Tractable Query Answering in Indefinite Constraint Databases: Basic Results and Applications to Querying Spatiotemporal Information*

Manolis Koubarakis[1] and Spiros Skiadopoulos[2]

[1] Dept. of Informatics, University of Athens
Panepistimioupolis, TYPA Buildings
157 81 Athens, Greece
manolis@di.uoa.gr
www.di.uoa.gr/~manolis
[2] Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73 Athens, Greece
spiros@dbnet.ece.ntua.gr

Abstract. We consider the scheme of indefinite constraint databases proposed by Koubarakis. This scheme can be used to represent indefinite information arising in temporal, spatial and truly spatiotemporal applications. The main technical problem that we address in this paper is the discovery of tractable classes of databases and queries in this scheme. We start with the assumption that we have a class of constraints $\mathcal{C}$ with satisfiability and variable elimination problems that can be solved in PTIME. Under this assumption, we show that there are several general classes of databases and queries for which query evaluation can be done with PTIME data complexity. We then search for tractable instances of $\mathcal{C}$ in the area of temporal and spatial constraints. Classes of constraints with tractable satisfiability problems can be easily found in the literature. The largest class that we consider is the class of Horn disjunctive linear constraints over the rationals. Because variable elimination for Horn disjunctive linear constraints cannot be done in PTIME, we try to discover subclasses with tractable variable elimination problems. The class of UTVPI$^{\neq}$ constraints is the largest class that we show to have this property. Finally, we restate the initial general results with $\mathcal{C}$ ranging over the newly discovered tractable classes. Tractable query answering problems for indefinite temporal and spatial constraint databases are identified in this way.

* This research has been partially supported by European project CHOROCHRONOS (funded under Framework IV) and by a grant from the Greek Secretariat for Research and Technology. Spiros Skiadopoulos has also been supported by a postgraduate fellowship from NATO.

M. H. Böhlen, C. S. Jensen and M. O. Scholl (Eds.): STDBM'99, LNCS 1678, pp. 204–223, 1999.

A Introduction

Linear constraints have recently been used by constraint database researchers to represent temporal, spatial and spatiotemporal information [22,13,14,37,24,31,4,12,33]. The information represented by linear constraints can be of type *absolute* (e.g., the explosion occured at 3:30 a.m. yesterday), *relative* (e.g., point A is north of point B), *indefinite* or *indeterminate* (e.g., the course lasted between 1 and 2 hours), and combination of these types (e.g., point A is north or west of point B).

Query evaluation in the presence of indefinite information is a hard problem even in the case of relational databases [47,1,11]. This tells us that the situation will probably be much worse for the case of indefinite information in spatiotemporal databases. This remark can be contrasted with reality: indefinite information is crucial for some applications of spatiotemporal databases e.g., Geographic Information Systems [34] or image and video database systems [42]. It is therefore important to determine whether there exist interesting cases where query evaluation in indefinite spatiotemporal databases can be done in PTIME.

In this paper we consider the scheme of indefinite constraint databases proposed by Koubarakis [25,26,31]. This scheme can be used to represent indefinite information arising in temporal, spatial and truly spatiotemporal applications (e.g., moving objects [40,16]). The main technical problem that we address is the discovery of tractable classes of databases and queries in this scheme. Our results are original and can be summarized as follows:

1. We start with the assumption that we have a class of constraints $\mathcal{C}$ with satisfiability and variable elimination problems that can be solved in PTIME. Under this assumption, we demonstrate several general classes of databases and queries for which query evaluation can be done with PTIME data complexity.
2. We then try to find useful instances of $\mathcal{C}$ from the area of temporal and spatial constraints. Classes with tractable satisfiability problems can be easily found in the literature. The largest class that we consider is the class of Horn disjunctive linear constraints over the rationals[1] [21,29]. A *Horn disjunctive linear constraint* is a disjunction of weak linear inequalities (e.g., $x_1 + x_2 + 3x_3 \leq 4$) and linear disequations (e.g., $x_1 - 4x_4 + 5x_3 \neq 6$) such that the number of inequalities does not exceed one.
   Because variable elimination for Horn disjunctive linear constraints cannot be done in PTIME [29], we try to discover subclasses with tractable variable elimination problems. The class of UTVPI$^{\neq}$ constraints is the largest class that we show to have this property. A *UTVPI constraint* is a linear constraint $\pm x_1 \leq c$ or $\pm x_1 \pm x_2 \leq c$ where x_1, x_2 are variables and c is a rational number. The class of UTVPI$^{\neq}$ *constraints* also includes disequations of the form $\pm x_1 \neq c$ or $\pm x_1 \pm x_2 \neq c$.

[1] All the results of this paper still hold if our constraints are taken over the set of real numbers.

Our results on UTVPI$^{\neq}$ constraints are original and extend the results of [28,30]. In the full paper we also provide efficient algorithms for consistency checking and global consistency enforcement for UTVPI and UTVPI$^{\neq}$ constraints.

3. Finally, we restate the results of Item 1 with $\mathcal{C}$ ranging over the newly discovered tractable classes. Tractable query answering problems for indefinite temporal and spatial constraint databases are identified in this way. Two of them are significant extensions of tractable problems identified previously by [5,43] while all others are new.

Our results will be of interest to spatiotemporal database researchers, constraint database researchers but also people working on traditional constraint satisfaction problems. The methodology of this paper has also been applied to the problem of querying temporal constraint networks in [32].

The paper is organized as follows. The next section reviews the scheme of indefinite constraint databases and a method for query evaluation. In Section C we identify tractable cases of query evaluation. Section D presents subclasses of linear constraints with a satisfiability problem and a variable elimination problem that can be solved in PTIME. Section E uses the results of Sections C and D to identify tractable query answering problems in temporal and spatial domains. This section also discusses some questions left open by our analysis. All proofs are omitted and can be found in the full version of the paper.

