Querying Temporal and Spatial Constraint Networks in PTIME*

Manolis Koubarakis
Dept. of Electronic and Computer Engineering,
Technical University of Crete
University Campus, Kounoupidiana
73 100 Chania, Crete, Greece
manolis@ced.tuc.gr, www.ced.tuc.gr/~manolis

Spiros Skiadopoulos
Knowledge and Database Systems Laboratory
Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73 Athens, Greece
spiros@dbnet.ece.ntua.gr

August 31, 2000

Abstract

We start with the assumption that temporal and spatial knowledge usually captured by constraint networks can be represented and queried more effectively by using the scheme of indefinite constraint databases. Because query evaluation in this scheme is in general a hard computational problem, we seek tractable instances of query evaluation. We assume that we have a class of constraints $\mathcal{C}$ with some reasonable computational and closure properties (the computational properties of interest are that the satisfiability problem and an appropriate version of the variable elimination problem for $\mathcal{C}$ should be solvable in PTIME). Under this assumption, we exhibit general classes of indefinite constraint databases and first-order modal queries for which query evaluation can be done with PTIME data complexity. We then search for tractable instances of $\mathcal{C}$ among the subclasses of Horn disjunctive linear constraints over the rationals. From previous research we know that the satisfiability problem for Horn disjunctive linear constraints is solvable in PTIME, but not the variable elimination problem. Thus we try to discover subclasses of Horn disjunctive linear constraints 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 our general tractability results with $\mathcal{C}$ ranging over the newly discovered tractable classes. Interesting tractable query answering problems for indefinite temporal and spatial constraint databases are identified in this way. We close our complexity

*A preliminary version of this paper appears in the proceedings of AAAI-99. A different version of this paper (aimed at a database audience) appears in the proceedings of the workshop STDBM99.

analysis by precisely outlining the frontier between tractable and possibly intractable query answering problems.

Keywords: constraint networks, constraint databases, temporal reasoning, spatial reasoning, tractable reasoning, computational complexity

1 Introduction

When temporal constraint networks are used in applications, their nodes represent the times when certain facts are true, or when certain events take place, or when events start or end. By labeling nodes with appropriate natural language expressions (e.g., *breakfast* or *walk*) and arcs by temporal relations, temporal constraint networks can be queried in useful ways. For example the query "Is it possible (or certain) that event *walk* happened after event *breakfast*?" or "What are the known events that come after event *breakfast*?" can be asked [BCPT94, BCPT97, vB91]. However, other kinds of queries cannot be asked even though the knowledge required to answer them might be available. These kinds of queries usually involve more than a temporal constraint e.g., "*Who* is certainly having breakfast before taking a walk?". This problem arises because temporal constraint networks do not have the required expressive power for representing all kinds of knowledge needed in a real application.

This situation has been understood by temporal reasoning researchers, and application-oriented systems where temporal reasoners were combined with more general knowledge representation systems have been implemented. These systems include EPILOG[1], Shocker[2], Telos [MBJK90], TMM [DM87, SBC92] and the database systems of [Kou97b, BCPT95]. EPILOG uses the temporal reasoner Timegraph [GS95], Shocker uses TIMELOGIC, Telos uses a subclass of Allen's interval algebra [All83] while TMM and the database systems of [Kou97b, BCPT95] use networks of difference constraints [DMP91].

Similar issues arise in spatial constraint networks [Her94, RN99] when they are used in the context of appropriate applications.

In this paper we start from the assumption that temporal and spatial knowledge usually captured by a constraint network can be represented more effectively if the network is complemented by a *database* for storing the information typically used to label the nodes of the network. In this way the constraint network is reserved for temporal and spatial information that can nicely be captured by constraints, while the database contains all other information (researchers in temporal and spatial databases usually call the latter information *thematic* [Sno95, Güt94]). The combined system can then be queried using a *first order modal query language*.

For the case of temporal applications, the above assumption has been made explicity in the TMM system [DM87, SBC92] and the temporal relational database models of [Kou97b, BCPT95]. Of these two database proposals the most general one is the scheme of *indefinite constraint databases* proposed in [Kou97b]. In this paper we redefine the scheme of [Kou97b] (using first order logic instead of relational database theory) and take it as the formalism in which we present our contributions.

Our results are original and can be summarized as follows:

1. Firstly, we observe that evaluating queries over indefinite constraint databases can be a difficult computational problem (NP-hard or co-NP-hard) even when

[1] See `www.cs.rochester.edu/research/epilog/`.
[2] See `www.cs.rochester.edu/research/kr-tools.html`.

the constraints involved are equalities and inequalities of atomic values [AKG91]. It is therefore important to seek *tractable* instances of query evaluation.

We want our tractability results to be very general so that they can be reused by other researchers. Thus we assume that we have an arbitrary class of constraints $\mathcal{C}$ with some reasonable computational and closure properties (the computational properties of interest are that the satisfiability problem and an appropriate version of the variable elimination problem for $\mathcal{C}$ should be solvable in PTIME). Under this assumption, we demonstrate various abstract classes of indefinite constraint databases and first-order modal queries for which query evaluation can be done with PTIME data complexity (Theorems 4.1 to 4.4).

These results provide us with *general theorems* that can be used by any future researcher who wants to discover tractable cases of query evaluation in databases with indefinite information that can be expressed by constraints from a class $\mathcal{C}$ satisfying the assumed properties. In fact in Section 7.2 we discuss how two important theorems dealing with indefinite information in relational databases [Gra91] easily follow from our work.

2. Then we try to find appropriate instantiations of the abstract class $\mathcal{C}$ among the subclasses of Horn disjunctive linear constraints over the rationals [JB96, Kou96, JB98]. This will enable us to make our abstract complexity results concrete and apply them to interesting temporal and spatial domains. We choose to restrict our search among the subclasses of Horn disjunctive linear constraints because they include many interesting classes of temporal and spatial constraints, and their satisfiability problem is solvable in PTIME [JB96, Kou96, JB98]. Unfortunately, this is not the case for the variable elimination problem [Yan88]. We therefore try to discover subclasses of the class of Horn-disjunctive linear constraints with the property that variable elimination can be performed in PTIME. The class of UTVPI$^{\neq}$ constraints is the largest class that we show to have this property. 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 rational number. The class of UTVPI$^{\neq}$ *constraints* also includes disequations of the form $\pm x_1 \neq c$ or $\pm x_1 \pm x_2 \neq c$.

   Our results on UTVPI$^{\neq}$ constraints (Section 6) are original and extend the results of [Kou95, Kou97a]. In the course of our developments we also provide efficient algorithms for consistency checking and global consistency enforcement for UTVPI and UTVPI$^{\neq}$ constraints (Sections 5 and 6). These algorithms are interesting independently of the rest of our results.

3. Finally, we restate our general results with the abstract class $\mathcal{C}$ ranging over the newly discovered tractable classes. Tractable query answering problems for various interesting temporal and spatial domains are identified in this way (Section 7). Two of these query answering problems (Theorems 7.1 (a) and 7.2 (a)) are significant extensions of tractable problems identified previously in [BCT95, vB91] while all others are new.

   We close our complexity analysis by showing that our tractability results identify *precisely* the frontier between tractable and possibly intractable cases of query evaluation when the constraints of the database and the query range over subclasses of Horn-disjunctive linear constraints. As the results of Section 7.3 demonstrate, even very simple increases in the expressive power of the query and database classes shown tractable can lead to NP-hardness or co-NP-hardness.

Summarizing our work, we can say that this paper contains results that will be of interest to people working on temporal reasoning, spatial reasoning, temporal databases, spatial databases (Sections 4 to 7) but also to people working on traditional constraint satisfaction problems (Sections 5 and 6).

The paper is organized as follows. The next section presents some preliminary concepts. In Section 3 we review the scheme of indefinite constraint databases and a method for query evaluation. In Section 4 we identify tractable cases of query evaluation. Sections 5 and 6 present subclasses of Horn disjunctive linear constraints with satisfiability and variable elimination problems that can be solved in PTIME. Section 7 uses the results of Sections 4, 5 and 6 to identify tractable query answering problems in temporal and simple spatial domains. Finally, Section 8 discusses future work.

2 Preliminaries

We start by introducing some useful concepts for the developments in forthcoming sections. We will deal with many-sorted first order languages [End72]. For each first-order language $\mathcal{L}$ we will define a structure $\mathcal{M}_{\mathcal{L}}$ that will give the *intended interpretation* of formulas of $\mathcal{L}$ (this is called the *intended structure* for $\mathcal{L}$). The theory $Th(\mathcal{M}_{\mathcal{L}})$ (i.e., the set of sentences of $\mathcal{L}$ that are true in $\mathcal{M}_{\mathcal{L}}$) will also be considered. Finally, for each language $\mathcal{L}$ a special class of formulas called *$\mathcal{L}$-constraints* will be defined. $\mathcal{L}$-constraints are assumed to be formulas without quantifiers.

2.1 The language of linear constraints

We start by defining the first order language LC. The logical symbols of LC include: parentheses, a countably infinite set of variables, the equality symbol $=$ and the standard sentential connectives. The non-logical symbols of LC include: a countably infinite set of constants (one for each rational numeral), the binary function symbols $+$ and $*$ (the symbol $*$ can only be applied to a variable and a constant) and the binary predicate symbol $<$.

The intended structure $\mathcal{M}_{LC}$ for interpreting the symbols of LC is defined as follows. The domain of $\mathcal{M}_{LC}$ is the set of rational numbers $\mathcal{Q}$.[3] $\mathcal{M}_{LC}$ assigns to each constant symbol, an element of $\mathcal{Q}$ (a rational number), to function symbol $+$ the addition operation for rational numbers, to function symbol $*$ the multiplication operation for rational numbers, and to predicate symbol $<$ the relation "less than" over the rational numbers.

Definition 2.1 *An LC-*constraint *or* linear constraint *is a formula of LC of the form $\sum_{i=1}^{n} a_i x_i \ \theta \ r$ where $a_1, \ldots, a_n, r$ are constants, $x_1, \ldots, x_n$ are variables and θ is $\leq$ or $<$. We freely use $\geq, >$ and $=$ as well.*

The class of linear constraints will be denoted by LIN.

Example 2.1 *Below we give some examples of linear constraints:*

$$2x_1 + 3x_2 - 4x_3 < 12, \ x_1 > 2, \ x_1 - x_2 = 0, \ x_3 - x_4 \leq 5$$

Definition 2.2 *Let C be a set of linear constraints in variables $x_1, \ldots, x_n$. The* solution set *of C, denoted by $Sol(C)$, is the following relation:*

$$\{(x_1^0, \ldots, x_n^0) : \ (x_1^0, \ldots, x_n^0) \in \mathcal{Q}^n \text{ and for every } c \in C, \ (x_1^0, \ldots, x_n^0) \text{ satisfies } c\}.$$

Each member of $Sol(C)$ is called a solution *of C.*

[3] The symbol $\mathcal{Q}$ is overloaded and will also refer to the only sort of LC throughout this paper. We hope this does not create any confusion to the readers.

The concepts of solution and solution set for an arbitrary class of constraints are defined similarly.

Definition 2.3 *A set of constraints (in some language $\mathcal{L}$) is called* satisfiable *or* consistent *if and only if its solution set is nonempty.*

Example 2.2 *The set of constraints of Example 2.1 is satisfiable. Tuple* $(3,3,1,0)$ *is one of its solutions.*

We will also consider classes of $\mathcal{L}$-constraints. A class will usually be denoted by $\mathcal{C}$ and its members will be called *$\mathcal{C}$-constraints.*

Definition 2.4 *Let $\mathcal{C}$ be a class of constraints. The problem of deciding whether a given set of $\mathcal{C}$-constraints is satisfiable will be denoted by $SAT(\mathcal{C})$.*

It is a very interesting question to determine the computational complexity of the problem $SAT(\mathcal{C})$ for a given class of constraints $\mathcal{C}$. In Sections 5, 6 and 7, we will see some elegant classes of constraints with PTIME satisfiability problems.

