i must be $\Phi(N)$ from Lemma 7.6. Therefore $\mathbf{Q} \models \forall(\Phi(N) \ \supset \ \phi)$ since $\Phi(N)$ is a disjunct of ϕ'. ■

The following theorem gives the main result of this section.

Theorem 7.5 *Let ϕ be a dePCL formula. A quantifier-free formula equivalent to ϕ in DNF can be computed in PSPACE.*

Proof: Let us assume that ϕ has K quantifiers and $\overline{t}$ is the vector of all its free variables. We can find a quantifier-free formula equivalent to ϕ as follows. First, we transform ϕ into a formula ϕ' which involves only integer constants. This can be done as in Theorem 7.1: we multiply every fraction of ϕ by the product P of all denominators of fractions in ϕ.

Secondly, we generate, one by one, all rational BICN of size $|\overline{t}|$ with bounds from

$$Z_{\phi'} = \{i : \ i \in \mathcal{Z} \text{ and } \ |i| \leq |\overline{t}| \, 2^K maxabs(\phi)^{|\phi|+1}\}$$

(since $maxabs(\phi') \leq maxabs(\phi)^{|\phi|+1}$). For each BICN N, we find a solution $\overline{\tau}$ of N (using the minimal network M) and check whether $\mathbf{Q} \models \phi'(\overline{\tau})$ using algorithm DEPC-EVAL of Theorem 7.1. If this check succeeds then we divide each constant of $\Phi(N)$ by P and use the result to form a disjunct of the returned formula. The correctness of this procedure follows from the previous lemmas. It is not difficult to see that the above algorithm can be implemented by a deterministic Turing machine in PSPACE [32]. ■

7.3 Quantifier Elimination in Theory $Th(\mathbf{Z})$

The decision and quantifier elimination problems for theory $Th(\mathbf{Z})$ can also be handled with similar techniques [32].

Theorem 7.6 *If ϕ is a diPCL sentence then the problem of deciding whether $\mathbf{Z} \models \phi$ is PSPACE-complete. If ϕ is a $\exists_k$ sentence of diPCL then this problem Σ_k^p-complete. If ϕ is a diPCL formula then a quantifier-free formula equivalent to ϕ in DNF can be computed in PSPACE.*

Proof: In [29] we showed that $Th(\mathbf{Z})$ can be decided in $O(|\phi|^2)$ space.

After we wrote [29], we discovered that this actually follows immediately from Theorem 5.32 of [17] where the theory of natural numbers with the successor function is considered. The bound for the quantifier elimination problem can also be found in [29]. ■

Let us observe that a sentence of length n in $Th(\mathcal{Z}, +, <)$ can be decided by a deterministic Turing machine in space $O(2^{2^{cn}})$ for some constant c [16]. More precisely, the decision problem for $Th(\mathcal{Z}, +, <)$ is complete for the class $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$ [4,5]. In addition, there exist constants $d, e > 0$ such that a deterministic Turing machine can decide a sentence of PA with length $n > 4$ and at most k alternations of quantifiers in $O(2^{dn^{k+4}})$ space and $O(2^{2^{en^{k+4}}})$ time [40]. Finally, [19] shows that there is a constant $c > 0$ such that no $2^{(n/m)^{ck}}$ time-bounded non-deterministic Turing machine can decide a Presburger arithmetic formula of size n and quantifier alternation depth k.

Figures 3 and 4 summarize the best known complexity bounds for decision and quantifier elimination for the various theories discussed in this section. The first figure considers arbitrary formulas of size n. The second figure considers formulas of size n with k (fixed) alternations of quantifiers. For the theory $Th(\mathbf{Z})$ our main contribution is a quantifier elimination procedure which complements the decision procedure given in [17]. For the theory $Th(\mathbf{Q})$ our main contribution is a decision procedure with complexity DSPACE(n^3), and also a quantifier elimination procedure.