B The Scheme of Indefinite Constraint Databases

In this section we present the scheme of indefinite $\mathcal{L}$-constraint databases [31] where $\mathcal{L}$ is a many-sorted first order constraint language and $\mathbf{M}_{\mathcal{L}}$ is its intended $\mathcal{L}$-*structure* (i.e., the structure that gives the intended interpretation of the symbols of $\mathcal{L}$). In this paper we only use the following first-order constraint languages: the language of linear inequalities $\mathcal{L}_{LIN}$ and the language of polynomial inequalities $\mathcal{L}_{POLY}$. A *polynomial inequality* is an expression of the form $\Sigma_{i=1}^{n} a_i \Pi_{j=1}^{m} x_j^{k_j} \sim 0$ where the a_i's are integer coefficients, the x_j's are variables ranging over the rational numbers, the k_j's are natural numbers and $\sim$ is $<$ or $=$. A *linear inequality* is an expression of the form $\Sigma_{i=1}^{n} a_i x_i \sim c$ where the a_i's are integer coefficients, the x_i's are variables ranging over the rational numbers, $\sim$ is $<$ or $=$ and c is an integer. The language $\mathcal{L}_{POLY}$ (and its sublanguage $\mathcal{L}_{LIN}$) will be interpreted over the set of rational numbers $\mathcal{Q}$ with the obvious denotations for their symbols.

In a constraint database model with constraint language $\mathcal{L}$, we usually distinguish a class of formulas of $\mathcal{L}$ and call it the class of $\mathcal{L}$-*constraints.* In the examples of this section linear inequalities and polynomial inequalities form the classes of $\mathcal{L}_{LIN}$-constraints and $\mathcal{L}_{POLY}$-constraints respectively. Section D considers more specific classes of $\mathcal{L}_{LIN}$-constraints. In this paper we only consider $\mathcal{L}$-constraint classes that *admit variable elimination* and are *weakly closed under negation* (i.e., the negation of every $\mathcal{L}$-constraint is equivalent to a disjunction of

$\mathcal{L}$-constraints). Many interesting constraint classes have this property (e.g., the class of polynomial or linear inequalities).

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.[2] A *relation scheme* R is a finite subset of $\mathcal{U}$ (R will be called the *name* of the scheme).

For every $s \in sorts(\mathcal{L})$, we 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.

U-variables are like the variables in the constraint database scheme of [23]. E-variables are exactly like the marked nulls of [19]. 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 1. *Let R be a relation scheme. An* indefinite $\mathcal{L}$-constraint tuple *t over scheme R is a mapping from $R \cup \{CON\}$ to $UVAR_\mathcal{L} \cup WFF(\mathcal{L})$[3] 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 2. *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.*

Definition 3. *An* $\mathcal{L}$-constraint rule *is an expression of the form*

$$R_0(t_{01}, \ldots, t_{0k_0}) \leftarrow R_1(t_{11}, \ldots, t_{1k_1}), \quad \ldots, R_n(t_{n1}, \ldots, t_{nk_n}), c_1, \ldots, c_m$$

where $R_0, R_1, \ldots, R_n$ are relation scheme names, $c_1, \ldots, c_m$ are $\mathcal{L}$-constraints and each t_{ij} is a term of $\mathcal{L}$ (i.e., a variable, a constant or a function term). An $\mathcal{L}$-constraint rule of the above form is recursive *iff R_0 is one of $R_1, \ldots, R_n$. For the purposes of this paper $\mathcal{L}$-constraint rules are assumed to be* non-recursive.[4]

Definition 4. *A* database scheme *$\widetilde{R}$ is a sequence $(R_1, \ldots, R_n)$ where each R_i is a relation scheme.*

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

[3] $WFF(\mathcal{L})$ denotes the set of quantifier-free well-formed formulas of $\mathcal{L}$.

[4] Let us notice that the indefinite constraint database scheme of [31] does not contain $\mathcal{L}$-constraint rules. The introduction of rules in the present paper is done simply for convenience so that new relations can be defined from existing ones. Since rules are assumed to be non-recursive, no expressive power with respect to querying is added to the scheme.

Definition 5. *An* indefinite $\mathcal{L}$-constraint database *DB over scheme* $\widetilde{R} = (R_1, \ldots, R_n)$ *is a triple* $(\widetilde{r}, RS, CS)$ *where* $\widetilde{r} = (r_1, \ldots, r_n)$ *is a sequence of indefinite* $\mathcal{L}$*-constraint relations (the scheme of* r_i *is* R_i*), RS is a set of* $\mathcal{L}$*-constraint rules and CS is a conjunction of* $\mathcal{L}$*-constraints over* $EVAR_{\mathcal{L}}$*.* $\widetilde{r}$ *is called the* relational part *of DB, RS is called the* rule set *and CS is called the* constraint store*.*

The above definitions extend the constraint database scheme of Kanellakis, Kuper and Revesz [22] by introducing an additional kind of variables called *e-variables*. E-variables have the semantics of marked nulls of [19]. As in [11], the possible values of the e-variables can be constrained by a *constraint store*. The semantics of our scheme are the usual *closed-world* semantics discussed in [11]. The interested reader can consult [25,31] for details.

The following example from [31] shows the power of the scheme for the representation of *temporal* information.

Example 1. Let us consider a planning database used by a medical laboratory for keeping track of patient appointments for the year 1996. The set of rationals $\mathcal{Q}$ 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 is an indefinite $\mathcal{L}_{LIN}$-constraint database for this application domain.[5] The implicit constraint language is the union of two simpler languages: the first order language of linear constraints (for temporal data), and a first order language with equality and an infinite number of constants (for administrative data).

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*

$$CONSTRAINT\_STORE:$$
$$\omega_1 \geq 0,\ \omega_2 \geq 0,\ \omega_3 \geq 0,\ \omega_4 \geq 0,\ \omega_5 \geq 0,\ \omega_6 \geq 0,$$
$$\omega_2 = \omega_1 + 1,\ \omega_4 = \omega_3 + 1,\ \omega_6 = \omega_5 + 2,\ \omega_2 \leq 91,\ \omega_3 \geq 91,\ \omega_4 \leq 182,$$
$$\omega_3 - \omega_2 \geq 60,\ \omega_5 - \omega_4 \geq 20,\ \omega_6 \leq 213$$

The above database represents the following information. There are three scheduled appointments for patient Smith. This is represented by three tuples in relation APPOINTMENT. Chemotherapy appointments must be scheduled

[5] We do not follow strictly the definition of an $\mathcal{L}$-constraint tuple for reasons of economy of expression and clarity. These examples can easily be made to conform with Definition 1 by introducing u-variables and adding equality constraints in the local condition of each tuple.