Complexity bounds for satisfiability (and other problems) are usually specified in terms of two parameters: the *number of variables* and the *number of constraints* in the input constraint set. When a complexity parameter is not specified explicitly, it can be implicitly assumed to be the size of the input constraint set. The *size* of a constraint set can be defined as the number of symbols of a binary alphabet needed for its encoding. These are the assumptions we use in the rest of this paper when we refer to computational problems with sets of constraints as inputs.

2.2 Quantifier and Variable Elimination

In this section we define the operations of quantifier and variable elimination. Quantifier elimination is an operation from mathematical logic [End72]. Variable elimination is an algebraic operation [Sch86]. As we will see below, quantifier elimination algorithms utilize variable elimination algorithms as subroutines. In the scheme of indefinite constraint databases introduced in Section 3, the operation of quantifier elimination is very useful because it can be used for query evaluation.

Definition 2.5 *Let Th be a theory in some first order 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 $Th \models \phi \equiv \phi'$.*

This definition is stronger than the traditional one where ϕ' is simply required to be quantifier-free [End72]. We require ϕ' to be in the above form because we do not want to deal with negations of $\mathcal{L}$-constraints.

Let Th be a theory in some first order language $\mathcal{L}$, and let ϕ be a formula. If Th admits elimination of quantifiers, then a quantifier-free formula ϕ' equivalent to ϕ can be computed in the following standard way [End72]:

1. Compute the prenex normal form $(Q_1 x_1) \cdots (Q_m x_m) \psi(x_1, \ldots, x_m)$ of ϕ.

2. If Q_m is $\exists$ then let $\theta_1 \vee \cdots \vee \theta_k$ be a disjunction equivalent to $\psi(x_1, \ldots, x_m)$ where the θ_i's are conjunctions of $\mathcal{L}$-constraints. Then *eliminate variable* x_m from each θ_i to compute θ_i' using a *variable elimination* algorithm for $\mathcal{L}$-constraints. The resulting expression is $\theta_1' \vee \cdots \vee \theta_k'$.

   If Q_m is $\forall$ then let $\theta_1 \vee \cdots \vee \theta_k$ be a disjunction equivalent to $\neg\psi(x_1, \ldots, x_m)$ where the θ_i's are conjunctions of $\mathcal{L}$-constraints. Then *eliminate variable x_m* from each θ_i to compute θ_i' as above. The resulting expression is $\neg(\theta_i' \vee \cdots \vee \theta_k')$.

3. Repeat step 2 to eliminate all remaining quantifiers and obtain the required quantifier-free formula.

Step 2 of the above algorithm assumes the existence of a variable elimination algorithm for conjunctions (or, equivalently, *sets*) of $\mathcal{L}$-constraints. The operation of variable elimination can be defined as follows.

Definition 2.6 *The operation of* variable elimination *takes as input a set C of $\mathcal{L}$-constraints with set of variables X and a subset Y of X, and returns a new set of constraints C' such that $Sol(C') = \Pi_{X \setminus Y}(Sol(C))$ where Π_Z is the standard operation of projection of a relation on a subset Z of its set of columns.*

For the class of linear constraints defined above variable elimination can be performed using Fourier's algorithm. Fourier's algorithm can be summarized as follows [Sch86]. 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.

Example 2.3 *Let C be the following set of linear constraints:*

$$x_3 \leq x_1, \ x_5 < x_1, \ x_1 - x_2 \leq 2, \ x_4 \leq x_5$$

The elimination of variable x_1 from C using Fourier's algorithm results in the following set:

$$x_3 - x_2 \leq 2, \ x_5 - x_2 < 2, \ x_4 \leq x_5.$$

Definition 2.7 *Let $\mathcal{C}$ be a class of constraints. The problem of eliminating an arbitrary number of variables from a given set of $\mathcal{C}$-constraints will be denoted by VAR-$ELIM(\mathcal{C})$. If the number of variables is fixed then the problem will be denoted by $FVAR$-$ELIM(\mathcal{C})$.*

It is a very interesting question to determine the computational complexity of the problems VAR-$ELIM(\mathcal{C})$ and $FVAR$-$ELIM(\mathcal{C})$ for a given class of constraints $\mathcal{C}$. [4] In Sections 5 and 6 we will see some nice classes of constraints with PTIME variable elimination problems.

The following theorem is well-known and will be relied on in this paper without proof.

Theorem 2.1 *The theory $Th(\mathcal{M}_{LC})$ admits quantifier elimination.*

The presentation of preliminary concepts is now complete. We can therefore proceed to define the scheme of indefinite constraint databases.

[4] The complexity of the second problem is usually trivial to compute. For all interesting contraint classes that we can think of, $FVAR$-$ELIM(\mathcal{C})$ can be solved in PTIME.

3 The Scheme of Indefinite Constraint Databases

In this section we present the scheme of indefinite constraint databases originally proposed in [Kou97b]. We follow the spirit of the original proposal but use first order logic instead of relational database theory.

We assume the existence of a many-sorted first-order language $\mathcal{L}$ with a fixed intended structure $\mathcal{M}_{\mathcal{L}}$. Let us also assume that $Th(\mathcal{M}_{\mathcal{L}})$ *admits quantifier elimination* (Section 2.2 has defined this concept precisely). For the purposes of this paper $\mathcal{L}$ can be a language like LC that can be used to describe temporal and spatial objects (e.g., points, intervals or polygons).

Let us now consider, as an example, the information contained in the following two sentences:

> Mary took a walk in the park. After walking around for a while, she met Fred and started talking to him.

The information in the above sentences is about activities (e.g., walking, talking), constraints on the times of their occurence (e.g., after) and, finally, other information about real-world entities (e.g., names of persons). Temporal constraint networks [All83, vB92, DMP91] can be used to represent such information by capturing temporal constraints in their edges and storing all other information as node labels.

In the scheme of indefinite constraint databases (and in similar formalisms like [DM87, SBC92, BCPT95]) information like the above is represented by utilising a first-order temporal language like LC and extending it to represent non-temporal (i.e., thematic) information. Let us now show how to do this formally in an abstract setting by considering an arbitrary many-sorted first order language $\mathcal{L}$ with the properties discussed above.

3.1 From $\mathcal{L}$ to $\mathcal{L} \cup \mathcal{EQ}$ and $(\mathcal{L} \cup \mathcal{EQ})^*$

Let $\mathcal{EQ}$ be a fixed first order language with only equality ($=$) and a countably infinite set of constant symbols. The intended structure $\mathcal{M}_{\mathcal{EQ}}$ for $\mathcal{EQ}$ interprets $=$ as equality and constants as "themselves". $\mathcal{EQ}$ is a very simple language which can only be used to represent knowledge about things that are or are not equal. *$\mathcal{EQ}$-constraints* or *equality constraints* are formulas of the form $x = v$ or $x \neq v$ where x is a variable, and v is a variable or a constant.

We now consider the language $\mathcal{L} \cup \mathcal{EQ}$. The set of sorts for $\mathcal{L} \cup \mathcal{EQ}$ will contain the special sort $\mathcal{D}$ (for terms of $\mathcal{EQ}$) and all the sorts of $\mathcal{L}$. The intended structure for $\mathcal{L} \cup \mathcal{EQ}$ is $\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}} = \mathcal{M}_{\mathcal{L}} \cup \mathcal{M}_{\mathcal{EQ}}$.

The following lemma is straighforward.

Lemma 3.1 *If theory $Th(\mathcal{M}_{\mathcal{L}})$ admits quantifier elimination then the same holds for $Th(\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}})$.*

Finally, we define a new first order language $(\mathcal{L} \cup \mathcal{EQ})^*$ by augmenting $\mathcal{L} \cup \mathcal{EQ}$ with a countably infinite set of *database predicate symbols* $p_1, p_2, \ldots$ of various arities. These predicate symbols can be used to express thematic information i.e., information with no special temporal or spatial semantics (e.g., the name of the person who went for a walk is Mary). The indefinite constraint databases and queries defined below are formulas of $(\mathcal{L} \cup \mathcal{EQ})^*$.

In Section 4 we will also need the language $\mathcal{L}^*$ which is obtained by augmenting $\mathcal{L}$ with a countably infinite set of database predicate symbols of various arities. $\mathcal{L}^*$ can be used instead of $(\mathcal{L} \cup \mathcal{EQ})^*$ whenever we do not want to consider the equality constraints offered to us by $\mathcal{EQ}$.

Example 3.1 *Let $\mathcal{L}$ be the language LC defined in Section 2. Let walk be a ternary database predicate symbol with arguments of sort $\mathcal{D}$, $\mathcal{Q}$ and $\mathcal{Q}$ respectively. The following is a formula of the language $(LC \cup \mathcal{EQ})^*$ capturing the fact that somebody took a walk during some unknown interval of time:*

$$(\exists x/\mathcal{D})(\exists t_1/\mathcal{Q})(\exists t_2/\mathcal{Q})(t_1 < t_2 \ \wedge \ walk(x, t_1, t_2))$$

3.2 Databases And Queries

In this section the symbols $\overline{\mathcal{T}}$ and $\overline{\mathcal{T}}_i$ will denote vectors of sorts of $\mathcal{L}$. Similarly, the symbol $\overline{\mathcal{D}}$ will denote a vector with all its components being the sort $\mathcal{D}$.

Indefinite constraint databases and queries are special formulas of $(\mathcal{L} \cup \mathcal{EQ})^*$ and are defined as follows.

Definition 3.1 *An* indefinite constraint database *is a formula $DB(\overline{\omega})$ of $(\mathcal{L} \cup \mathcal{EQ})^*$ of the following form:*

$$\bigwedge_{i=1}^{m} (\forall \overline{x_i}/\overline{\mathcal{D}})(\forall \overline{t_i}/\overline{\mathcal{T}}_i)(\bigvee_{j=1}^{l_i} Local_j(\overline{x_i}, \overline{t_i}, \overline{\omega}) \equiv p_i(\overline{x_i}, \overline{t_i})) \quad \wedge$$

$$ConstraintStore(\overline{\omega})$$

where

- *$Local_j(\overline{x_i}, \overline{t_i}, \overline{\omega})$ is a conjunction of $\mathcal{L}$-constraints in variables $\overline{t_i}$ and Skolem constants $\overline{\omega}$, and $\mathcal{EQ}$-constraints in variables $\overline{x_i}$.*
- *$ConstraintStore(\overline{\omega})$ is a conjunction of $\mathcal{L}$-constraints in Skolem constants $\overline{\omega}$.*

The second component of the above formula defining a database is a *constraint store*. This store is a conjunction of $\mathcal{L}$-constraints and corresponds to a constraint network. $\overline{\omega}$ is a vector of *Skolem constants* denoting entities (e.g., points and intervals in time or points and regions in a multi-dimensional space) about which *only partial knowledge* is available. This partial knowledge has been coded in the constraint store using the language $\mathcal{L}$.

The first component of the database formula is a set of equivalences *completely defining* the database predicates p_i (this is an instance of the well-known technique of predicate completion in first order databases [Rei84]).

These equivalences may refer to the Skolem constants of the constraint store. In temporal reasoning applications, the constraint store will contain the temporal constraints usually captured by a constraint network, while the predicates p_i will encode, in a flexible way, the events or facts usually associated with the nodes of this constraint network.

For a given database DB the first conjunct of the database formula will be denoted by $EventsAndFacts(DB)$, and the second one by $ConstraintStore(DB)$. For clarity we will sometimes write sets of conjuncts instead of conjunctions. In other words a database DB can be seen as the following pair of sets of formulas:

$$(EventsAndFacts(DB), \ ConstraintStore(DB)).$$

We will feel free to use whichever definition of database fits our needs in the rest of this paper.

The new machinery in the indefinite constraint database scheme (in comparison with relational or Prolog databases) is the Skolem constants in $EventsAndFacts(DB)$ and the constraint store which is used to represent "all we know" about these Skolem constants. Essentially this proposal is a combination of constraint databases (without indefinite information) as defined in [KKR90], and the marked null values proposal of [IL84, Gra91]. Similar ideas can also be found in the first order databases of [Rei84].