Theory	Upper bound	Lower bound
$Th(\mathcal{Z}, +, <)$	DSPACE($2^{2^{cn}}$) Ferrante-Rackoff, 75	NTIME($2^{2^{cn}}$) Fischer-Rabin, 74
	TA($2^{2^{cn}}, n$)-complete, Berman, 80	
$Th(\mathcal{R}, +, <)$	DSPACE(2^{cn}) Ferrante-Rackoff, 75	NTIME(2^{cn}) Fischer-Rabin, 74
	TA($2^{cn}, n$)-complete, Berman, 80	
$Th(\mathbf{Z})$	DSPACE(n^2) Ferrante-Rackoff, 79 Koubarakis, 94	NSPACE(cn) Ferrante-Rackoff, 79
$Th(\mathbf{Q})$	DSPACE(n^3) Koubarakis, 94	NSPACE(cn) Ferrante-Rackoff, 79
$Th(\mathcal{Q}, <)$	DSPACE($n \log n$) Ferrante-Geiser, 77 Kanellakis et. al., 90	NSPACE($cn^{1/2}$) Stockmeyer, 77

Fig. 3. Complexity bounds for various logical theories

Theory	Upper bound	Lower bound
$Th(\mathcal{Z}, + <)$	DSPACE($2^{cn^{k+4}}$) Reddy-Loveland, 78	NTIME($2^{(n/k)^{ck}}$) Fürer, 82
$Th(\mathcal{R}, +, <)$	Σ_k^p-complete Sontag, 85	
$Th(\mathbf{Z})$ $Th(\mathbf{Q})$	Σ_k^p-complete Koubarakis, 94	

Fig. 4. Complexity bounds for various logical theories (fixed number of quantifier alternations)

8 The Complexity of Query Evaluation in TCDB and ITCDB

In this section we study the complexity of query evaluation in TCDB and ITCDB. We use the results of Section 6 to transform any query answering problem to a quantifier-elimination problem and then exploit the techniques of Section 7 to derive the complexity bounds.

The complexity of database query evaluation is usually measured using the notions of data complexity and combined complexity introduced in [51,6]. These notions can be defined as follows:

- *Combined complexity.* In this case we measure the complexity of query evaluation as a function of the size of the database, the size of the schema and the size of the query.
- *Data complexity.* In this case we measure the complexity of query evaluation as a function of the database size only; the query and the schema are considered *fixed.* We also assume that the size of any integer constant in the database is *logarithmic* in the size of the database. When a database is viewed as a formula of dePCL or diPCL, this is equivalent to assuming that any integer constant in a formula ϕ is assumed to have size $O(\log |\phi|)$. This

assumption has also been made in [26] and [42].

8.1 Temporal Constraint Databases

We first consider the case of TCDB (i.e., no indefinite information is allowed). The following theorem deals with yes-no queries in relational calculus with dePCL-constraints over dePCL-constraint databases. It is an easy consequence of Theorems 6.1, 7.1 and 7.4.

Theorem 8.1 *Let $\tilde{r}$ be a dePCL-constraint database and f be a yes/no query in relational calculus with dePCL-constraints. The problem of deciding whether $f(\tilde{r}) = yes$ has PSPACE-complete combined complexity. If f is a $\exists_k$ yes/no query then this problem has Σ_k^p-complete combined complexity.*

Proof: If f is an arbitrary yes/no query then $f(\tilde{r})$ can be evaluated as follows. First, we tranform f into PNF. Secondly, we form the formula ϕ of dePCL which corresponds to f and $\tilde{r}$. Then we use the quantifier elimination algorithm DEPCL-EVAL to decide whether $\mathbf{Q} \models \phi$. Theorem 7.1 shows that this computation can be carried out in PSPACE. The lower bound follows easily from a reduction from QBF [47]. If f is a $\exists_k$ yes/no query the result follows from Theorem 7.4. ■

For data complexity we cannot achieve a LOGSPACE bound using the decision procedure for $Th(\mathbf{Q})$ developed in Section 7. The problem is that we might need at most $O(|\phi| \log |\phi|)$ space for storing the product of all denominators of fractions in some dePCL formula ϕ. However, under the data complexity measure, the decision procedure of [16] for the theory of real addition with order can be implemented in LOGSPACE. [4] Therefore, we have the following result.

Theorem 8.2 *Let $\tilde{r}$ be a dePCL-constraint database and f be a yes/no query in relational calculus with dePCL-constraints. The problem of deciding whether $f(\tilde{r}) = yes$ has LOGSPACE data complexity.*

Proof: We will use the decision procedure of [16] for the theory of real addition with order. The following are some important observations:

(i) Lemma 3 and Theorem 2 of [16] assume that the input formula is in PNF. These results still hold even if we do not make this assumption.
(ii) In Theorem 2 of [16] space is measured in terms of the length of the input formula. In our case it is enough to measure space in terms of number of

[4] But not with PSPACE combined complexity! This is why we developed the procedure of Theorem 7.1.

quantifiers.