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 $\omega_6 = \omega_5 + 2$. There is also more information about individual appointments. E.g., the first chemotherapy appointment for Smith should take place in the first three months of 1996 (i.e., days 0-91) etc.

The following example shows that *truly spatiotemporal information* can also be captured in our scheme. The example has been motivated by the work of [40,41].

Example 2. Let us consider an object moving on a straight line in $\mathcal{Q}^2$ with motion vector $d = (d_1, d_2)$ and speed v. Let us also assume that its initial position at time t_0 is $(x(t_0), y(t_0))$. The position $(x(t), y(t))$ of the object at future time t can be computed using the following equations:

$$x(t) = x(t_0) + v(t - t_0)d_1 \text{ and } y(t) = y(t_0) + v(t - t_0)d_2$$

Let us now consider a concrete example and assume that we have an indefinite $\mathcal{L}_{POLY}$-constraint database containing information about the moving object $Car1$:

OBJECT

ID	CON
$Car1$	*true*

INIT_POS

ID	TIME	XCORD	YCORD	CON
$Car1$	0	1	1	*true*

VECTOR

ID	XVAL	YVAL	CON
$Car1$	2	1	*true*

SPEED

ID	VALUE	CON
$Car1$	ω	*true*

$$CONSTRAINT\_STORE: \ 40 \leq \omega \leq 50$$

$$FUTURE\_POS(o,t,x_0+v(t-t_0)d_1,y_0+v(t-t_0)d_2) \leftarrow OBJECT(o), \\ INIT\_POS(o, t_0, x_0, y_0), \\ VECTOR(o, d_1, d_2), \\ SPEED(o, v).$$

The only relation with truly indefinite information is $SPEED$. The speed of $Car1$ is not known precisely: all we know is that it is between 40 and 50 miles per time unit. The future position of any object is computed using the $\mathcal{L}_{POLY}$-constraint rule which defines the relation $FUTURE\_POS$. Later in this section, we will pose queries to the above database to compute the position of object $Car1$ at times other than 0.

B.1 Query Evaluation

In [25,27,31] Koubarakis defined *modal relational calculus with $\mathcal{L}$-constraints* as a declarative query language for indefinite $\mathcal{L}$-constraint databases. This modal calculus is essentially relational calculus with $\mathcal{L}$-constraints plus the modal operator $\Diamond$ or $\Box$ appended in front of relational calculus expressions. Therefore modal queries can be distinguished between *possibility* queries and *necessity* (or *certainty*) queries. A model-theoretic semantics for our query language has been given in [25,27,31]. The query evaluation procedure sketched below gives an equivalent proof-theoretic semantics.

Example 3. Let us consider the database of Example 1 and the query "Find all appointments for patients that possibly start at the 92th day of 1996". This query can be expressed as follows:[6]

$$\{\ APP\_S92(PATIENT, TREATMENT), x, y/\mathcal{D} : \\ \Diamond(\exists t_1, t_2/\mathcal{Q})(APPOINTMENT(x, y, t_1, t_2)\ \wedge\ t_1 = 92)\ \}$$

The answer to this query is the following:

APP_S92

PATIENT	TREATMENT	CON
Smith	Chemotherapy2	*true*
Smith	Radiation	*true*

$CONSTRAINT\_STORE : true$

Example 4. Let us consider the database of Example 1 and the query "Find all appointments for patients that should necessarily take place in the first six months of 1996". This query can be expressed as follows:

$$\{\ APP(PATIENT, TREATMENT), x, y/\mathcal{D} : \\ \Box(\exists t_1, t_2/\mathcal{Q})(APPOINTMENT(x, y, t_1, t_2)\ \wedge\ t_1 \geq 0\ \wedge\ t_2 \leq 182)\ \}$$

The answer to this query is the following:

APP

PATIENT	TREATMENT	CON
Smith	Chemotherapy1	*true*
Smith	Chemotherapy2	*true*

$CONSTRAINT\_STORE : true$

Example 5. Consider the database of Example 2 and the query "What are the possible future positions of object $Car1$ at time 5". This query can be expressed as follows:

$$\{\ POS\_AT\_5(XCORD, YCORD), x, y/\mathcal{Q} :\ \Diamond FUTURE\_POS(Car1, 5, x, y)\ \}$$

The above query has the following answer:

[6] $\mathcal{D}$ is the *data sort*.

POS_AT_5

XCORD	YCORD	CON
x	y	$401 \leq x \leq 501,\ y = \frac{1}{2}x + \frac{1}{2}$

$$CONSTRAINT\_STORE:\ true$$

Because the database does not contain definite information about the speed of $Car1$, a line segment in $\mathcal{Q}^2$ is returned as the answer to the query.

In our scheme we can express many of the queries on moving points discussed in [40,41,16]. We can not express queries involving special functions (e.g., aggregates) since we have not introduced them in our framework. Queries on moving regions can also be expressed. However as shown in [4,12] spatial regions can be represented more gracefully in a constraint database model based on nested relations with one level of nesting. Extending our scheme in this direction is a topic of current research.

As in the definite information case [22], query evaluation over indefinite $\mathcal{L}$-constraint databases can be viewed as quantifier elimination in the theory $Th(\mathbf{M}_{\mathcal{L}})$ [25,27,31]. Quantifier elimination is always possible in our framework since $Th(\mathbf{M}_{\mathcal{L}})$ admits quantifier elimination. If q is a possibility query with free variables $\overline{x}$ over database $DB = (\widetilde{r}, RS, CS(\overline{\omega}))$ then query evaluation is equivalent to quantifier elimination from a formula of the form $(\exists \overline{\omega}/\overline{s'})(CS(\overline{\omega}) \ \wedge \psi(\overline{x}, \overline{\omega}))$ where $\overline{s'}$ is a vector of sorts and ψ is a formula of $Th(\mathbf{M}_{\mathcal{L}})$ which depends on q and DB. Similarly, if q is a certainty query with free variables $\overline{x}$ then query evaluation is equivalent to quantifier elimination from a formula of the form $(\forall \overline{\omega}/\overline{s'})(CS(\overline{\omega}) \Rightarrow \psi(\overline{x}, \overline{\omega}))$.