Let us now give some examples of indefinite constraint databases. The constraint language used is LC.

Example 3.2 *The following is an indefinite constraint database which formalises the information in the paragraph considered at the beginning of this section.*

$$\begin{gathered}(\ \{ \ (\forall x/\mathcal{D})(\forall t_1, t_2/\mathcal{Q})((x = Mary \ \wedge \ t_1 = \omega_1 \ \wedge \ t_2 = \omega_2) \equiv walk(x, t_1, t_2)), \\ (\forall x/\mathcal{D})(\forall y/\mathcal{D})(\forall t_3, t_4/\mathcal{Q}) \\ ((x = Mary \ \wedge \ y = Fred \ \wedge \ t_3 = \omega_3 \ \wedge \ t_4 = \omega_4) \equiv talk(x, y, t_3, t_4)) \ \}, \\ \{ \ \omega_1 < \omega_2, \ \omega_1 < \omega_3, \ \omega_3 < \omega_2, \ \omega_3 < \omega_4 \ \} \ )\end{gathered}$$

This database contains information about the events walk and talk in which Mary and Fred participate. The temporal information expressed by order constraints is indefinite since we do not know the exact constraint between Skolem constants ω_2 and ω_4.

Example 3.3 *Let us consider the following planning database used by a medical laboratory for keeping track of patient appointments for the year 1996.*

$$\begin{gathered}(\ \{ \ (\forall x, y/\mathcal{D})(\forall t_1, t_2/\mathcal{Q}) \\ (((x = Smith \ \wedge \ y = Chem1 \ \wedge \ t_1 = \omega_1 \ \wedge \ t_2 = \omega_2) \ \vee \\ (x = Smith \ \wedge \ y = Chem2 \ \wedge \ t_1 = \omega_3 \ \wedge \ t_2 = \omega_4) \ \vee \\ (x = Smith \ \wedge \ y = Radiation \ \wedge \ t_1 = \omega_5 \ \wedge \ t_2 = \omega_6)) \equiv treatment(x, y, t_1, t_2)) \ \}, \\ \{ \ \omega_1 \geq 0, \ \omega_2 \geq 0, \ \omega_3 \geq 0, \ \omega_4 \geq 0, \ \omega_5 \geq 0, \ \omega_6 \geq 0, \\ \omega_2 = \omega_1 + 1, \ \omega_4 = \omega_3 + 1, \ \omega_6 = \omega_5 + 2, \ \omega_2 \leq 91, \ \omega_3 \geq 91, \ \omega_4 \leq 182, \\ \omega_3 - \omega_2 \geq 60, \ \omega_5 - \omega_4 \geq 20, \ \omega_6 \leq 213 \ \} \ )\end{gathered}$$

In this example the set of rationals $\mathcal{Q}$ is 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 above database represents the following information:

1. *There are three scheduled appointments for treatment of patient Smith. This is represented by three conjuncts in the disjunction defining the extension of predicate treatment.*

2. *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 $\omega_6 = \omega_5 + 2$.*

3. *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 the constraints $\omega_1 \geq 0$ and $\omega_2 \leq 91$.*

4. *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 $\omega_4 \leq 182$.*

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

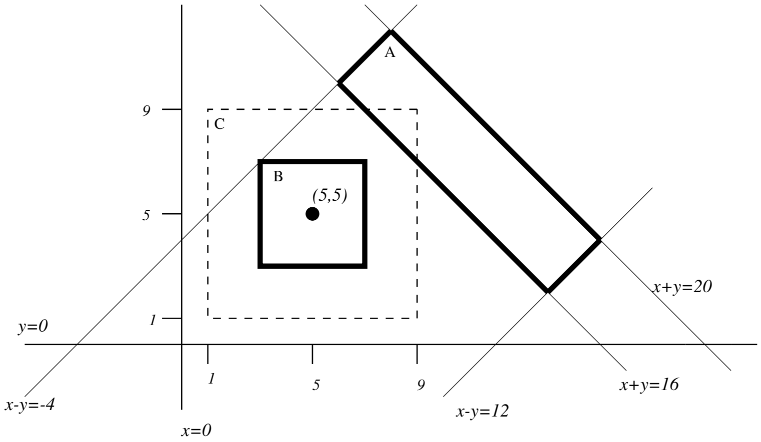

Figure 1: The areas of Example 3.4

6. *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* $\omega_6 \leq 213$.

Let us now give an example involving indefinite spatial information. In this paper we only deal with indefinite spatial information that can be expressed by linear constraints (i.e., we do not consider other interesting kinds of spatial constraints e.g., topological [RN99] or directional [Lig98]).

Example 3.4 *Consider Figure 1 and assume that there is some species inhabiting the rectangular area A. Now assume that there is some form of atmospheric pollution which has been generated due to some industrial accident at point* $(5,5)$. *The extent of the pollution is not known precisely at this time. All we know is that the minimum area polluted is given by rectangle B and the maximum by rectangle C. The following indefinite constraint database represents all the known information.*

$$\begin{gathered} (\ \{\ (\forall x, y/\mathcal{Q})((5-\omega \leq x\ \wedge\ x \leq 5+\omega\ \wedge\ 5-\omega \leq y\ \wedge\ y \leq 5+\omega) \equiv polluted(x,y)), \\ (\forall x, y/\mathcal{Q})(x+y \leq 20\ \wedge\ x+y \geq 16\ \wedge\ x-y \leq 12\ \wedge\ x-y \geq -4 \equiv species(x,y))\ \}, \\ \{\omega \geq 2,\ \omega \leq 4\}\ ) \end{gathered}$$

Let us now define queries. The concept of query defined here is more expressive than the query languages for temporal constraint networks proposed in [BCPT94, BCPT97, vB91], and it is similar to the concept of query in TMM [SBC92].

Definition 3.2 *A* first order modal query *over an indefinite constraint database is an expression of the form* $\overline{x}/\overline{\mathcal{D}}, \overline{t}/\overline{\mathcal{T}}\ :\ OP\ \phi(\overline{x},\overline{t})$ *where* OP *is the modal operator* $\Diamond$ *or* $\Box$, *and* ϕ *is a formula of* $(\mathcal{L} \cup \mathcal{EQ})^*$. *The constraints in formula* ϕ *are only* $\mathcal{L}$*-constraints and* $\mathcal{EQ}$*-constraints.*

Modal queries will be distinguished as *certainty* or *necessity* queries ($\Box$) and *possibility* queries ($\Diamond$).

Example 3.5 *The following query refers to the database of Example 3.2 and asks "Who was the person who possibly had a conversation with Fred during this person's walk in the park?":*

$$\begin{gathered} x/\mathcal{D}:\ \Diamond(\exists t_1, t_2, t_3, t_4/\mathcal{Q}) \\ (walk(x,t_1,t_2)\ \wedge\ talk(x,Fred,t_3,t_4)\ \wedge\ t_1 < t_3\ \wedge\ t_4 < t_2) \end{gathered}$$

Let us observe that each query can only have *one* modal operator which should be placed in front of a formula of $(\mathcal{L} \cup \mathcal{EQ})^*$. Thus we do not have a full-fledged modal query language like the ones in [Lev84, Lip79, Rei88]. Such a query language can be beneficial in any application involving indefinite information but we will not consider this issue in this paper.

We now define the concept of an answer to a query.

Definition 3.3 *Let q be the query $\overline{x}/\overline{\mathcal{D}}, \overline{t}/\overline{\mathcal{T}} : \Diamond\phi(\overline{x}, \overline{t})$ over an indefinite constraint database DB. The answer to q is a pair $(answer(\overline{x}, \overline{t}),\ \emptyset)$ such that*

1. *$answer(\overline{x}, \overline{t})$ is a formula of the form*

$$\bigvee_{j=1}^{k} Local_j(\overline{x}, \overline{t})$$

   where $Local_j(\overline{x}, \overline{t})$ is a conjunction of $\mathcal{L}$-constraints in variables $\overline{t}$ and $\mathcal{EQ}$-constraints in variables $\overline{x}$.

2. *Let V be a variable assignment for variables $\overline{x}$ and $\overline{t}$. If there exists a model M of DB which agrees with $\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}}$ on the interpretation of the symbols of $\mathcal{L} \cup \mathcal{EQ}$, and M satisfies $\phi(\overline{x}, \overline{t})$ under V then V satisfies $answer(\overline{x}, \overline{t})$ and vice versa.*

We have chosen the notation $(answer(\overline{x}, \overline{t}),\ \emptyset)$ to signify that *an answer is also a database* which consists of a single predicate defined by the formula $answer(\overline{x}, \overline{t})$ and the empty constraint store. In other words, no Skolem constant (i.e., no uncertainty) is present in the answer to a modal query. Although our databases may contain uncertainty, we know for sure what is possible and what is certain.

Example 3.6 *The answer to the query of Example 3.5 is $(x = Mary,\ \emptyset)$.*

The definition of answer in the case of certainty queries is the same as Definition 3.3 with the second condition changed to:

2. *Let M be any model of DB which agrees with $\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}}$ on the interpretation of the symbols of $\mathcal{L} \cup \mathcal{EQ}$. Let V be a variable assignment for variables $\overline{x}$ and $\overline{t}$. If M satisfies $\phi(\overline{x}, \overline{t})$ under V then V satisfies $answer(\overline{x}, \overline{t})$ and vice versa.*

Definition 3.4 *A query is called* closed *or* yes/no *if it does not have any free variables. Queries with free variables are called* open.

Example 3.7 *The query of Example 3.5 is open. The following is its corresponding closed query:*

$$: \ \Diamond(\exists x/\mathcal{D})(\exists t_1, t_2, t_3, t_4/\mathcal{Q})$$
$$(walk(x, t_1, t_2)\ \wedge\ talk(x, Fred, t_3, t_4)\ \wedge\ t_1 < t_3\ \wedge\ t_4 < t_2)$$

By convention, when a query is closed, its answer can be either $(true, \emptyset)$ (which means *yes*) or $(false, \emptyset)$ (which means *no*).

Example 3.8 *The answer to the query of Example 3.7 is $(true, \emptyset)$ i.e., yes.*

Let us now give some more examples of queries.

Example 3.9 *Let us consider the database of Example 3.3 and the query "Find all appointments for patients that can possibly start at the 92th day of 1996". This query can be expressed as follows:*

$$\{\ x, y/\mathcal{D} :\ \Diamond(\exists t_1, t_2/\mathcal{Q})(treatment(x, y, t_1, t_2)\ \wedge\ t_1 = 92)\ \}$$

The answer to this query is the following:

$$(\ (x = Smith\ \wedge\ y = Chem2)\ \vee\ (x = Smith\ \wedge\ y = Radiation),\ \emptyset\ )$$

Example 3.10 *The following query refers to the database of Example 3.4 and asks "Is it certain that the polluted area overlaps the area inhabited by the species of interest?":*

$$:\ \Box(\exists x, y/\mathcal{Q})(polluted(x, y)\ \wedge\ species(x, y))$$

The answer to this query is no.

3.3 Query Evaluation is Quantifier Elimination

Query evaluation over indefinite constraint databases can be viewed as quantifier elimination in the theory $Th(\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}})$. $Th(\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}})$ admits quantifier elimination. This is a consequence of the assumption that $Th(\mathcal{M}_{\mathcal{L}})$ admits quantifier elimination (see beginning of this section) and the fact that $Th(\mathcal{M}_{\mathcal{EQ}})$ admits quantifier elimination (proved in [KKR95]). The following theorem is essentially from [Kou97b].