(iii) As in [27], we do not construct the entire formula corresponding to a query and a database. We will simply scan the query and switch to the database whenever needed.

The procedure of [16] works like DEPC-EVAL by substituting unbounded quantifiers by quantifiers ranging over finite sets of rationals. If ϕ is the input formula, these rationals are limited by $maxabs(\phi)^{2(c_1+1)n}$ where c_1 is a constant and n is the number of quantifiers in the input formula (Theorem 2, [16]). The amount of space needed to write down these rationals (e.g., $2^{(c_1+1)n}\log(maxabs(\phi))$) dominates the amount of space used by the procedure. Under the data complexity measure the total space requirements of the procedure are $O(\log|\phi|)$. ■

We will now consider queries with free variables. Let us assume we are given dePCL-constraint database $\tilde{r}$ and query $\{R(\overline{T}), \overline{t}/\mathcal{Q}: \ \phi(\overline{t})\}$ in relational calculus with dePCL-constraints. At first, using lemma 8.2 below, we estimate how large rational constants can grow in the constraints of the answer. Secondly, we use this information to construct a finite partition of the space $\mathcal{Q}^{|\overline{t}|}$ into *regions* with the following properties:

(i) Every region can be represented by a conjunction of dePCL-constraints.

(ii) The truth value of the sentence $\phi(\overline{\tau})$ is the same for all points $\overline{\tau}$ in the same region.

Then, using the algorithm of [16], we can check whether *all* points in a region satisfy $\phi(\overline{t})$ by picking a *single* point in the region and checking whether $\mathbf{Q} \models \phi(\overline{\tau})$ is true.

For the purposes of this section we have to *redefine* the term "limited" for rational numbers. The meaning of "limited" is now as in [16], and it is *different* than the one in Definition 7.4.

Definition 8.1 Let r be a rational number. We will say that r is *limited* by the positive integer L, denoted by $r \preceq L$, iff there exist integers a and b such that $r = a/b$ and $|a| \leq L$ and $|b| \leq L$.

Example 8.1 The rationals limited by 2 are $-2, -1, -1/2, 0, 1/2, 1$ and 2.

In this section the following lemma of [16] can be used to estimate the size of rationals produced after eliminating one or more quantifiers. The proof is trivial.

Lemma 8.1 *Let $r_1, \ldots, r_n$ be rational numbers limited by the positive integers $L_1, \ldots, L_n$. Then*

$$r_1 + r_2 \cdots + r_n \preceq kL_1L_2 \cdots L_n \text{ and } r_1r_2 \cdots r_n \preceq L_1L_2 \cdots L_n.$$

The following lemma estimates how large rational constants can grow in the quantifier-free formula equivalent to a given dePCL formula ϕ.

Lemma 8.2 *Let $\phi(\overline{t})$ be a dePCL formula with K quantifiers and $\phi'(\overline{t})$ be the quantifier free formula equivalent to $\phi(\overline{t})$ which is computed by the algorithm of proposition 3.3. Every rational constant of ϕ' is limited by*

$$2^{2^K(\log(maxabs(\phi))+1)}.$$

Proof: If r_1, r_2 are rationals such that $r_1, r_2 \preceq L$ then $r_1+r_2 \preceq 2L^2$. Thus after the first quantifier of ϕ is eliminated the rational constants of the resulting formula are limited by $2maxabs(\phi)^2$. The result follows by a simple inductive argument. ■

Let us now introduce the machinery required for presenting our method. We first define the concept of a rational BICN and its dePCL formula. These definitions are *different* than Definitions 7.6 and 7.7. Now we allow bounds to be rationals (not just integers).

Definition 8.2 Let Q be a set such that $Q \subseteq \mathcal{Q}$. A *rational BICN with bounds from the set Q* is a consistent BICN with all edges labeled with $[c, c], (c, d), (c, \infty)$ or $(-\infty, c)$ where $c < d$ and $c, d \in Q$.

Definition 8.3 Let N be a rational BICN of size n. The formula with free variables $\overline{x} = (x_1, \ldots, x_n)$ corresponding to N is the dePCL formula

$$\bigwedge_{\theta \in Constraints(M)} \theta$$

where M is the minimal network equivalent to N. This formula will be denoted by $\Phi(N)$.