Example 6. Let us consider the query of Example 3 over the database of Example 1. This query can be evaluated by eliminating quantifiers from the following formula:

$$\begin{aligned}
&(\exists \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6/\mathcal{Q})\\
&\quad((\omega_1 \geq 0 \ \wedge\ \omega_2 \geq 0 \ \wedge\ \omega_3 \geq 0 \ \wedge\ \omega_4 \geq 0 \ \wedge\ \omega_5 \geq 0 \ \wedge\ \omega_6 \geq 0 \wedge\\
&\quad\omega_2 = \omega_1 + 1 \ \wedge\ \omega_4 = \omega_3 + 1 \ \wedge\ \omega_6 = \omega_5 + 2 \ \wedge\ \omega_2 \leq 91 \ \wedge\ \omega_3 \geq 91 \wedge\\
&\quad\omega_4 \leq 182 \ \wedge\ \omega_3 - \omega_2 \geq 60 \ \wedge\ \omega_5 - \omega_4 \geq 20 \ \wedge\ \omega_6 \leq 213) \wedge\\
&\quad(\exists t_1, t_2/\mathcal{Q})((x = Smith \ \wedge\ y = Chemotherapy1 \ \wedge\ t_1 = \omega_1 \ \wedge\ t_2 = \omega_2) \vee\\
&\qquad\qquad(x = Smith \ \wedge\ y = Chemotherapy2 \ \wedge\ t_1 = \omega_3 \ \wedge\ t_2 = \omega_4) \vee\\
&\qquad\qquad(x = Smith \wedge y = Radiation \wedge t_1 = \omega_5 \wedge t_2 = \omega_6)) \wedge t_1 = 92).
\end{aligned}$$

The result of the elimination is

$$(x = Smith \ \wedge\ y = Chemotherapy2) \ \vee\ (x = Smith \ \wedge\ y = Radiation)$$

and it is shown as a relation in Example 3.

Example 7. Let us now consider the query of Example 5 over the database of Example 2. First of all notice that the instance of relation $FUTURE\_POS$ derived from the instances of relations $OBJECT, INIT\_POS, VECTOR$ and $SPEED$ after applying the given rule contains only the tuple

$$(Car1,\ t,\ 1 + 2\omega t,\ 1 + \omega t).$$

Therefore our query can be evaluated by eliminating quantifiers from the following formula:

$$(\exists \omega / \mathcal{Q})(40 \leq \omega \leq 50 \ \wedge \ x = 1 + 10\omega \ \wedge \ y = 1 + 5\omega)$$

The result of the elimination is

$$401 \leq x \leq 501 \ \wedge \ y = \frac{1}{2}x + \frac{1}{2}$$

and it is shown as a relation in Example 5.

C Tractable Query Evaluation in Indefinite Constraint Databases

The complexity of database query evaluation is usually measured using the notion of data complexity introduced in [46,6].[7] When we use data complexity, we measure the complexity of query evaluation as a function of the database size only; the query and the schema are considered *fixed*. In our case we also assume that the size of any integer constant in the database is *logarithmic* in the size of the database.[8]

Assuming the data complexity measure, is it easy or hard to answer a query over an indefinite constraint database? Not surprisingly, the answer depends primarily on the type of constraints involved and secondarily on the type of the database and the query.

In previous research, Koubarakis [26,31] has provided upper bounds for modal calculus queries over indefinite point constraint databases where point constraints have been restricted to inequalities of the form $t_1 - t_2 \leq c$ interpreted over the rationals or the integers. [45] has done very good work on lower bounds and PTIME cases for indefinite order databases.

In the Artificial Intelligence literature, [43] has exhibited a class of indefinite interval constraint databases and a class of conjunctive possibility queries with a query answering problem that can be done with PTIME data complexity. [5] has a very similar result for indefinite point constraint databases where the point constraints considered are as in [26,31].

In this paper we start with the assumption that we have an arbitrary class of constraints $\mathcal{C}$ with an "easy" satisfiability problem and variable elimination problem. Under this assumption, we show that there are several classes of databases and modal queries involving $\mathcal{C}$-constraints, for which query evaluation can be done with PTIME data complexity.

We will reach tractable cases of query evaluation by restricting the classes of constraints, databases and queries we allow in our framework. We introduce the concepts of query type and database type to make these distinctions.

[7] There is also *expression complexity* and *combined complexity* but we will not deal with them in this paper.

[8] When a database is viewed as a formula in some constraint language, 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 [22] and [38].

C.1 Query Types

A *query type* is a tuple of the following form:

$$Q(Open/Closed, Modality, PositiveExistential/SingleRelation, Constraints)$$

The first argument of a query type distinguishes between *closed* and *open* queries. A query is *open* when it has free variables and it is *closed* (or *yes/no*) when it has no free variables. The argument *Modality* can be $\Diamond$ or $\Box$ representing possibility or necessity queries respectively.

The third argument can be *PositiveExistential* or *SingleRelation*. We use *PositiveExistential* to denote that the relational calculus expression in the query is a positive existential one i.e., it is of the form $(\exists \overline{x}/\overline{s})\phi(\overline{x})$ where ϕ involves only the logical symbols $\wedge$ and $\vee$.

We use *SingleRelation* to denote that the query is of the form $\overline{u}/\overline{s}_1 : \ OP\ (\exists \overline{t}/\overline{s}_2)p(\overline{u}, \overline{t})$ where $\overline{u}$ and $\overline{t}$ are vectors of variables, $\overline{s}_1$, $\overline{s}_2$ are vectors of sorts, p is a relation name and OP is a modal operator.

The fourth argument *Constraints* denotes the class of constraints that is used as query conditions. For example consider a database DB with two binary relations r and p. The query

$$: \Box(\exists x, y, t, u/\mathcal{Q})((r(x,t) \ \vee \ p(y,u)) \ \wedge \ 2t + 3u \leq 4 \ \wedge \ x = u + 1)$$

belongs to the query class

$$Q(Closed, \Box, PositiveExistential, Linear).$$

C.2 Database Types

We can characterise an indefinite constraint database according to the arity of its predicates, the class of constraints of its constraint store, and the class of local conditions in the database:

$$DB(Arity, LocalCondition, ConstraintStore)$$

The attribute *Arity* can have values *N-ary* and *Monadic* and gives the arity of the predicates in the database. The attribute *LocalCondition* gives the constraint class of the local condition of every tuple in the database. The attribute *ConstraintStore* gives the constraint class of the database constraint store.