Theorem 3.1 *Let DB be the indefinite constraint database*

$$\bigwedge_{i=1}^{m}(\forall\overline{x_i}/\overline{\mathcal{D}})(\forall\overline{t_i}/\overline{\mathcal{T}}_i)(\bigvee_{j=1}^{l_i} Local_j(\overline{x_i}, \overline{t_i}, \overline{\omega}) \equiv p_i(\overline{x_i}, \overline{t_i}))\quad\wedge$$

$$ConstraintStore(\overline{\omega})$$

and q be the query $\overline{y}/\overline{\mathcal{D}}, \overline{z}/\overline{\mathcal{T}} :\ \Diamond\phi(\overline{y}, \overline{z})$. The answer to q is $(answer(\overline{y}, \overline{z}),\ \emptyset)$ where $answer(\overline{y}, \overline{z})$ is a disjunction of conjunctions of $\mathcal{EQ}$-constraints in variables $\overline{y}$ and $\mathcal{L}$-constraints in variables $\overline{z}$ obtained by eliminating quantifiers from the following formula of $\mathcal{L}_=$:

$$(\exists\overline{\omega}/\overline{\mathcal{T}}')(ConstraintStore(\overline{\omega})\ \wedge\ \psi(\overline{y}, \overline{z}, \overline{\omega}))$$

In this formula the vector of Skolem constants $\overline{\omega}$ has been substituted by a vector of appropriately quantified variables with the same name ($\overline{\mathcal{T}}'$ is a vector of sorts of $\mathcal{L}$). $\psi(\overline{y}, \overline{z}, \overline{\omega})$ is obtained from $\phi(\overline{y}, \overline{z})$ by substituting every atomic formula with database predicate p_i by an equivalent disjunction of conjunctions of $\mathcal{L}$-constraints. This equivalent disjunction is obtained by consulting the definition

$$\bigvee_{j=1}^{l_i} Local_j(\overline{x_i}, \overline{t_i}, \overline{\omega}) \equiv p_i(\overline{x_i}, \overline{t_i})$$

of predicate p_i in the database DB.

If q is a certainty query then $answer(\overline{y}, \overline{z})$ is obtained by eliminating quantifiers from the formula

$$(\forall\overline{\omega}/\overline{\mathcal{T}}')(ConstraintStore(\overline{\omega})\ \supset\ \psi(\overline{y}, \overline{z}, \overline{\omega}))$$

where $ConstraintStore(\overline{\omega})$ and $\psi(\overline{y}, \overline{z}, \overline{\omega})$ are defined as above.

Example 3.11 *Using the above theorem, the query of Example 3.5 can be answered by eliminating quantifiers from the formula:*

$$\begin{array}{c}(\exists \omega_1, \omega_2, \omega_3, \omega_4/\mathcal{Q}) \\ (\omega_1 < \omega_2 \ \wedge \ \omega_1 < \omega_3 \ \wedge \ \omega_3 < \omega_2 \ \wedge \ \omega_3 < \omega_4 \ \wedge \\ (\exists t_1, t_2, t_3, t_4/\mathcal{Q})((x = Mary \ \wedge \ t_1 = \omega_1 \ \wedge \ t_2 = \omega_2) \ \wedge \\ (x = Mary \ \wedge \ t_3 = \omega_3 \ \wedge \ t_4 = \omega_4) \ \wedge \ t_1 < t_3 \ \wedge \ t_4 < t_2)\end{array}$$

The result of this elimination is the formula $x = Mary$.

Example 3.12 *Using the above theorem, the query of Example 3.10 can be answered by eliminating quantifiers from the formula:*

$$\begin{array}{c}(\forall \omega/\mathcal{Q})(\omega \geq 2 \ \wedge \ \omega \leq 4 \ \supset \\ (\exists x, y/\mathcal{Q})(5 - \omega \leq x \ \wedge \ x \leq 5 + \omega \ \wedge \ 5 - \omega \leq y \ \wedge \ y \leq 5 + \omega \ \wedge \\ x + y \leq 20 \ \wedge \ x + y \geq 16 \ \wedge \ x - y \leq 12 \ \wedge \ x - y \geq -4))\end{array}$$

The result of this elimination is the formula $false$.

Let us close this section by pointing out that what we have defined is a *database scheme*. Given various choices for $\mathcal{L}$, one can use the developed formalism to study any kind of databases with indefinite information (e.g., temporal or spatial) that can be expressed by constraints of $\mathcal{L}$. The complexity results of the next section have been developed in a similar spirit. They talk about arbitrary constraint classes that satisfy certain properties. The instantiation of these results to classes of temporal and spatial constraints constraints is given in Section 7.

4 Tractable Query Evaluation in Indefinite Constraint Databases

In this section we consider the computational complexity of evaluating possibility and certainty queries over indefinite constraint databases. The complexity of query evaluation will be measured using the notion of *data complexity* originally introduced by database theoreticians [Var82]. When we use data complexity, we measure the complexity of query evaluation as a function of the database size only; the size of the query is considered *fixed*. This assumption is reasonable and it has also been made in previous work on querying temporal constraint networks [vB91]. For the purposes of this paper the *size* of the database under the data complexity measure can be defined as the number of symbols of a binary alphabet that are used for its encoding.

We already know that evaluating possibility queries over indefinite constraint databases can be NP-hard even when we only have equality and inequality constraints between atomic values [AKG91]; similarly evaluating certainty queries is co-NP-hard. It is therefore important to seek *tractable* instances of query evaluation.

As with the definitions of Section 3, we would like to develop our tractability results in a general way. Thus we assume the existence of a many-sorted first-order language $\mathcal{L}$ with a fixed intended structure $\mathcal{M}_{\mathcal{L}}$ such that $Th(\mathcal{M}_{\mathcal{L}})$ admits quantifier elimination. We also assume that we have subclasses of the class of $\mathcal{L}$-constraints with some reasonable computational and closure properties. Under these assumptions, we show that there are several classes of indefinite constraint databases and modal queries, for which query evaluation can be done with PTIME data complexity.

The rest of this paper does not consider equality constraints (from language $\mathcal{EQ}$) as they have been used in the definition of databases (Definition 3.1) and queries (Definition 3.2). This can be done without loss of generality because they do not change our results in any way. We reach tractable cases of query evaluation by restricting the classes of $\mathcal{L}$-constraints, databases and queries we allow. The concepts of query type and database type introduced below allow us to make these distinctions.

4.1 Query Types

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

$$Q(OpenOrClosed, Modality, FO\text{-}Formula\text{-}Type, Constraints)$$

The first argument of a query type can take the values *Open* or *Closed* and distinguishes between open and closed queries. The argument *Modality* can be $\Diamond$ or $\Box$ representing possibility or necessity queries respectively.

The third argument *FO-Formula-Type* can take the values

$$FirstOrder,\ PositiveExistential \text{ or } SinglePredicate.$$

The value *FirstOrder* denotes that the first-order expression part of the query can be an *arbitrary* first-order formula. Similarly, *PositiveExistential* denotes that the first order part of the query is a *positive existential* formula i.e., it is of the form $(\exists \overline{x}/\overline{s})\phi(\overline{x})$ where ϕ involves only the logical symbols $\wedge$ and $\vee$. Finally, *SinglePredicate* denotes 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 database predicate symbol and OP is a modal operator.

The fourth argument *Constraints* denotes the class of constraints that are used in the query. Definition 3.2 allows queries to contain any constraint from the class of $\mathcal{L}$-constraints. This section will also consider restricting query constraints to members of any constraint class $\mathcal{C}$ such that $\mathcal{C}$ is a subclass of the class of $\mathcal{L}$-constraints. For example the query

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

is in the query class $Q(Closed, \Box, PositiveExistential, UTVPI)$ and uses constraints from the class UTVPI (a subclass of linear constraints to be defined in Section 5). The special constraint class $NONE$ will also be used in the cases where we want to consider queries that contain database predicates but *no* constraints.

4.2 Database Types

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

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

Argument *Arity* denotes the maximum arity of the database predicates. It can take values

$$Monadic,\ Binary,\ Ternary, \ldots, N\text{-}ary \text{ (i.e., arbitrary).}$$

Argument *LocalCondition* denotes the constraint class used in the definition of the database predicates. Finally, argument *ConstraintStore* denotes the class of constraints in the constraint store. Definition 3.1 allows the local conditions and the constraint store to contain any constraint from the class of $\mathcal{L}$-constraints. This section will also consider restrictions to members of any constraint class $\mathcal{C}$ such that $\mathcal{C}$ is a subclass of the class of $\mathcal{L}$-constraints. For example a database having (i) N-ary predicates defined by a disjunction of conjunctions of linear equalities, and (ii) a constraint store which is a set of linear constraints can be represented by type $DB(N\text{-}ary, LINEQ, LIN)$ where LINEQ is the class of linear equalities.

4.3 Complexity Results

In this section we present our results assuming the data complexity measure. Before we proceed we need the following definitions.

Definition 4.1 *If $\mathcal{C}$ is a class of constraints then $\vee\mathcal{C}$ denotes a new class consisting of all possible disjunctions of $\mathcal{C}$-constraints.*

Definition 4.2 *A class of constraints $\mathcal{C}$ is* closed under variable elimination *if for every set C of $\mathcal{C}$-constraints with set of variables X and every subset Z of X, there exists a set C' of $\vee\mathcal{C}$-constraints such that $Sol(C') = \Pi_Z(Sol(C))$.*

The notion of closure under variable elimination is very useful in our formalism. During query processing if a variable is eliminated from a set constraints that are used to define a database predicate, the result of the elimination must define a new database predicate.

The class of linear constraints defined in Section 2 is closed under variable elimination. Also, as we will see, all the classes of constraints to be considered in Sections 5 and 6 are closed under variable elimination.

Let us now proceed with our results. Some proofs are straightforward and are omitted. We first consider closed queries.

Lemma 4.1 *Let $\mathcal{C}$ be a class of constraints which is closed under variable elimination. Evaluating a query of the class*

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

over an indefinite constraint database of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ is equivalent to deciding the consistency of m formulas of the form $CS(\overline{\omega}) \wedge \theta(\overline{\omega})$ where CS and θ are conjunctions of $\mathcal{C}$-constraints in variables $\overline{\omega}$, and m is a parameter depending on the size of the database.

Proof: A query of the class

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

is of the form : $\Diamond(\exists\bar{t})\phi(\bar{t})$ where ϕ is a formula of $\mathcal{L}^*$ with $\mathcal{C}$-constraints involving only the logical symbols $\wedge$ and $\vee$. According to Theorem 3.1, evaluating the above query is equivalent to performing quantifier elimination to the formula

$$(\exists\overline{\omega})(CS(\overline{\omega}) \ \wedge \ (\exists\bar{t})\xi(\bar{t}, \overline{\omega}))$$

where ξ is a formula of $\mathcal{L}$ with $\mathcal{C}$-constraints as atomic formulas. ξ is the formula which is derived from ϕ and the database. Using standard methods [KKR90, Kou97b], the above formula can be transformed into

$$(\exists\overline{\omega})(CS(\overline{\omega}) \wedge (\exists\bar{t})(\psi_1(\bar{t}, \overline{\omega}) \vee \ldots \vee \psi_k(\bar{t}, \overline{\omega})))$$

where ψ_i, $1 \leq i \leq k$ are conjunctions of $\mathcal{C}$-constraints in variables $\overline{\omega}$ and $\bar{t}$, and k is a parameter depending on the size of the database. Distributing $\exists$ over $\vee$ we arrive at the following formula:

$$(\exists\overline{\omega})(CS(\overline{\omega}) \wedge ((\exists\bar{t})\psi_1(\bar{t}, \overline{\omega}) \vee \ldots \vee (\exists\bar{t})\psi_k(\bar{t}, \overline{\omega})))$$

Then, we can eliminate $(\exists\bar{t})$ from each disjunct and get the formula

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

where each θ_i is a conjunction of $\mathcal{C}$-constraints (because $\mathcal{C}$ is closed under variable elimination), and $m \geq k$ is a parameter depending on the size of the database. Distributing $\wedge$ and $\exists$ over $\vee$ we have:

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

To answer this query we have to perform a satisfiability check for each of formulas $CS(\overline{\omega}) \wedge \theta_i(\overline{\omega})$. If we find a satisfiable one, we stop answering *yes*. Otherwise, if we exhaust all disjuncts and there is no satisfiable one, we answer *no*.