Notation 8.1 If ϕ is a dePCL formula with K quantifiers and vector of free variables $\overline{t}$ then Q_ϕ will denote the set of rationals

$$\{r : \ r \in \mathcal{Q} \text{ and } r \preceq |\overline{t}| \, 2^{2^K(\log(maxabs(\phi))+1)}\}.$$

The following lemmas help to establish the main result. The similarity with the development of Section 7.2 is obvious. Lemma 8.3 tells us how to partition the answer space into regions. Subsequent lemmas give properties of rational BICN. The proofs of some results are omitted.

Lemma 8.3 *If $\phi(\bar{t})$ is a dePCL formula then there is a quantifier-free formula $\phi'(\bar{t})$ equivalent to ϕ with the following properties:*

(i) ϕ' is in DNF
(ii) Every disjunct of ϕ' is the dePCL formula corresponding to a rational BICN N of size $|\bar{t}|$ with bounds from the set Q_ϕ.

Proof: The proof is almost identical to the proof of Lemma 7.5. First we eliminate quantifiers from $\phi(\bar{t})$ to arrive at formula $\phi_1(\bar{t})$ with rationals limited by $2^{2^{K(\log(maxabs(\phi)+1))}}$ (Lemma 8.2). Then we can compute ϕ' as in Lemma 7.5. ■

Lemma 8.4 *Let $\bar{\tau}$ be an element of $\mathcal{Q}^n$. For any set $Q \subseteq \mathcal{Q}$, there exists a unique rational BICN N with bounds from Q such that $\mathbf{Q} \models \Phi(N)(\bar{\tau})$.*

Lemma 8.5 *Let $\phi(\bar{t})$ be a dePCL formula and N be a rational BICN with bounds from the set Q_ϕ. Then $\mathbf{Q} \models \Phi(N) \supset \phi$ iff $\mathbf{Q} \models \phi(\bar{\tau})$ is true for an arbitrary $\bar{\tau}$ such that $\mathbf{Q} \models \Phi(N)(\bar{\tau})$.*

Lemma 8.6 *Let N be a consistent minimal BICN with bounds rationals limited by L. Then for every solution $\tau_1, \ldots, \tau_n$ of N and $i = 1, \ldots, n$,*

$$\tau_i \preceq 2^{2^{i+1}+2^i-4} L^{2^{i+1}-2}.$$

Proof: Since N is minimal, a solution can be found by *backtrack-free* search as follows [12]. We initially assign the value 0 to t_0. Then we assign to t_1 any value which satisfies the constraints involving t_1 and t_0. We proceed in the same fashion with t_2, t_3 and so on. It is easy to see that $\tau_1 \preceq 2^2 L^2$, $\tau_2 \preceq 2^8 L^6$, and so on. The result can be proved by induction. ■

The following theorem gives the main result of this section.

Theorem 8.3 *Let $\tilde{r}$ be a dePCL-constraint database and f be an open query in relational calculus with dePCL-constraints. A dePCL-constraint relation r such that $r = f(\tilde{r})$ can be computed with LOGSPACE data complexity.*

Proof: Let f be $\{R(\overline{T}), \bar{t}/\mathcal{Q} : \phi(\bar{t})\}$. Let us compute the dePCL formula $\phi'(\bar{t})$ corresponding to f and $\tilde{r}$. We can now find a quantifier-free formula equivalent to ϕ' in DNF as follows. We generate one by one all rational BICN of size $|\bar{t}|$ with bounds from Q_ϕ. For every such BICN N, we find a solution $\bar{\tau}$ of N and check whether $\mathbf{Q} \models \phi(\bar{\tau})$ using the decision procedure of [16] for the first-order theory of real addition with order. If this check succeeds then we form a tuple of r from the formula $\Phi(N)$ corresponding to N. The correctness of this procedure follows from the above lemmas.

The above procedure starts with BICN N with bounds rationals limited by $|\bar{t}|\, 2^{2^{K \log(maxabs(\phi)+1)}}$. If N is not minimal then it can be transformed into one with bounds limited by $L_1 = |\bar{t}|\, 2^{|\bar{t}| 2^{K \log(maxabs(\phi)+1)}}$. From Lemma 8.6, we can see that any solution of N will have components limited by

$$L_2 = 2^{2^{|\bar{t}|+1} + 2^{|\bar{t}|} - 4} L_1^{2^{|\bar{t}|+1} - 2}.$$

To write down any of these components we need

$$(2^{|\bar{t}|+1} + 2^{|\bar{t}|} - 4) + (2^{|\bar{t}|+1} - 2) \log L_1$$

space. Under the data complexity assumption, this amount of space is logarithmic in the size of the database. As in Theorem 8.2 we can also see that the rest of the computation can be carried out with LOGSPACE data complexity.
■

Let us now consider queries in relational calculus with diPCL constraints over diPCL-constraint databases. The following theorem is a consequence of Theorems 6.1 and 7.6.