For instance, a database having:

- N-ary predicates with local condition a conjunction of linear constraints.
- A constraint store which is a set of linear constraints.

can be represented by type $DB(N\text{-}ary, Linear, Linear)$.

In one of our results we consider the type of a database which consists of a *single* N-ary relation, the constraints in the constraint store belong to some class $\mathcal{C}$, and all local conditions of tuples consist of a *single* constraint from class $\mathcal{C}$. This database type is represented by

$$SingleRelDB(N\text{-}ary, Single\text{-}\mathcal{C}, \mathcal{C}).$$

C.3 Results

The following definitions are useful in the rest of this section.

Definition 6. *If $\mathcal{C}$ is a class of constraints then $\vee\overline{\mathcal{C}}$ is a new class of constraints defined as follows. A constraint c is in $\vee\overline{\mathcal{C}}$ iff c is a disjunction of negations of $\mathcal{C}$-constraints.*

Definition 7. *Let $\mathcal{C}$ be a class of constraints. The problem of deciding whether a given set of constraints from $\mathcal{C}$ is satisfiable will be denoted by $SAT(\mathcal{C})$. The problem of eliminating an arbitrary number of variables from a given set of constraints from class $\mathcal{C}$ will be denoted by $VAR\text{-}ELIM(\mathcal{C})$. If the number of variables is fixed then the problem will be denoted by $FVAR\text{-}ELIM(\mathcal{C})$.*

Let us now present our results assuming the data complexity measure. We first consider closed queries. In the rest of this section sorts in query variables are omitted for brevity. If $\mathcal{C}$ is a class of constraints then the members of $\mathcal{C}$ will be referred to as $\mathcal{C}$-constraints.

Lemma 1. *Let $\mathcal{C}$ be a class of constraints. Evaluating a query of the class*

$$Q(Closed, \diamond, PositiveExistential, \mathcal{C})$$

over an indefinite constraint database $\tilde{r}$ of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ is equivalent to deciding the consistency of a set of m formulas of the form

$$CS(\overline{\omega}) \wedge \theta_i(\overline{\omega}),\ i = 1, \ldots, m$$

where CS and θ_i are conjunctions of $\mathcal{C}$-constraints, and m is a polynomial in the size of $\tilde{r}$.

Theorem 1. *Let $\mathcal{C}$ be a class of constraints such that $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{C})$ can be solved in PTIME. Let $\tilde{r}$ be an indefinite constraint database of the class*

$$DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$$

and f be a query of the class

$$Q(Closed, \diamond, PositiveExistential, \mathcal{C})$$

The problem of deciding whether $f(\tilde{r}) = yes$ can be solved with PTIME data complexity.

Lemma 2. *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E} \subseteq \mathcal{C}$ and $\vee\overline{\mathcal{E}} \subseteq \mathcal{C}$. Evaluating a query of the class $Q(Closed, \Box, PositiveExistential, \mathcal{E})$ over an indefinite constraint database $\tilde{r}$ of the class $DB(N\text{-}ary, \mathcal{E}, \mathcal{C})$ is equivalent to deciding the consistency of a formula of the form*

$$CS(\overline{\omega}) \wedge \theta_1(\overline{\omega}) \wedge \ldots \wedge \theta_m(\overline{\omega})$$

where CS is a conjunction of $\mathcal{C}$-constraints, θ_i, $1 \leq i \leq m$ are $\mathcal{C}$-constraints, and m is a polynomial in the size of $\tilde{r}$.

Theorem 2. *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E} \subseteq \mathcal{C}$, $\vee\overline{\mathcal{E}} \subseteq \mathcal{C}$ and $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{E})$ can be solved in PTIME. Let $\tilde{r}$ be an indefinite constraint database of the class*

$$DB(N\text{-}ary, \mathcal{E}, \mathcal{C})$$

and f be a query of the class

$$Q(Closed, \Box, PositiveExistential, \mathcal{E}).$$

The problem of deciding whether $f(\tilde{r}) = yes$ can be solved with PTIME data complexity.

We now turn our attention to tractable query evaluation for open queries.

Lemma 3. *Let $\mathcal{C}$ be a class of constraints. Evaluating a query of the class*

$$Q(Open, \Diamond, PositiveExistential, \mathcal{C})$$

over an indefinite constraint database $\tilde{r}$ of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ is equivalent to eliminating quantifiers from a set of m formulas of the form

$$(\exists\overline{\omega})(CS(\overline{\omega}) \ \wedge \ \theta_i(\overline{u}, \overline{\omega})), \quad i = 1, \ldots, m$$

where CS and θ_i are conjunctions of $\mathcal{C}$-constraints, $\overline{u}$ is the vector of free variables of the query, and m is a polynomial in the size of $\tilde{r}$.

Theorem 3. *Let $\mathcal{C}$ be a class of constraints such that $VAR\text{-}ELIM(\mathcal{C})$ can be done in PTIME. Let $\tilde{r}$ be an indefinite constraint database of the class*

$$DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$$

and f be a query of the class

$$Q(Open, \Diamond, PositiveExistential, \mathcal{C}).$$

The problem of evaluating $f(\tilde{r})$ can be solved with PTIME data complexity.

Lemma 4. *Let $\mathcal{C}$ be a class of constraints which is closed under negation. Evaluating a query of the class*

$$Q(Open, \Box, SingleRelation, None)$$

over an indefinite constraint database $\tilde{r}$ of the class $SingleRelDB(N\text{-}ary, Single\text{-}\mathcal{C}, \mathcal{C})$ is equivalent to eliminating quantifiers from a formula of the form

$$CS(\overline{\omega}) \wedge \theta_1(\overline{u}, \overline{\omega}) \wedge \ldots \wedge \theta_m(\overline{u}, \overline{\omega})$$

where CS is a conjunction of $\mathcal{C}$-constraints, θ_i, $1 \leq i \leq m$ are $\mathcal{C}$-constraints, $\overline{u}$ is the vector of free variables of the query and m is the number of tuples of the single relation referred in the query.