■

Theorem 4.1 *Let $\mathcal{C}$ be a class of constraints which is closed under variable elimination, and the problems $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{C})$ can be solved in PTIME. Let DB be an indefinite constraint database of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ and q be a query of the class $Q(Closed, \diamond, PositiveExistential, \mathcal{C})$. The problem of deciding whether $q(DB) = yes$ can be solved with PTIME data complexity.*

Proof: All steps of Lemma 4.1 can be done in PTIME. The parameters k and m are polynomials in the size of the database. The step where quantifier $(\exists\overline{t})$ is eliminated can be done in PTIME because the size of $\overline{t}$ is fixed and $FVAR\text{-}ELIM(\mathcal{C})$ can be done in PTIME. The satisfiability checks can be done in PTIME because $SAT(\mathcal{C})$ can be solved in PTIME. Therefore, the total computation for the evaluation is in PTIME. ■

The following definition is needed below.

Definition 4.3 *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.*

Lemma 4.2 *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E}$ is closed under variable elimination, $\mathcal{E} \subseteq \mathcal{C}$ 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 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 in variables $\overline{\omega}$, the θ_i's are $\mathcal{C}$-constraints, and m is a parameter depending on the size of the database.

Theorem 4.2 *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E}$ is closed under variable elimination, $\mathcal{E} \subseteq \mathcal{C}$, $\vee\overline{\mathcal{E}} \subseteq \mathcal{C}$ and the problems $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{E})$ can be solved in PTIME. Let DB be an indefinite constraint database of the class $DB(N\text{-}ary, \mathcal{E}, \mathcal{C})$ and q be a query of the class $Q(Closed, \Box, PositiveExistential, \mathcal{E})$. The problem of deciding whether $q(DB) = yes$ can be solved with PTIME data complexity.*

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

Lemma 4.3 *Let $\mathcal{C}$ be a class of constraints which is closed under variable elimination. Evaluating a query of the class*

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

over an indefinite constraint database of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ is equivalent to eliminating quantifiers from m formulas of the form $(\exists\overline{\omega})(CS(\overline{\omega}) \wedge \theta(\overline{u}, \overline{\omega}))$ where CS and θ are conjunctions of $\mathcal{C}$-constraints, $\overline{u}$ is the vector of free variables of the query, and m is a parameter depending on the size of the database.

Theorem 4.3 *Let $\mathcal{C}$ be a class of constraints which is closed under variable elimination, and the problem VAR-ELIM($\mathcal{C}$) can be solved in PTIME. Let DB be an indefinite constraint database of the class $DB(N\text{-}ary, \mathcal{C}, \mathcal{C})$ and q be a query of the class $Q(Open, \Diamond, PositiveExistential, \mathcal{C})$. The problem of evaluating $q(DB)$ can be solved with PTIME data complexity.*

Before we proceed to the next result, we give two useful definitions.

Definition 4.4 *Let $\mathcal{C}$ be a class of constraints. $\mathcal{C}$ is* closed under negation *if the negation of every $\mathcal{C}$-constraint is equivalent to a single $\mathcal{C}$-constraint.*

Definition 4.5 *Let $\mathcal{E}$ and $\mathcal{C}$ be two classes of constraints, and k be a positive integer. Then $\overline{\mathcal{E}} \vee \mathcal{C}_{\leq k}$ will denote the class of constraints such that each of its elements is a disjunction of an arbitrary number of negations of $\mathcal{E}$-constraints and at most k $\mathcal{C}$-constraints.*

The next lemma and theorem deal with a special case of the general database type specification

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

Here (but also in Section 7) we consider the type of a database which consists of predicates defined by a disjunction of conjunctions with the following additional property: each conjunction consists of an arbitrary number of conjuncts that are $\mathcal{E}$-constraints and *at most* k conjuncts that are $\mathcal{C}$-constraints (where $\mathcal{E}, \mathcal{C}$ are classes of constraints and k is a positive integer).[5] We will use the notation $\mathcal{E} \cup \mathcal{C}_{\leq k}$ to refer to conjunctions (or sets) of constraints of this type, and reserve the notation

$$DB(Arity, \mathcal{E} \cup \mathcal{C}_{\leq k}, ConstraintStore)$$

for the corresponding database type.

Lemma 4.4 *Let $\mathcal{E}, \mathcal{C}$ be classes of constraints such that $\mathcal{C}$ is closed under negation and constraint sets of the type $\mathcal{E} \cup \mathcal{C}_{\leq 1}$ are closed under variable elimination.[6] Evaluating a query of the class $Q(Open, \Box, SinglePredicate, NONE)$ over an indefinite constraint database of the class*

$$DB(N\text{-}ary, \mathcal{E} \cup \mathcal{C}_{\leq 1}, \mathcal{C})$$

is equivalent to eliminating quantifiers from a formula of the form

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

where CS is a conjunction of constraints in $\mathcal{C}$, each θ_i is a constraint in $\overline{\mathcal{E}} \vee \mathcal{C}_{\leq 1}$, $\overline{u}$ is the vector of free variables of the query, and m is parameter depending on the size of the database.

Theorem 4.4 *Let $\mathcal{E}, \mathcal{C}$ be classes of constraints. Let us also assume that $\mathcal{C}$ is closed under negation, constraint sets of type $\mathcal{E} \cup \mathcal{C}_{\leq 1}$ are closed under variable elimination, and the problems FVAR-ELIM($\mathcal{E} \cup \mathcal{C}_{\leq 1}$) and VAR-ELIM($\overline{\mathcal{E}} \vee \mathcal{C}_{\leq 1}$) can be solved in PTIME. Let DB be an indefinite constraint database of the class*

$$DB(N\text{-}ary, \mathcal{E} \cup \mathcal{C}_{\leq 1}, \mathcal{C})$$

and q be a query of the class $Q(Open, \Box, SinglePredicate, NONE)$. The problem of evaluating $q(DB)$ can be solved with PTIME data complexity.

[5] Our definition is general but only the values $k = 1$ and $k = 2$ will be used.

[6] Instead of a formal definition for the term "a set of type ... is closed under variable elimination", we refer the reader to the similar Definition 4.2.

This section presented *general* results on tractable query answering problems for the scheme of indefinite constraint databases. Our theorems can be used by any future researcher who wants to discover tractable cases of query evaluation in databases with indefinite information that can be expressed by constraints from a class $\mathcal{C}$ satisfying the assumed properties.

The careful reader would have noticed that the results for possibility queries (Theorems 4.1 and 4.3) are possibly more interesting than the results for certainty queries (Theorems 4.2 and 4.4) because they impose only reasonable assumptions on databases and queries. In the case of necessity queries we were forced to restrict databases and queries quite severely in order to come up with tractable query evaluation problems. We do not view this as a shortcoming of our work since it agrees with the experiences of other researchers [Gra91, AKG91, vdM92].

Our final goal is to see how the abstract results of this section can become concrete by considering various useful languages $\mathcal{L}$ of linear constraints. To achieve this, we have to find subclasses of linear constraints that have the nice properties assumed by the theorems of this section. This is what is done in Sections 5 and 6.

5 The Class of UTVPI Constraints

Linear constraints have been studied a lot in computer science, and more recently in temporal and spatial reasoning. The following theorem is a well-known result from linear programming.

Theorem 5.1 ([Kac79]) *The satisfiability of a set of linear constraints can be decided in PTIME.*

The class HDL of Horn-disjunctive linear constraints is an interesting extension of the class of linear constraints because the satisfiability problem still remains in PTIME.

Definition 5.1 ([Kou96, JB96, JB98]) *A* Horn-disjunctive linear constraint *or* HDL constraint *is a formula of LC of the form $d_1 \vee ... \vee d_n$ 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.*

HDL constraints are formulas of *LC* therefore they are defined over the sets of rational numbers. However, the results of this paper do not change if we consider HDL constraints over the real numbers (density is the only important property). Similar results *cannot* be developed for the case of integers because the presence of disequations and the discreteness property quickly lead to intractability (NP-completeness of the satisfiability problem).

[JB98, JB96, Kou96] have shown that HDL constraints subsume many interesting classes of temporal and simple spatial constraints.

Example 5.1 *The following is a set of HDL constraints:*

$$3x_1 + x_5 - 3x_4 \leq 10$$

$$x_1 + x_3 + x_5 \neq 7$$

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

$$4x_1 + x_3 \neq 3 \vee 5x_2 + 3x_5 + x_4 \neq 5$$

Theorem 5.2 ([Kou96, JB96, JB98, CJK97]) *The satisfiability of a set of HDL constraints can be decided in PTIME.*

The class HDL is closed under variable elimination [Kou96]. However, we do not have a nice theorem like the above concerning the complexity of variable elimination. 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 [Yan88].

The following is a weaker result which considers HDL constraints and a fixed number of variables.

Theorem 5.3 ([Kou96]) *We can eliminate a fixed number of variables from a set of Horn disjunctive linear constraints in PTIME.*

Variable elimination is very closely connected to the concept of global consistency [Dec92].

Definition 5.2 *Let C be a set of constraints in variables $x_1, \ldots, x_n$. For any i such that $1 \leq i \leq n$, $C(x_1, \ldots, x_i)$ will denote the set of constraints in C involving* only *variables $x_1, \ldots, x_i$.*

Definition 5.3 *Let C be a set of HDL constraints in variables $x_1, \ldots, x_n$ and $1 \leq i \leq n$. C is called* i-consistent *iff for every $i-1$ distinct variables $x_1, \ldots, x_{i-1}$, every tuple $u = (x_1^0, \ldots, x_{i-1}^0) \in \mathcal{Q}^{i-1}$ such that u satisfies the constraints $C(x_1, \ldots, x_{i-1})$ and every variable x_i different from $x_1, \ldots, x_{i-1}$, there exists $x_i^0 \in \mathcal{Q}$ such that u can be extended to a tuple $(x_1^0, \ldots, x_{i-1}^0, x_i^0)$ which satisfies the constraints $C(x_1, \ldots, x_{i-1}, x_i)$. C is called* strong i-consistent *if it is j-consistent for every j, $1 \leq j \leq i$. C is called* globally consistent *iff it is i-consistent for every i, $1 \leq i \leq n$.*

The above definition can be given similarly for an arbitrary class of constraints [Dec92].

If we have a globally consistent set then eliminating variables is trivial: we discard the constraints involving the eliminated variables and the remaining constraints form our answer. This property of globally consistent sets has most recently been used in the constraint query algebras of [Gol97, GK97].

Given the above, the question that naturally comes to mind is the following. Are there *subclasses* of HDL constraints where variable elimination and global consistency can be done in PTIME? We already know one such class.

Definition 5.4 *A* DIFF$^{\neq}$ constraint *is a formula $t_1 \sim c$ or $t_1 - t_2 \sim c$ where t_1, t_2 are variables ranging over the rational numbers, c is a rational constant and $\sim$ is $\leq$ or $\neq$ [Kou97a, GC95].*

The above constraint class extends the class of *difference constraints* or *DIFF constraints* studied in temporal reasoning [DMP89] by adding disequations of the form $x_1 - x_2 \neq c$ or $x_1 \neq c$. DIFF constraints have been used in the database of Example 3.3.

DIFF$^{\neq}$ constraints have some nice computational properties as the following theorem shows.

Theorem 5.4 ([Kou97a]) *Let C be a set of* DIFF$^{\neq}$ *constraints. We can enforce global consistency of C (and thus 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 (or DIFF$^{\neq}$) constraints without sacrificing its nice computational properties? This question is answered in this and the next section.

Two natural extensions of DIFF constraints are the classes of *TVPI constraints* and *UTVPI constraints* [Sho81, JMSY94]. TVPI is an acronym for linear inequalities with at most *T*wo *V*ariables *P*er *I*nequality. UTVPI constraints are TVPI constraints with *U*nit coefficients.

Definition 5.5 *A* TVPI constraint *is a formula* $ax_1 + bx_2 \leq c$ *where* x_1, x_2 *are variables ranging over the rational numbers and* a, b, c *are rational constants. A* UTVPI constraint *is a TVPI constraint where* $a, b \in \{-1, 0, 1\}$.