Theorem 8.4 *Let $\tilde{r}$ be a diPCL-constraint database. If f is a yes/no query in relational calculus with diPCL-constraints then the problem of deciding whether $f(\tilde{r}) = yes$ can be solved with LOGSPACE data complexity and PSPACE-complete combined complexity. If f is a $\exists_k$ yes/no query then this problem has Σ_k^p-complete combined complexity. If f is an open query then a diPCL-constraint relation r such that $r = f(\tilde{r})$ can be computed with LOGSPACE data complexity.*

Remark 8.1 Grumbach, Su and Tollu have recently improved the LOGSPACE data complexity bound of the above theorem [22]. In this case, it follows from the main theorem of [22] that query evaluation can be done in AC^0 (the class of functions computable in constant time with a polynomial amount of hardware [24]).

8.2 Indefinite Temporal Constraint Databases

Let us now turn our attention to query evaluation in ITCDB. The following theorems demonstrate that evaluation of queries in modal relational calculus with temporal constraints over ITCDB is probably intractable.

Theorem 8.5 *Let $\mathcal{L}$ be dePCL or diPCL. Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database and f be a yes/no $\Diamond$-query in modal relational calculus with $\mathcal{L}$-constraints. The problem of deciding whether $f(\tilde{r}) = yes$ is NP-complete for data complexity and PSPACE-complete for combined complexity.*

Proof: *Upper bound.* Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database, $\overline{\omega}$ be the vector of e-variables of sort $\overline{s}$ in $\tilde{r}$, $G(\overline{\omega})$ be the global condition of $\tilde{r}$ and f be the query $\{: \Diamond\phi\}$. Theorem 6.2 tells us that $f(\tilde{r}) = yes$ iff

$$\mathbf{M}_{\mathcal{L}} \models (\exists \overline{\omega}/\overline{s})(G(\overline{\omega}) \ \wedge \ \psi(\overline{\omega})) \tag{1}$$

where ψ is the formula of $\mathcal{L}$ corresponding to ϕ and $\tilde{r}$. For combined complexity, the upper bound follows immediately from Theorems 7.6 and 7.1.

Let us now consider data complexity. For indefinite dePCL-constraint databases, if $\overline{\omega} = (\omega_1, \ldots, \omega_N)$ then Lemma 7.3 tells us that (1) is equivalent to

$$\begin{aligned}
\mathbf{Q} \models \ & (\exists \omega_1 \preceq \frac{2^{N+K+1}(maxabs(G \ \wedge \ \phi) + 1)}{2}) \\
& (\exists \omega_2 \preceq \frac{2^{N+K+4}(maxabs(G \ \wedge \ \phi) + 1)}{4}) \\
& \quad \cdots \\
& (\exists \omega_N \preceq \frac{2^{4N+K-2}(maxabs(G \ \wedge \ \phi) + 1)}{2^N}) \\
& \quad (G(\omega_1, \ldots, \omega_N) \ \wedge \ \psi(\omega_1, \ldots, \omega_N)).
\end{aligned}$$

Therefore a non-deterministic Turing machine can guess values $\beta_1, \ldots, \beta_N$ for $\omega_1, \ldots, \omega_N$ in the ranges specified above, and then verify whether

$$\mathbf{Q} \models G(\beta_1, \ldots, \beta_N) \ \wedge \ \psi(\beta_1, \ldots, \beta_N).$$

The guessing step can be done in polynomial time because the sizes of $\beta_1, \ldots, \beta_N$ do not exceed $(4N+K-1)\log(maxabs(G \wedge \psi))+N$ bits. The verification step can also be done in polynomial time (Theorem 8.1). Therefore the whole computation can be performed in polynomial time by a non-deterministic Turing machine. The proof for diPCL is similar.

Lower bounds. The PSPACE lower bound is obvious. The problem of query evaluation in relational calculus with $\mathcal{L}$-constraints over $\mathcal{L}$-constraint databases has PSPACE-complete combined complexity (Theorems 8.4 and 8.1). The data complexity lower bounds follow from [50]. ∎

The following theorem is an easy consequence of the previous one.