Theorem 4. *Let $\mathcal{C}$ be a class of constraints such that $\mathcal{C}$ is closed under negation and $VAR\text{-}ELIM(\mathcal{C})$ can be done in PTIME. Let $\tilde{r}$ be an indefinite constraint database of the class*

$$SingleRelDB(N\text{-}ary, Single\text{-}\mathcal{C}, \mathcal{C})$$

and f be a query of the class

$$Q(Open, \Box, SingleRelation, None).$$

The problem of evaluating $f(\tilde{r})$ can be solved with PTIME data complexity.

This section presented some results on tractable query answering problems for the scheme of indefinite $\mathcal{L}$-constraint databases. Let us now see how these results are specialized for the model of indefinite $\mathcal{L}_{LIN}$-constraint databases. The first thing that we have to do is find subclasses of linear constraints that have the nice properties assumed by the theorems of this section.

D The Class of UTVPI$^{\neq}$ Constraints

Linear constraints and their subclasses have been studied in many areas of computer science, and more recently in temporal and spatial reasoning. The following is an interesting class of linear constraints with good computational properties.

Definition 8 ([29,21]). *A* Horn disjunctive linear constraint *is a disjunction $d_1 \vee ... \vee d_n$ where each d_i, $i = 1, ..., n$ is a weak linear inequality or a linear disequation and the number of inequalities among d_1, ..., d_n does not exceed one.*

[21,29] have shown that Horn disjunctive linear constraints subsume many interesting classes of temporal and spatial constraints.

Example 8. The following is a set of Horn disjunctive linear constraints:

$$3x_1 + x_5 - 3x_4 \leq 10,\ x_1 + x_3 + x_5 \neq 7,\ 4x_1 + x_3 \neq 3 \vee 5x_2 + 3x_5 + x_4 \neq 5$$

$$3x_1 + x_5 - 3x_4 \leq 10 \vee 2x_1 + 3x_2 - 4x_3 \neq 4 \vee x_1 + x_3 + x_5 \neq 7$$

Theorem 5 ([29,21]). *The satisfiability of a set of Horn disjunctive linear constraints can be decided in PTIME.*

Unfortunately we do not have a nice theorem like the above concerning variable elimination for Horn disjunctive linear constraints. In fact, variable elimination cannot be done in PTIME even for sets of linear inequalities. If we have a set C of linear inequalities, it might not be possible to describe the result of a variable elimination operation on C by a set of linear inequalities with size less than exponential in the number of eliminated variables [50]. The following is a weaker result which considers Horn disjunctive linear constraints and a fixed number of variables.

Theorem 6 ([29]). *We can eliminate a fixed number of variables from a set of Horn disjunctive linear constraints in PTIME.*

Luckily there are *subclasses* of linear constraints where variable elimination can be done in PTIME. Let us consider them in turn.

Definition 9 ([28,30]). *A* DIFF$^{\neq}$ constraint *is a linear constraint of the form* $t_1 \sim c$ *or* $t_1 - t_2 \sim c$ *where* t_1, t_2 *are variables,* c *is a constant and* $\sim$ *is* $\leq$ *or* $\neq$.

The above constraint class extends the class of *difference constraints* or *DIFF constraints* studied in temporal reasoning [8] by adding disequations of the form $x_1 - x_2 \neq c$ or $x_1 \neq c$.

Theorem 7 ([28,30]). *Let* C *be a set of* DIFF$^{\neq}$ *constraints. The consistency of* C *can be decided in* $O(n^3)$ *time where* n *is the number of variables in* C. *We can eliminate any number of variables from* C *in* $O(dn^4)$ *time where* d *is the number of disequations and* n *is the number of variables in* C.

Can we extend the class of DIFF (resp. DIFF$^{\neq}$) constraints without sacrificing its nice computational properties? The first natural extension that comes to mind is the class of UTVPI (resp. UTVPI$^{\neq}$) constraints.[9]

Definition 10. *A* UTVPI constraint *is a linear constraint of the form* $\pm x_1 \leq c$ *or* $\pm x_1 \pm x_2 \leq c$ *where* x_1, x_2 *are variables and* c *is a constant. If we also allow disequations, we have the class of* UTVPI$^{\neq}$ constraints.

Example 9. Below we give some examples of UTVPI$^{\neq}$ constraints:

$$-x_1 \leq 12,\ x_1 + x_2 \leq 2,\ x_3 - x_2 \leq 0.5,\ x_3 + x_2 \neq 6$$

UTVPI$^{\neq}$ constraints are an interesting subclass of linear constraints for spatial databases over $\mathcal{Q}^2$. They are more expressive than dense order constraints [23,9,10], and thus they allow more precise approximations of arbitrary regions of $\mathcal{Q}^2$ traditionally approximated by bounding boxes. The disequations present in this class allowe a limited form of negative information (e.g., they can be used to *remove* points and straight lines from polygons).[10]

The following theorem is the main result of this section. It shows that UTVPI$^{\neq}$ constraints can give us more expressive power than DIFF$^{\neq}$ constraints without sacrificing none of their nice computational properties.

Theorem 8. *Let* C *be a set of* UTVPI$^{\neq}$ *constraints. The consistency of* C *can be decided in* $O(n^3)$ *time where* n *is the number of variables in* C. *We can eliminate any number of variables from* C *in* $O(dn^4)$ *time where* d *is the number of disequations and* n *is the number of variables in* C.

[9] UTVPI is an acronym for linear inequalities with *U*nit coefficients and at most *T*wo *V*ariables *P*er *I*nequality.

[10] UTVPI constraints *over the integers* have also been studied [20,17]. [17] presents a constraint logic programming system with UTVPI constraints and several applications that would otherwise be solved using finite domain constraints.

It is natural now to ask the following question. Can we extend the above theorem to the class of *two-variables-per-inequality* or *TVPI* constraints (i.e., linear constraints of the form $ax + by \leq c$ where x, y are variables and a, b are rational constants)? TVPI constraints have previously been considered in several papers including [39,18,9,10]. It is currently an open problem whether variable elimination for TVPI constraints can be done in PTIME. [9,10] has considered *monotone* TVPI constraints (i.e., TVPI constraints of the form $x \leq by + c$ with $b \geq 0$) and showed that under a condition involving the graph representation of the constraints, variable elimination can be done in strongly polynomial time (i.e., where the polynomials do not depend on the coefficient sizes).