Example 5.2 *Below we give some examples of TVPI and UTVPI constraints:*

$$-x_1 \leq 12, \; x_1 + x_2 \leq 2, \; 3x_3 - 4x_2 \leq 6$$

UTVPI constraints have also been used in the database of Example 3.4.

TVPI constraints are a very useful class of linear constraints for spatial databases over $\mathcal{Q}^2$ since they can be used to define any kind of polygon. The restriction to unit coefficients is also useful since several kinds of polygons can be defined using only UTVPI constraints (e.g., arbitrary rectangles, several kinds of triangles and octagons and so on). [Stu97] has proposed to use UTVPI constraints (instead of the standard bounding boxes) for indexing arbitrary k-dimensional objects in spatial databases. In this paper we will only consider UTVPI constraints. UTVPI constraints *over the integers* have also been studied in [JMSY94, HS97].

UTVPI constraints in variables $x_1, \ldots, x_n$ can be represented by an undirected labelled graph $G_C = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ and for every constraint $c \in C$ involving variables x_i and x_j there is an edge $(v_i, v_j) \in E$ labelled with c [Sho81]. We will refer to the graph representation of UTVPI constraints only when we estimate the computational complexity of our algorithms.

In this section we show that deciding consistency, performing variable elimination and enforcing global consistency for UTVPI constraints can be done in PTIME. These results follow easily from the results of [JMSY94] on UTVPI constraints over the integers and are included here only because they will be used in Section 6. One original contribution of this section is the complexity bound of Theorem 5.6 in terms of the parameter width in the tradition of directional consistency algorithms [DP88].

If x_i and x_j are rational variables then there are *four* possible types of UTVPI constraints between them:

$$x_i + x_j \leq s_{ij}, \; x_i - x_j \leq d_{ij}, \; x_j - x_i \leq d_{ji}, \; -x_i - x_j \leq s'_{ij}$$

Equivalently $-s'_{ij} \leq x_i + x_j \leq s_{ij}, \;\; -d_{ji} \leq x_i - x_j \leq d_{ij}$.

Theorem 5.5 *Consider the following system of UTVPI constraints:*

$$x_i + x_j \leq s_{ij}, \; x_i - x_j \leq d_{ij}, \; x_j - x_i \leq d_{ji}, \; -x_i - x_j \leq s'_{ij}.$$

This system has a solution iff $-s'_{ij} \leq s_{ij}$ *and* $-d_{ji} \leq d_{ij}$.

The algorithm in Figure 2 decides whether a given set of UTVPI constraints C is consistent. The algorithm repeatedly computes some form of *limited transitive closure* of the input constraint set. More precisely, it propagates constraints on pairs of variables (x_i, x_j) and (x_j, x_k) to produce constraints on pairs of variables (x_i, x_k). The propagation occurs only in one direction following the variable order $x_1, \ldots, x_n$. Any order of variables could have been used for the same purpose. UTVPI consistency is therefore a *directional consistency* algorithm in the sense of [Dec92].

Algorithm UTVPI-CONSISTENCY
For $j = 1$ to n do
  For every i, k such that $j < i, k \leq n$ do
    $a := min(s_{ij},\ s_{ij} + d_{kj},\ d_{ij} + s_{jk})$
    $a' := max(-s'_{ij},\ -d_{ij} - s'_{jk},\ -s'_{ij} - d_{jk})$
    $b := min(d_{ij},\ s_{ij} + s'_{jk},\ d_{ij} + d_{jk})$
    $b' := max(-d_{ji},\ -s'_{ij} - s_{jk},\ -d_{ji} - d_{kj})$
    If $a' \leq a$ and $b' \leq b$ then
      Substitute the constraints on variables $x_i,\ x_k$
      of C by $a' \leq x_i + x_k \leq a$ and $b' \leq x_i - x_k \leq b$.
    Else Return "Inconsistent"
  EndFor
EndFor
Return "Consistent"

Figure 2: Deciding consistency of a set C of UTVPI constraints in variables $x_0, x_1, \ldots, x_n$.

We can also see that UTVPI-CONSISTENCY is essentially Fourier's variable elimination algorithm [Sch86] specialized for the class of UTVPI constraints. Variables are eliminated in the order $x_1, \ldots, x_n$ but constraints involving eliminated variables are not discarded (but are not used in subsequent steps too). UTVPI-CONSISTENCY can be slightly modified to give a classic variable elimination algorithm for UTVPI constraints.

The next theorem makes use of the definition of width of a constraint set [Fre82, DP88].

Definition 5.6 *Let C be a set of UTVPI constraints in variables $x_1, \ldots, x_n$. Let $w_j(C)$ denote the number of variables x_i such that $i > j$ and there exists a constraint involving x_j and x_i in C. The* width *of constraint set C along the variable ordering $x_1, \ldots, x_n$, denoted by $w(C)$, is $max_{1 \leq j \leq n} w_j(C)$.*

Theorem 5.6 *Algorithm* UTVPI-CONSISTENCY *is correct and runs in $O(nw^2)$ time where w is the width of the constraint set C along the variable ordering $x_1, \ldots, x_n$ after termination.*

Proof: The correctness follows from the correctness of Fourier's variable elimination algorithm for linear constraints. The time complexity bound can be achieved if C is represented by the equivalent graph G_C and for every variable x_l we have constant time access to all the edges used when the outer for-loop of UTVPI-CONSISTENCY reaches $j = l$. This can be implemented by using an adjacency list representation of G_C where each node has four different adjacency lists corresponding to the four possible types of UTVPI constraints on two variables. ■

Corollary 5.1 *The class UTVPI is closed under variable elimination.*

Corollary 5.2 *Let C be a set of UTVPI constraints in variables $x_1, \ldots, x_n$. We can eliminate variables $x_1, \ldots, x_m$ from C in $O(mw^2)$ time where w is the width of the constraint set C along variable ordering $x_1, \ldots, x_m, x_{m+1}, \ldots, x_n$ after termination.*

Global consistency of UTVPI constraints can also be achieved in PTIME by Algorithm UTVPI-GCONSISTENCY. This algorithm will not be given explicitly. It is obtained by the following simple modification of Algorithm UTVPI-CONSISTENCY:

> Substitute the first two lines that introduce For statements by "For $i, j, k = 0$ to n do" and remove the first of the lines with the keyword EndFor.

Theorem 5.7 *Algorithm* UTVPI-GConsistency *is correct and it runs in* $O(n^3)$ *time.*

Proof: The correctness proof is easy and follows from [JMSY94]. If we assume an adjacency matrix implementation of the graph representing C then the time complexity bound is trivially implied. ■

Having studied the properties of UTVPI constraints we now turn to an extension of this class.

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

We will extend the class of UTVPI constraints to include disequations of the form $\pm x_1 \neq c$ or $\pm x_1 \pm x_2 \neq c$. UTVPI$^{\neq}$ constraints give us more expressive power by allowing a limited form of negative information. For example, in spatial constraint databases over $\mathcal{Q}^2$ disequations can be used to *remove* points and straight lines from polygons defined by UTVPI constraints. Our developments will follow [Kou97b] where the class of DIFF$^{\neq}$ constraints has been considered. All the results of this section are original.

Definition 6.1 *A* UTVPI$^{\neq}$ constraint *is a formula* $\pm x_1 \sim c$ *or* $\pm x_1 \pm x_2 \sim c$ *where* x_1, x_2 *are variables ranging over the rational numbers,* c *is a rational constant and* $\sim$ *is* $\leq$ *or* $\neq$.

Example 6.1 *Below we give some examples of* UTVPI$^{\neq}$ *constraints:*

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

As in the case of UTVPI constraints, a set C of UTVPI$^{\neq}$ constraints can also be represented by an undirected labelled graph G_C. The only difference is that now edges can also be labelled by disequations.

Definition 6.2 *If* C *is a set of* UTVPI$^{\neq}$ *constraints then* $conv(C)$ *will denote the* convex part *of* C *i.e.,* C *minus its disequations.*

Example 6.2 *If* $C = \{\ x_1 \leq 12,\ x_1 + x_2 \leq 2,\ x_3 + x_2 \neq 6\ \}$ *then* $conv(C) = \{\ x_1 \leq 12,\ x_1 + x_2 \leq 2\ \}$.

Theorem 6.1 *Let* C *be a set of* UTVPI$^{\neq}$ *constraints. Determining the consistency of* C *can be done in* $O(n^3 + d)$ *time where* d *is the number of disequations and* n *is the number of variables in* C.

Proof: We first run UTVPI-GConsistency on $conv(C)$. Then we check each disequation $x_i + x_j \neq r$ (or $x_i - x_j \neq r$) and see if it contradicts the constraints of C. The only way there might be a contradiction is if the globally consistent set C contains the pair of constraints $r \leq x_i + x_j \leq r$ (or $r \leq x_i - x_j \leq r$). The correctness of this algorithm follows from the main result of [LM92] on the independence of disequations for systems of linear constraints. If we assume that the graph representing C is implemented by an adjacency matrix, the complexity of the above task is $O(n^3 + d)$ time. ■

This theorem extends a result of Gerevini and Cristani [GC97] who consider consistency of $DIFF^{\neq}$ constraints.

The above theorem does not hold if UTVPI$^{\neq}$ constraints over the integers are considered. In this case the consistency problem can be easily shown to be NP-complete.

6.1 Variable Elimination and Global Consistency

We now turn our attention to variable elimination for UTVPI$^{\neq}$ constraints. We show that the class UTVPI$^{\neq}$ is closed under variable elimination and that this operation can be performed in PTIME. For technical reasons it will be easier to show first that global consistency for UTVPI$^{\neq}$ constraints can be enforced in PTIME.

Algorithm UTVPI$^{\neq}$-GConsistency, shown in Figure 3, takes as input a consistent set C of UTVPI$^{\neq}$ constraints and returns an equivalent globally consistent constraint set. The algorithm works as follows. First $conv(C)$ is made globally consistent. Then the algorithm considers all disequations of C one by one. For each disequation $\pm x_i \pm x_k \neq r$ the algorithms has three steps. In Step 2.1 we consider all pairs of inequalities from $conv(C)$ that bound x_i from above and below e.g., $\pm x_m + a \leq x_i \leq \pm x_l + b$. The constraint resulting from the elimination of x_i from the constraint set

$$\pm x_m + a \leq x_i, \quad x_i \leq \pm x_l + b, \quad \pm x_i \pm x_k \neq r$$

is added to the output of the algorithm. Note that the result of this elimination step can be a *disjunction* of disequations as Example 6.3 shows.

Step 2.2 is similar to Step 2.1 but now variable x_k is considered. In Step 2.3 we consider simultaneously pairs of inequalities that bound x_i and pairs of inequalities that bound x_k e.g.,

$$\pm x_m + a \leq x_i, \quad x_i \leq \pm x_l + b, \quad \pm x_s + f \leq x_k, \quad x_k \leq \pm x_t + g.$$

The constraint resulting from the elimination of x_i and x_k from the constraint set

$$\pm x_m + a \leq x_i, \quad x_i \leq \pm x_l + b,$$

$$\pm x_s + f \leq x_k, \quad x_k \leq \pm x_t + g, \quad \pm x_i \pm x_k \neq r$$

is added to the output of the algorithm. The result of this elimination can again be a disjunction of disequations (with at most three disjuncts) as Example 6.3 shows.

Steps 2.1, 2.2 and 2.3 do all the constraint propagation that is needed for dealing with disequations. No more propagation is necessary because all possible propagations for inequalities have already been performed by Step 1, and the results are included in $conv(C)$ and are duly considered by Steps 2.1, 2.2 and 2.3 of the algorithm. Let us now show a complete example of Algorithm UTVPI$^{\neq}$-GConsistency in operation.