Theorem 8.6 *Let $\mathcal{L}$ be dePCL or diPCL. Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database and f be a yes/no $\Box$-query in modal relational calculus with $\mathcal{L}$-constraints. The problem of deciding whether $f(\tilde{r}) = yes$ is co-NP-complete for data complexity and PSPACE-complete for combined complexity.*

For the case of queries with a fixed alternation of quantifiers, we have the following result.

Theorem 8.7 *Let $\mathcal{L}$ be dePCL or diPCL. Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database and f be a $\exists_k$ yes/no $\Box$-query in modal relational calculus with $\mathcal{L}$-constraints. The problem of deciding whether $f(\tilde{r}) = yes$ is Π^p_{k+1}-complete for combined complexity.*

Proof: Let $\tilde{r}$ be an indefinite $\mathcal{L}$-constraint database, $\overline{\omega}$ be the vector of e-variables of sort $\overline{s}$ in $\tilde{r}$, $G(\overline{\omega})$ be the global condition of $\tilde{r}$ and f be the query $\{: \Box\phi\}$. Theorem 6.2 tells us that $f(\tilde{r}) = yes$ iff

$$\mathbf{M}_{\mathcal{L}} \models (\forall \overline{\omega}/\overline{s})(G(\overline{\omega}) \supset \psi(\overline{\omega}))$$

where ψ is the formula of $\mathcal{L}$ corresponding to ϕ and $\tilde{r}$. Thus we have to decide a $\forall_{k+1}$ formula of $\mathcal{L}$. The upper bounds now follow from Theorems 7.6 and 7.4.

Lower bounds. The lower bound follows easily from reductions from $\mathrm{QBF}^{\forall}_{\mathbf{k+1}}$. ■

The results of this section are summarized in the tables of Figure 1. In [32] we also study the complexity of query evaluation in (indefinite) $\mathcal{L}$-constraint databases when $\mathcal{L}$ is ECL+dePCL and ECL+diPCL. The addition of ECL into the picture does not alter the complexity bounds. ECL-constraint databases were first studied in [26] where a PSPACE quantifier elimination algorithm for ECL was presented. This algorithm is very similar to the quantifier elimination algorithms for diPCL and dePCL given in Section 7. [32] shows how to combine these algorithms to achieve quantifier elimination for ECL+diPCL and ECL+dePCL in PSPACE. For data complexity, these algorithms can easily be seen to be in LOGSPACE.

9 Related Work and Future Research

The complexity of query answering in ITCDB has also been studied in [50]. Van der Meyden considers *indefinite order databases*. Indefinite order databases do not contain u-variables (no infinite information) and their global condition consists only of constraints of the form $\omega_1 < \omega_2$ where $<$ is a linear order, and ω_1 and ω_2 are order constants or e-variables. [50] concentrated on the problem of deciding whether a yes/no positive existential query is true in all possible worlds represented by an indefinite order database. This problem has co-NP-complete data complexity and Π_2^p-complete combined complexity. To obtain tractability [50] investigates indefinite order databases with monadic proper predicates. In this case the data complexity of conjunctive queries is in linear time but the combined complexity is co-NP-hard. Finally, [50] investigated several specializations of this case where the combined complexity is in PTIME as well. The complexity bounds hold independently of whether $<$ is interpreted as a discrete, dense or finite linear order. These results seem to point out that indefinite order databases will be very hard to implement in their full generality.

The research reported in [46] is also closely related to our own. [46] investigates the use of constraints for the representation of infinite and indefinite information in a simple object-oriented model. The authors do not consider the properties of an abstract constraint-object model; instead, they concentrate on a special case which uses order constraints and simple set constraints (e.g., constraints using $\in$, $\subseteq$ but not $\cap$ or $\cup$). The important contribution of this paper is the study of quantifier-elimination for this simple class of set constraints and the development of a Datalog-like query language which incorporates them. This query language is intended for the expression of "certainty" queries only; "possibility" queries are not considered. In a similar spirit [13,34] have considered an extension of Datalog with null values, and presented a top-down query evaluation method.

[8] is an interesting survey on temporal query languages from a constraint-based perspective. Infinite and indefinite information is discussed in substantial depth and comparisons with other "more conventional" temporal query languages are presented. [49] have also made an important step forward by studying temporal deductive databases with order, distance and periodicity constraints. However, they do not consider indefinite information.