E Tractable Query Answering with Indefinite Information in Temporal and Spatial Domains

Let us recall that a query type is a tuple of the form

$$Q(Open/Closed, Modality, PositiveExistential/SingleRelation, Constraints)$$

and a database type is a tuple of the form

$$DB(Arity, LocalCondition, ConstraintStore).$$

The arguments *Constraints* or *LocalCondition* vary over constraint classes. For the purposes of this section, these classes can be *HornDisjLinear, Linear, LinearEquality,* UTVPI$^{\neq}$*, DIFF* (with obvious meaning). We will also encounter the classes *Single*-UTVPI$^{\neq}$ and *None*. In the former case the local condition in every tuple of the database consists of a single UTVPI$^{\neq}$ constraint. In the latter case we have no constraints (equivalently, all constraints are *true*).

We will also have the following new classes of temporal and spatial constraints:

- **IA**. This is the Interval Algebra of Allen [2].
- **SIA**. This is the subalgebra of **IA** which includes only interval relations that can be translated into conjunctions of order constraints $x \leq y$ or $x \neq y$ on the endpoints of intervals [44].
- **ORDHorn**. This is the subalgebra of **IA** which includes only interval relations that can be translated into conjunctions of ORD-Horn constraints on the endpoints of intervals [35]. An *ORD-Horn constraint* is a disjunction of weak inequalities of the form $x \leq y$ and disequations of the form $x \neq y$ with the additional constraint that the number of inequalities should not exceed one [35].
- **2d-IA** and **2d-OrdHorn**.
  2d-IA is a generalization of **IA** in two dimensions and it is based on the concept of *rectangle* in $\mathcal{Q}^2$ [15,36,3]. Every rectangle r can be defined by a 4-tuple $(L_x^r, L_y^r, U_x^r, U_y^r)$ that gives the coordinates of the lower left and upper right corner of r. There are 13^2 basic relations in **2d-IA** describing all possible configurations of 2 rectangles in $\mathcal{Q}^2$ [36].

2d-OrdHorn is the subalgebra of **2d-IA** which includes only these relations R with the property $r_1\ R\ r_2 \equiv \phi(r_1, r_2)\ \wedge\ \psi(r_1, r_2)$ where ϕ is a conjunction of ORD-Horn constraints on variables L_x^r and U_x^r, and ψ is a conjunction of ORD-Horn constraints on variables L_y^r and U_y^r.

– **PA** and **CPA**. **PA** is the Point Algebra of [48]. **CPA** is the subalgebra of **PA** which does not include the relation $\neq$ [49].

The following theorem uses the results of Sections C and D to identify tractable query answering problems in indefinite linear constraint databases.

Theorem 9. *Evaluation of*

1. $Q(Closed, \diamond, PositiveExistential, HornDisjLinear)$ *queries over* $DB(N\text{-}ary, HornDisjLinear, HornDisjLinear)$ *databases,*
2. $Q(Closed, \Box, PositiveExistential, LinearEquality)$ *queries over* $DB(N\text{-}ary, LinearEquality, HornDisjLinear)$ *databases,*
3. $Q(Open, \diamond, PositiveExistential, \text{UTVPI}^{\neq})$ *queries over* $DB(N\text{-}ary, \text{UTVPI}^{\neq}, \text{UTVPI}^{\neq})$ *databases and*
4. $Q(Open, \Box, SingleRelation, None)$ *queries over* $SingleRelDB(N\text{-}ary, Single\text{-}\text{UTVPI}^{\neq}, \text{UTVPI}^{\neq})$ *databases*

can be performed in PTIME (using the data complexity measure).

The first part of Theorem 9 is a significant extension of the PTIME result of [5] on possibility queries over conjunctions of DIFF constraints.

Theorem 9 does not mention constraints on higher-order objects (e.g., intervals, rectangles) explicitly so one might think that it useful only for one-dimensional databases. Luckily this is not true. For example, results for interval constraint databases can be deduced immediately by taking into account the subsumption relations between classes of interval and point constraints. For example, the first part of Theorem 9 implies that evaluating

$$Q(Closed, \diamond, PositiveExistential, \textbf{ORDHorn})$$

queries over $DB(N\text{-}ary, None, \textbf{ORDHorn})$ databases can be done with PTIME data complexity. This is a significant extension of the PTIME result of [43] on possibility queries over conjunctions of **SIA** constraints.

The first part of Theorem 9 also implies that evaluating

$$Q(Closed, \diamond, PositiveExistential, \textbf{2d-ORDHorn})$$

queries over $DB(N\text{-}ary, None, \textbf{2d-ORDHorn})$ databases can be done with PTIME data complexity. This is an interesting result for spatial databases with indefinite information over $\mathcal{Q}^2$. This result can be generalized to $\mathcal{Q}^n$ if one defines an appropriate algebra **nd-ORDHorn**.

Theorem 9 does not constrain the arity of database predicates. The following is a result by van der Meyden [45] which constrains the database to consist of monadic predicates only. In this way van der Meyden has discovered a tractable query answering problem that allows a certainty query to contain order constraints (compare with the second result of Theorem 9).

Theorem 10. *Evaluating* $Q(Closed, \Box, Conjunctive, \mathbf{CPA})$ *queries over* $DB(Monadic, None, \mathbf{CPA})$ *databases can be done in PTIME (using the data complexity measure).*

If we move to databases with binary predicates, the corresponding query answering problem becomes NP-complete [45].

Unlike [45], we have not proven any lower bound results that show where the boundary is between tractable and hard query answering problems with respect to the classes of queries and databases considered. The boundary can sometimes be identified easily with respect to the constraint class considered. For example, when we go from Horn disjunctive constraints to a constraint class with an intractable consistency checking problem (e.g., **IA** constraints) then the first tractability result of Theorem 9 is invalidated. In general more research is necessary for drawing an informative picture of tractable vs. intractable query answering problems.

We do not deal with *expression complexity* and *combined complexity* at all in this paper. Query anwering problems like the ones considered in Theorems 9 and 10 can be shown to be very hard in the combined complexity measure [45]. [45] has introduced several additional restrictions on queries and databases to arrive at query answering problems with PTIME combined complexity (but only the simple case of **PA** constraints is considered). A similar detailed study of expression/combined complexity of query answering in our framework is left to further research.