Example 6.3 *Let us consider the set of* UTVPI$^{\neq}$ *constraints*

$$x_3 - x_2 \leq 3, \; x_2 + x_1 \leq 3,$$

$$-x_5 - x_6 \leq -1, \; x_5 + x_4 \leq 2, \; x_2 + x_5 \neq 4$$

and apply Algorithm UTVPI$^{\neq}$-GConsistency. *Step 1 enforces global consistency on the convex part of the above set. The resulting set of UTVPI constraints is the following:*

$$x_3 - x_2 \leq 3, \; x_2 + x_1 \leq 3, \; -x_5 - x_6 \leq -1, \; x_5 + x_4 \leq 2,$$

$$x_3 + x_1 \leq 6, \; x_4 - x_6 \leq 1$$

This set is strong 3-consistent but not 4-consistent. Step 2.1 adds to this set the following disjunctions:

$$x_3 + x_5 \neq 7 \vee x_5 - x_1 \neq 1, \; x_2 - x_6 \neq 3 \vee x_2 - x_4 \neq 2$$

Algorithm UTVPI$^{\neq}$-GConsistency
Step 1: Enforce global consistency on $conv(C)$ using Algorithm UTVPI-GConsistency (see Section 5).

Step 2: Enforce global consistency.
$C_{new} := \emptyset$
<u>For</u> every disequation $\pm x_i \pm x_k \neq r$ in C <u>do</u>

  Step 2.1
  <u>For</u> every pair of inequalities $\pm x_m + a \leq x_i$ and
  $x_i \leq \pm x_l + b$ in $conv(C)$ <u>do</u>
    Let c be the disjunction of disequations resulting
    from eliminating x_i from $\pm x_m + a \leq x_i$,
    $x_i \leq \pm x_l + b$ and $\pm x_i \pm x_k \neq r$.
    $C_{new} := C_{new} \cup \{c\}$
  <u>EndFor</u>

  Step 2.2
  <u>For</u> every pair of inequalities $\pm x_s + f \leq x_k$ and
  $x_k \leq \pm x_t + g$ in $conv(C)$ <u>do</u>
    Let c be the disjunction of disequations resulting
    from eliminating x_k from $\pm x_s + f \leq x_k$,
    $x_k \leq \pm x_t + g$ and $\pm x_i \pm x_k \neq r$.
    $C_{new} := C_{new} \cup \{c\}$
  <u>EndFor</u>

  Step 2.3
  <u>For</u> every 4-tuple of inequalities $\pm x_m + a \leq x_i$,
  $x_i \leq \pm x_l + b$, $\pm x_s + f \leq x_k$ and $x_k \leq \pm x_t + g$ in
  $conv(C)$ <u>do</u>
    Let c be the disjunction of disequations resulting
    from eliminating x_i and x_k from $\pm x_m + a \leq x_i$,
    $x_i \leq \pm x_l + b$, $\pm x_s + f \leq x_k$, $x_k \leq \pm x_t + g$ and
    $\pm x_i \pm x_k \neq r$.
    $C_{new} := C_{new} \cup \{c\}$
  <u>EndFor</u>

<u>EndFor</u>
<u>Return</u> $C_{new} \cup C$

Figure 3: Global consistency for UTVPI$^{\neq}$ constraints

The resulting set is strong 4-consistent but not 5-consistent. Finally, step 3 adds the disjunction

$$x_6 - x_1 \neq 0 \vee x_4 + x_1 \neq 1 \vee x_3 + x_1 \neq 6.$$

The resulting set is strong 5-consistent and globally consistent.

Theorem 6.2 *Algorithm* UTVPI$^{\neq}$-GCONSISTENCY *is correct and it runs in* $O(dn^4)$ *time where* d *is the number of disequations and* n *is the number of variables in* C.

Proof: The complexity bound is trivial assuming that C is represented by the union of two data structures: an undirected labelled graph for inequalities, and a linked list for disequations and disjunctions of disequations.

The correctness proof is very similar to the proof for the case of DIFF$^{\neq}$ constraints that can be found in [Kou97a]. It is an easy proof but has a big number of cases.

Let C' denote $C_{new} \cup C$, i.e., the result of the algorithm UTVPI$^{\neq}$-GCONSISTENCY. Since 1-consistency holds trivially, we have to show that C' is ν-consistent for every $\nu,\ 2 \leq \nu \leq n$.

Let us take an arbitrary valuation $v = \{x_1 \leftarrow x_1^0, \ldots, x_{\nu-1} \leftarrow x_{\nu-1}^0\}$ such that $C'(x_1^0, \ldots, x_{\nu-1}^0)$ is satisfiable. We will show that for every variable x_ν, v can be extended to a valuation $v' = v \cup \{x_\nu \leftarrow x_\nu^0\}$ such that $C'(x_1^0, \ldots, x_\nu^0)$ is satisfiable.

If all constraints involving x_ν and any of $x_1, \ldots, x_{\nu-1}$ are inequalities, our result is immediate since C is globally consistent. Let us then assume that $C'(x_1, \ldots, x_\nu)$ contains disequations, and consider $C'(x_1^0, \ldots, x_{\nu-1}^0, x_\nu)$.

Let I_i be the set of natural numbers j such that $x_i \neq \pm x_j + r \vee \phi$ is a disjunction in C', and D_i denote the number of disequations in I_i. Then $C'(x_1^0, \ldots, x_{\nu-1}^0, x_\nu)$ can be written as

$$\{\pm x_\mu^0 + a_{\nu\mu} \leq x_\nu,\ x_\nu \leq \pm x_\lambda^0 + b_{\lambda\nu}\} \cup \bigcup_{\zeta \in I_\nu} \{x_\nu \neq \pm x_\zeta^0 + r_1, \ldots, x_\nu \neq \pm x_\zeta^0 + r_{D_\zeta}\}$$

where $\mu, \lambda, \zeta \in \{1, \ldots, \nu - 1\}$. Since the rational numbers are dense, there is only one case that would not allow us to find a value x_ν^0 such that $C'(x_1^0, \ldots, x_{\nu-1}^0, x_\nu^0)$ is satisfiable: when there exists $\rho \in I_\nu$ and $\eta \in \{1, \ldots, D_\rho\}$ such that

$$\pm x_\mu^0 + a_{\nu\mu} = \pm x_\lambda^0 + b_{\lambda\nu} = \pm x_\rho^0 + r_\eta \tag{1}$$

The proof now continues as in [Kou97a] depending on the step of Algorithm UTVPI$^{\neq}$-GCONSISTENCY that generated the disequation from which $x_\nu \neq \pm x_\rho^0 + r_\eta$ has been produced. For every possible case, we can show that a valuation satisfying equation (1) *cannot* arise.

The main idea behind this proof is that once propagation of inequalities has been completed (i.e., C is globally consistent) then disequations can be dealt with by the second step of the algorithm. The disjunctions generated in this way need *not* be propagated any further: they will not produce any constraint that is not subsumed by an existing one. The details are easy but tedious and are omitted. ■

Corollary 6.1 *Strong 5-consistency is necessary and sufficient for establishing global consistency of sets of* UTVPI$^{\neq}$ *constraints.*

Proof: Example 6.3 shows that 5-consistency is necessary. Also notice that Algorithm UTVPI$^{\neq}$-GCONSISTENCY asserts new constraints on at most four variables. All these constraints will be discovered when enforcing strong 5-consistency. ■

Corollary 6.2 *The class* UTVPI$^{\neq}$ *is closed under variable elimination.*

Corollary 6.3 *Let C be a set of* UTVPI$^{\neq}$ *constraints. 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.*

This section and Section 5 have provided us with many interesting results on subclasses of HDL constraints with satisfiability and variable elimination problems that can be solved in PTIME. We are now ready to combine these results with the abstract results of Section 4, to study the complexity of query answering in various interesting temporal and spatial domains.

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

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

$$Q(OpenOrClosed, Modality, FO\text{-}Formula\text{-}Type, Constraints)$$

and a database type is a tuple of the form

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

The arguments *Constraints, LocalCondition* and *ConstraintStore* vary over constraint classes. This section will consider the following classes of constraints:

- HDL, LIN and UTVPI$^{\neq}$ defined earlier.
- LINEQ. This is the subclass of LIN which contains only linear equalities.
- IA. This is the Interval Algebra of Allen [All83] which contains the following 13 *basic* relations:

  $$before,\ after,\ during,\ meets,\ starts,\ finishes,\ equals$$

  and their inverses. These 13 relations can express every possible relation between two time intervals. The full algebra consists of 2^{13} relations that are expressed as disjunctions of the 13 basic relations. Disjunctions of basic relations of IA are usually written in set notation e.g., $i\ \{before,\ after\}\ j$ is another way of writing $i\ before\ j\ \ \vee\ i\ after\ j$. Disjunctions of basic relations are used to express *indefinite* temporal information.
- 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 [vBC90]. As an example the relation $\{before,\ meets\}$ of IA is in SIA while the relation $\{before,\ after\}$ is not.
- ORD-Horn. 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 [NB95]. An *ORD-Horn constraint* is a disjunction of weak inequalities of the form $x \leq y$ and disequations of the form $x \neq y$ such that the number of inequalities should not exceed one [NB95]. Obviously ORD-Horn constraints are a subclass of HDL constraints. Also notice that $SIA \subset ORD\text{-}Horn$. As an example the relation

  $$\{before,\ after,\ during,\ meets,\ starts,\ finishes\}$$

  is not in SIA but is in ORD-Horn. The relation $\{before,\ after\}$ is not in either SIA or ORD-Horn.

- 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 $\mathcal{Q}^2$ [Gue89, PTSE95, BCdC98]. Every rectangle r can be defined by a 4-tuple $(L^r_x, L^r_y, U^r_x, U^r_y)$ 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 $\mathcal{Q}^2$.

  2d-Ord-Horn is the subclass 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^r_x and U^r_x.
  - ψ is a conjunction of ORD-Horn constraints on variables L^r_y and U^r_y.

- PA. This is the Point Algebra of [VK86] which

  possible relations between two time points. The full algebra consists of 2^3 relations: $<, >, =, \leq, \geq, \neq, ?$ and $\emptyset$ (? is the "unknown" relation and $\emptyset$ is the inconsistent one).

- CPA. This is the subalgebra of PA which does not include the relation $\neq$ [VKvB89].

- SORD. This is the subalgebra of PA which contains only the relations $\{<, >\}$. In other words, SORD is the class of *strict order* constraints.

- WORD. This is the subalgebra of PA which contains only the relations $\{\leq, \geq\}$. In other words, WORD is the class of *weak order* constraints.

- ORD-CON. This is the subclass of LIN which contains only constraints of the form $x \sim r$ where x is a variable, r is a *rational constant* and $\sim$ is $<$, $>$, $\leq$, or $\geq$.

- UTVPI-EQ. This is the subclass of UTVPI which contains only equality constraints.

- RAT-EQUAL. This is the subclass of LINEQ which contains only equality constraints of the form $x = v$ where x is a variable and v is a *variable or a rational constant* (ordinary or Skolem).

- RAT-EQUAL-CON. This is the subclass of RAT-EQUAL which contains only equality constraints of the form $x = a$ where x is a variable and a is a *rational constant* (ordinary or Skolem).

  Among other things, this class is useful for specifying databases of type

  $$DB(A, \mathit{RAT\text{-}EQUAL\text{-}CON}, \mathcal{C})$$

  where A is an arity and $\mathcal{C}$ is a constraint class. In databases of this type, predicates are defined by completions (in the sense of [Rei84]) of formulas of the form $p(\overline{a}, \overline{\omega})$ where $\overline{a}$ is a vector of rational constants and $\overline{\omega}$ is a vector of Skolem constants. For example, the database

  $$\begin{array}{c}(\; \{ \; (\forall t_1, t_2, t_3/\mathcal{Q})((t_1 = \omega_1 \; \wedge \; t_2 = \omega_2 \; \wedge \; t_3 = 1) \equiv p(t_1, t_2, t_3)) \; \}, \\ \{ \; \omega_1 < \omega_2 \; \} \;)\end{array}$$

is of type

$$DB(3\text{-}ary, RAT\text{-}EQUAL\text{-}CON, SORD).$$

These databases are typical of the kind of databases encountered in temporal and spatial problems involving indefinite information (where information about non-temporal entities like *Mary* and *Fred* of Example 3.2 has been abstracted away).