In other related work [2,21,22,38] study the expressive power of query languages with linear and polynomial constraints. The main result of [22] which we discussed in Section 8.1 is an important contribution from our point of view.

In our current research we study query languages based on first-order constraint languages more expressive than diPCL and dePCL. The interesting question is whether the results of [32] carry over to the new classes. Another important question is the study of tractable cases of indefinite diPCL/dePCL-constraint databases. The work of [50] can serve as the basis for such an effort. Finally, it would be interesting to consider Datalog-like query languages for ITCDB and TCDB.

Acknowledgements

The work presented in this paper was performed while the author was at the National Technical University of Athens. I would like to thank Timos Sellis for his support, and Stathis Zachos for many interesting discussions that improved my understanding of complexity theory. I am also grateful to Paris Kanellakis and Peter Revesz for answering my questions concerning constraint databases. Finally, I would like to thank the reviewers whose comments greatly improved the presentation of the paper.

References

[1] S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science*, 78(1):159–187, 1991.

[2] F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper. Linear vs. Polynomial Constraints in Database Query Languages. In A. Borning, editor, *Proceedings of the 2nd International Workshop on the Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 181–192. Springer Verlag, 1994.

[3] R. Alur. *Techniques for Automatic Verification of Real-Time Systems.* PhD thesis, Dept. of Computer Science, Stanford University, 1991.

[4] L. Berman. The Complexity of Logical Theories. *Theoretical Computer Science*, 11:71–78, 1980.

[5] A.R. Bruss and A.R. Meyer. On Time-Space Classes and their Relation to the Theory of Real Addition. *Theoretical Computer Science*, 11:59–69, 1980.

[6] A. Chandra and D. Harel. Structure and Complexity of Relational Queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[7] C.C. Chang and H.J. Keisler. *Model Theory.* North Holland, 1973.

[8] J. Chomicki. Temporal Query Languages: A Survey. In Gabbay, D. and Ohlbach, H., editor, *Temporal Logic, First International Conference (1994)*, volume 827 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, 1994.

[9] J. Cox and K. McAloon. Decision Procedures for Constraint Based Extensions of Datalog. Technical Report 90-09, Dept. of Computer and Information Sciences, Brooklyn College of C.U.N.Y., 1990.

[10] J. Cox and K. McAloon. Decision Procedures for Constraint Based Extensions of Datalog. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research.* MIT Press, 1993. Originally appeared as

Technical Report No. 90-09, Dept. of Computer and Information Sciences, Brooklyn College of C.U.N.Y.

[11] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. In R. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 83–93, Toronto, Ontario, 1989.

[12] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. Special Volume on Knowledge Representation.

[13] F. Dong and V.S. Lakshmanan. Deductive Databases with Incomplete Information. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, November 1992.

[14] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[15] J. Ferrante and J.R. Geiser. An Efficient Decision Procedure for the Theory of Rational Order. *Theoretical Computer Science*, 4(2):227–233, 1977.

[16] J. Ferrante and C. Rackoff. A Decision Procedure for the First Order Theory of Real Addition with Order. *SIAM Journal on Computing*, 4(1):69–76, 1975.

[17] J. Ferrante and C. Rackoff. *The Computational Complexity of Logical Theories*. Lecture Notes in Mathematics. Springer Verlag, 1979.

[18] M.J. Fischer and M.O. Rabin. Super-exponential Complexity of Presburger Arithmetic. In *Proc. of AMS Symposium on Complexity of Real Computational Processes*, volume III, 1974.

[19] M. Furer. The Complexity of Presburger Arithmetic with Bounded Quantifier Alternation Depth. *Theoretical Computer Science*, 18:105–111, 1982.

[20] G. Grahne. The Problem of Incomplete Information in Relational Databases. Technical Report Report A-1989-1, Department of Computer Science, University of Helsinki, Finland, 1989. Also published as Lecture Notes in Computer Science 554, Springer Verlag, 1991.

[21] S. Grumbach and J. Su. Finitely representable databases. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 289–300, 1994.

[22] S. Grumbach, J. Su, and C. Tollu. Linear constraint databases. In D. Leivant, editor, *Proceedings of the Logic and Computational Complexity Workshop*, Indianapolis, 1994. Springer Verlag. To appear in LNCS.

[23] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984.

[24] D.S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. North-Holland, 1990.

[25] P.C. Kanellakis and D. Goldin. Constraint Programming and Database Query Languages. In *Proceedings of Theoretical Aspects of Computer Software (TACS)*, volume 789 of *Lecture Notes in Computer Science*, pages 96–120. Springer-Verlag, April 1994.