Finally let us mention the work of [7] who consider tractable temporal reasoning in a framework similar to ours. A detailed comparison of our work with [7] is not very easy at this stage because the formalism of [7] allows several kinds of deductive rules that cannot be encoded in our model.

References

1. S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 34–48, 1987. 205
2. J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983. 218
3. P. Balbiani, J.-F. Condotta, and L. F. del Cerro. Bidimensional Temporal Relations. In *Proceedings of KR'98*, June 1998. 218
4. E. Bertino, A. Belussi, and B. Catania. Manipulating Spatial Data in Constraint Databases. In M. Scholl and A. Voisard, editors, *Proc. of the Fifth Int. Symp. on Spatial Databases*, number 1262 in Lecture Notes in Computer Science, pages 115–140, Berlin, Germany, July 1997. Springer Verlag, Berlin. 205, 211
5. V. Brusoni, L. Console, and P. Terenziani. On the computational complexity of querying bounds on differences constraints. *Artificial Intelligence*, 74(2):367–379, 1995. 206, 212, 219
6. A. Chandra and D. Harel. Structure and Complexity of Relational Queries. *Journal of Computer and System Sciences*, 25:99–128, 1982. 212
7. T. Dean and M. Boddy. Reasoning About Partially Ordered Events. *Artificial Intelligence*, 36:375–399, 1988. 220

8. 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. 217
9. D. Q. Goldin. *Constraint Query Algebras.* PhD thesis, Dept. of Computer Science, Brown University, 1997. 217, 218
10. D. Q. Goldin and P. Kanellakis. Constraint Query Algebras. *Constraints*, 1(1):45–83, 1997. 217, 218
11. 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. 205, 208
12. S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 213–224, 1998. 205, 211
13. 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. 205
14. 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. 205
15. H.-W. Guesgen. Spatial reasoning based on Allen's temporal logic. Technical Report TR-89-094, ICSI, 1989. 218
16. R. H. Gueting, M. H. Bohlen, M. Erwing, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. Technical Report 238-9, Informatik, FernUniversitat, 1998. 205, 211
17. W. Harvey and P. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Proceedings of Australian Computer Science Conference (Australian Computer Science Communications)*, pages 102–111, 1997. 217
18. D. S. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal of Computing*, 23(6):1179–1192, 1994. 218
19. T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984. 207, 208
20. Joxan Jaffar, Michael J. Maher, Peter Stuckey, and Ronald Yap. Beyond Finite Domains. In A. Borning, editor, *Proceedings of PPCP'94*, volume 874 of *Lecture Notes in Computer Science*, pages 86–94. Springer Verlag, 1994. 217
21. Jonsson, P. and Bäckström, C. A Linear Programming Approach to Temporal Reasoning. In *Proceedings of AAAI-96*, 1996. 205, 216
22. 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. 205, 208, 211, 212
23. P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995. 207, 217
24. M. Koubarakis. Database Models for Infinite and Indefinite Temporal Information. *Information Systems*, 19(2):141–173, March 1994. 205
25. 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. 205, 208, 210, 211

26. 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. 205, 212
27. 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. 210, 211
28. M. Koubarakis. From Local to Global Consistency in Temporal Constraint Networks. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, volume 976 of *LNCS*, pages 53–69, Cassis, France, September 1995. 206, 217
29. M. Koubarakis. Tractable Disjunctions of Linear Constraints. In *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming (CP'96)*, Boston, MA, August 1996. 297-307. 205, 216, 217
30. M. Koubarakis. From Local to Global Consistency in Temporal Constraint Networks. *Theoretical Computer Science*, 173:89–112, February 1997. Invited submission to the special issue dedicated to the 1st International Conference on Principles and Practice of Constraint Programming (CP95), Editors: U. Montanari and F. Rossi. 206, 217
31. M. Koubarakis. The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases. *Theoretical Computer Science*, 171:25–60, January 1997. Special Issue on Uncertainty in Databases and Deductive Systems, Editor: L. V. S. Lakshmanan. 205, 206, 207, 208, 210, 211, 212
32. M. Koubarakis and S. Skiadopoulos. Querying Temporal Constraint Networks in PTIME. In *Proceedings of AAAI-99*, 1999. Forthcoming. 206
33. G. M. Kuper, S. Ramaswamy, K. Shim, and J. Su. A Constrint-Based Spatial Extension to SQL. In *Proceedings of ACM-GIS98*, pages 112–117, 1998. 205
34. R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, 1992. 205
35. Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66, January 1995. 218
36. D. Papadias, Y. Theodoridis, T. Sellis, and M. Egenhofer. Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 92–103, 1995. 218
37. J. Paredaens. Spatial Databases: the Final Frontier. In *Proceedings of ICDT-95*, pages 14–32, 1995. 205
38. 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. 212
39. Robert Shostak. Deciding Linear Inequalities by Computing Loop Residues. *Journal of the ACM*, 28(4):769–779, 1981. 218
40. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proceedings of ICDE-97*, 1997. 205, 209, 211
41. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Querying the uncertain position of moving objects. In *Temporal Databases: Research and Practice*, volume 1399, pages 310–337. Springer Verlag, 1998. 209, 211
42. V. S. Subrahmanian. *Principles of Multimedia Database Systems*. Morgan Kaufmann, 1998. 205

43. Peter van Beek. Temporal Query Processing with Indefinite Information. *Artificial Intelligence in Medicine*, 3:325–339, 1991. 206, 212, 219
44. Peter van Beek and Robin Cohen. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence*, 6:132–144, 1990. 218
45. 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 appears in JCSS, 54(1), pp. 113-135, 1997. 212, 219, 220
46. M. Vardi. The Complexity of Relational Query Languages. In *Proceedings of ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 137–146, 1982. 212
47. M. Vardi. Querying Logical Databases. *Journal of Computer and System Sciences*, 33:142–160, 1986. 205
48. Marc Vilain and Henry Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of AAAI-86*, pages 377–382, 1986. 219
49. Marc Vilain, Henry Kautz, and Peter van Beek. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, 1989. 219
50. M. Yannakakis. Expressing Combinatorial Optimization Problems by Linear Programs. In *Proc. of ACM Symposium on the Theory of Computing*, pages 223–288, 1988. 216