[26] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 299–313, 1990.

[27] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[28] M. Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. In *Proceedings of the 9th International Conference on Data Engineering*, pages 327–334, April 1993.

[29] M. Koubarakis. Complexity Results for First-Order Theories of Temporal Constraints. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, pages 379–390. Morgan Kaufmann, San Francisco, CA, May 1994.

[30] M. Koubarakis. Database Models for Infinite and Indefinite Temporal Information. *Information Systems*, 19(2):141–173, March 1994.

[31] M. Koubarakis. Foundations of Indefinite Constraint Databases. In A. Borning, editor, *Proceedings of the 2nd International Workshop on the Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 266–280. Springer Verlag, 1994.

[32] M. Koubarakis. *Foundations of Temporal Constraint Databases*. PhD thesis, Computer Science Division, Dept. of Electrical and Computer Engineering, National Technical University of Athens, February 1994. Available electronically from `http://www.co.umist.ac.uk/~manolis/MK/M-Koubarakis.html`.

[33] M. Koubarakis. Databases and Temporal Constraints: Semantics and Complexity. In J. Clifford and A. Tuzhilin, editors, *Recent Advances in Temporal Databases (Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, September 1995)*, Workshops in Computing, pages 93–109. Springer, 1995.

[34] V.S. Lakshmanan. Evolution of Intelligent Database Systems: A Personal Perspective. In V.S. Alagar, S. Bergler, and F.Q. Dong, editors, *Incompleteness and Uncertainty in Information Systems (Proceedings of the SOFTEKS Workshop on Incompleteness and Uncertainty in Information Systems, Concordia University, Montreal, Canada, October 1993)*, Workshops in Computing, pages 189–208. Springer, 1993.

[35] H.J. Levesque. Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, 23:155–212, 1984.

[36] Witold Jr. Lipski. On Semantic Issues Connected with Incomplete Information Databases. *ACM Transcactions on Database Systems*, 4(3):262–296, September 1979.

[37] M. Maher. A Logic Programming View of CLP. In *Proceedings of the 10th International Conference on Logic Programming*, pages 737–753, 1993.

[38] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 279–288, 1994.

[39] M.O. Rabin. Decidable theories. In *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 595–629. North-Holland, 1977.

[40] C.R. Reddy and D.W. Loveland. Presburger Arithmetic with Bounded Quantifier Alternation. In *Proc. of ACM Symposium on the Theory of Computing*, pages 320–325, 1978.

[41] R. Reiter. On Integrity Constraints. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 97–111, Asilomar, CA, 1988.

[42] P.Z. Revesz. A Closed Form for Datalog Queries with Integer Order. In *Proceedings of the 3rd International Conference on Database Theory*, pages 187–201, 1990.

[43] P.Z. Revesz. A Closed Form Evaluation for Datalog Queries with Integer (Gap)-Order Constraints. *Theoretical Computer Science*, 116(1):117–149, 1993.

[44] A. Schrijver, editor. *Theory of Integer and Linear Programming.* Wiley, 1986.

[45] E. Sontag. Real Addition and the Polynomial Time Hierarchy. *Information Processing Letters*, 20:115–120, 1985.

[46] D. Srivastava, R. Ramakrishnan, and P. Revesz. Constraint Objects. In A. Borning, editor, *Proceedings of the 2nd International Workshop on the Principles and Practice of Constraint Programming (PPCP'94)*, volume 874 of *Lecture Notes in Computer Science*, pages 218–228. Springer Verlag, 1994.

[47] L.J. Stockmeyer. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[48] P.J. Stuckey. Constructive Negation for Constraint Logic Programming. In *Proceedings of Symposium on Logic in Computer Science*, pages 328–339, 1991.

[49] D. Toman, J. Chomicki, and D.S. Rogers. Datalog with Integer Periodicity Constraints. In *Proceedings of the International Symposium on Logic Programming*, pages 189–203, 1994.

[50] R. van der Meyden. The Complexity of Querying Indefinite Data About Linearly Ordered Domains (Preliminary Version). In *Proceedings of the 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 331–345, 1992. Full version to appear in JCSS.

[51] M. Vardi. The Complexity of Relational Query Languages. In *Proceedings of ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 137–146, 1982.

[52] M. Vardi. Querying Logical Databases. *Journal of Computer and System Sciences*, 33:142–160, 1986.