- EQUAL. This is the subclass of $\mathcal{EQ}$-constraints of the form $x = v$ where x is a variable and v is a variable or a constant (ordinary or Skolem). The constants are assumed to be from a countably infinite domain (see the definition of language $\mathcal{EQ}$ in Section 3.1).

  This class (and the class HDE defined below) is not useful for temporal or spatial reasoning. However it can be used to represent indefinite information it arises in traditional relational databases. Thus EQUAL and HDE are used only in Section 7.2 for comparing our results to some similar results for relational databases with indefinite information [Gra91, AKG91].

- HDE. This is the class of Horn-disjunctive equality constraints originally defined by Grahne [Gra91] who called them Horn conditions. A *Horn-disjunctive equality constraint* is a disjunction $d_1 \vee \cdots \vee d_n$ $(n \geq 1)$ such that all the d_i's are $\mathcal{EQ}$-constraints (see Section 3.1) but only at most one of the d_i's is an equality.

- NONE. This is the class which contains only the trivial constraints *true* and *false*. We have already used this class in Section 4 to specify queries with database predicates but no constraints. This class is also useful for specifying databases of the form

  $$(EventsAndFacts(DB),\ ConstraintStore(DB))$$

  where $ConstraintStore(DB) = \emptyset$ (i.e., there might be Skolem constants but we know nothing about them).

Now that we have introduced the constraints classes that we will consider, we are ready to present our results.

7.1 PTIME Problems

The following theorem is the culmination of the complexity analysis in this paper.

Theorem 7.1 *The evaluation of*

(a) $Q(Closed, \Diamond, PositiveExistential, HDL)$ *queries over* $DB(N\text{-}ary, HDL, HDL)$ *databases,*

(b) $Q(Closed, \Box, PositiveExistential, LINEQ)$ *queries over* $DB(N\text{-}ary, LINEQ, HDL)$ *databases,*

(c) $Q(Open, \Diamond, PositiveExistential, \mathrm{UTVPI}^{\neq})$ *queries over* $DB(N\text{-}ary, \mathrm{UTVPI}^{\neq}, \mathrm{UTVPI}^{\neq})$ *databases and*

(d) $Q(Open, \Box, SinglePredicate, NONE)$ *queries over* $DB(N\text{-}ary, \mathrm{UTVPI\text{-}EQ} \cup \mathrm{UTVPI}^{\neq}_{\leq 1}, \mathrm{UTVPI}^{\neq})$ *databases*

can be performed in PTIME.

Proof: Part (a) follows from Theorem 4.1 because HDL is closed under variable elimination and *SAT(HDL)* and *FVAR-ELIM(HDL)* can be done in PTIME.

Part (b) follows from Theorem 4.2 because LINEQ is closed under variable elimination,

$$LINEQ \subset HDL, \ \vee\overline{LINEQ} \subset HDL$$

and *SAT(HDL)* and *FVAR-ELIM(HDL)* can be done in PTIME.

Part (c) follows from Theorem 4.3 because the class UTVPI$^{\neq}$ is closed under variable elimination and *VAR-ELIM(*UTVPI$^{\neq}$*)* can be done in PTIME.

For Part (d) let us first notice that UTVPI$^{\neq}$ is closed under negation. The results of Section 6 show that constraint sets of type UTVPI-EQ $\cup$ UTVPI$^{\neq}{}_{\leq 1}$ are closed under variable elimination, and

$$FVAR\text{-}ELIM(\text{UTVPI-EQ} \cup \text{UTVPI}^{\neq}{}_{\leq 1})$$

can be done in PTIME. In general

$$VAR\text{-}ELIM(\text{UTVPI-EQ} \vee \text{UTVPI}^{\neq}{}_{\leq 1})$$

cannot be done in PTIME (Theorem 4.4 of [Kou92]). However if we examine carefully the proof of Lemma 4.4, we can see that in this case the number of variables in each disjunctive constraint is *fixed* (due to the data complexity assumption). Thus we can use the techniques of Theorem 6.2 of this paper, and Corollary 6.2 of [Kou97a] to see that this elimination can be achieved in PTIME. Therefore Part (d) follows from Theorem 4.4. ■

Theorem 7.1(a) is a significant extension of the PTIME result of Brusoni, Console and Terenziani on closed conjunctive possibility queries over conjunctions of DIFF constraints (Section 5 of [BCT95]). We have an extension of the result of [BCT95] because we have more expressive languages for defining databases and expressing queries: [BCT95] does not allow database predicates and the constraint store consists only of DIFF constraints.

Let us now consider databases and queries involving higher-order objects i.e., intervals and rectangles.

Theorem 7.2 *The evaluation of*

(a) $Q(Closed, \Diamond, PositiveExistential, ORD\text{-}Horn)$ *queries over* $DB(N\text{-}ary, ORD\text{-}Horn, ORD\text{-}Horn)$ *databases,*

(b) $Q(Closed, \Diamond, PositiveExistential, 2d\text{-}ORD\text{-}Horn)$ *queries over* $DB(N\text{-}ary, 2d\text{-}ORD\text{-}Horn, 2d\text{-}ORD\text{-}Horn)$ *databases,*

(c) $Q(Open, \Diamond, PositiveExistential, SIA)$ *queries over* $DB(N\text{-}ary, SIA, SIA)$ *databases*

can be performed in PTIME.

Proof: It is not difficult to show that all the classes of constraints involved are closed under variable elimination. Then Part (a) follows from Theorem 4.1 because *SAT(ORD-Horn)* and *FVAR-ELIM(ORD-Horn)* are in PTIME (the first result is from [NB95] and the second one from [Kou99]). Part (b) follows from Theorem 4.1 because *SAT(2d-ORD-Horn)* and *FVAR-ELIM(2d-ORD-Horn)* are in PTIME (these results follow easily from [NB95] and [Kou99]). Finally, Part (c) follows from Theorem 4.3 because *VAR-ELIM(SIA)* can be done in PTIME (the tractability of *VAR-ELIM(SIA)* is stated explicitly in [Kou99] but follows easily from [Kou97a]). ■

Theorem 7.2(a) is a significant extension of the PTIME result of vanBeek on closed conjunctive possibility queries over conjunctions of SIA constraints (Theorem

1 of [vB91]). We have an extension of the result of [vB91] because we have more expressive languages for defining databases and expressing queries: [vB91] does not allow database predicates and the constraint store consists of SIA constraints.[7]

Theorem 7.2(b) is an interesting result for rectangle databases with indefinite information over $\mathcal{Q}^2$. This result can be generalized to $\mathcal{Q}^n$ if one defines an appropriate algebra nd-ORD-Horn.

Theorem 7.2(c) is an interesting result on querying databases with indefinite temporal information expressed by SIA constraints and complements the results in [vB91].

Finally, let us notice that we cannot use Theorems 4.2 and 4.4 to derive any results involving subclasses of IA or 2d-IA because the closure properties assumed by these theorems are not true for any interesting subclasses of IA or 2d-IA.

7.2 Related Results from Relational Databases with Indefinite Information

We now demonstrate that the theorems of Section 4.3 are not simply useful for the temporal and spatial problems we are interested in, but have wider applicability. For example, we can use them to derive some known database results from the theory of relational databases with indefinite information.

Grahne [Gra91] has considered the data complexity of

$$Q(Closed, \diamond, PositiveExistential, HDE)$$

queries over $DB(N\text{-}ary, HDE, HDE)$ databases. Given that HDE is closed under variable elimination, and *SAT(HDE)* and *FVAR-ELIM(HDE)* can be done in PTIME[8], Theorem 5.5 of [Gra91] is a straightforward consequence of Theorem 4.1.

Grahne [Gra91] has also considered the data complexity of

$$Q(Closed, \Box, PositiveExistential, EQUAL)$$

queries over $DB(N\text{-}ary, EQUAL, HDE)$ databases.[9] Because $EQUAL$ is closed under variable elimination, $EQUAL \subset HDE$, $\vee\overline{EQUAL} \subset HDE$, and *SAT(HDE)* and *FVAR-ELIM(HDE)* can be done in PTIME, Theorem 5.12 of [Gra91] is a consequence of Theorem 4.2.

7.3 Lower Bounds

Let us now consider identifying the precise boundary between tractable and intractable query answering problems for indefinite constraint databases with linear constraints. We start our inquiry by considering whether the results of Theorem 7.1 can be extended to more expressive classes of queries.[10] For example, can we allow negation in the queries (equivalently, can we allow arbitrary first order formulas)

[7] In addition to possibility queries, [vB91, BCT95] deal also with certainty queries and discuss cases where query evaluation can be done in PTIME (Theorem 2 of [vB91] and Section 4 of [BCT95]). These results can be easily extended to the more expressive classes ORD-Horn ([vB91] considers SIA constraints) and HDL ([BCT95] considers DIFF constraints). In our framework this can be done by restricting the concepts of database and query so that they match the corresponding concepts of [vB91, BCT95], and then employing proof techniques similar to the ones in [vB91, BCT95]. The details follow easily from [vB91, BCT95] and are omitted.

[8] The result about satisfiability is from [Gra91]. The results about variable elimination are straightforward.

[9] Theorem 5.12 of [Gra91] actually gives a PTIME result for Datalog queries over Horn tables. In our framework the class of Horn tables is equivalent to the class of $DB(N\text{-}ary, EQUAL, HDE)$ databases.

[10] Similar issues arise for Theorem 7.2. The results of this section can easily be generalised to this case.

and still get results like Theorem 7.1(a) or 7.1(b)? The following theorem shows that the anwer to this question is negative.[11]

Theorem 7.3 ([AKG91]) *Let DBC be the set of databases of type*

$$DB(4\text{-}ary, RAT\text{-}EQUAL\text{-}CON, NONE)$$

with the additional restriction that every Skolem constant occurs at most once in any member of DBC. Then:

1. *There exists a query $q \in Q(Closed, \Diamond, FirstOrder, RAT\text{-}EQUAL)$ such that deciding whether $q(db) = yes$ is NP-complete even when db ranges over databases in the set DBC.*
2. *There exists a query $q \in Q(Closed, \Box, FirstOrder, RAT\text{-}EQUAL)$ such that deciding whether $q(db) = yes$ is co-NP-complete even when db ranges over databases in the set DBC.*

Theorem 7.1(a) and (b) together with the above theorem establish a clear separation between tractable and possibly intractable query answering problems. The presence of negation in the query language can easily lead us to computationally hard query evaluation problems (NP-complete or co-NP-complete) even with very simple input databases.

Another issue that we would like to consider is whether one can improve Theorem 7.1(b) with a class which is more expressive than LINEQ (for example LIN). The following result shows that this is not possible; even the presence of strict order constraints in the query is enough to lead us away from PTIME.

Theorem 7.4 ([vdM92]) *There exists a query in $Q(Closed, \Box, Conjunctive, SORD)$ with co-NP-hard data complexity over $DB(Binary, RAT\text{-}EQUAL\text{-}CON, SORD)$ databases.*

Note that for the above theorem to be true, SORD constraints must be present *both in the database and in the query.* Otherwise, as Theorems 7.5 and 7.6 imply, conjunctive query evaluation can be done in PTIME.

Theorem 7.5 *Evaluating $Q(Closed, \Box, PositiveExistential, NONE)$ queries over $DB(N\text{-}ary, RAT\text{-}EQUAL\text{-}CON, HDL)$ databases can be done in PTIME.*

Proof: Follows immediately from Theorem 7.1(b). ■

Theorem 7.6 *Evaluating $Q(Closed, \Box, Conjunctive, LIN)$ queries over*

$$DB(N\text{-}ary, RAT\text{-}EQUAL\text{-}CON, NONE)$$

databases can be done in PTIME.

Proof: A query in the class $Q(Closed, \Box, Conjunctive, LIN)$ is of the form

$$: \Box(\exists \overline{x})(\phi(\overline{x}) \ \wedge \ \psi(\overline{x}))$$

where $\phi(x)$ is a conjunction of atomic formulas with database predicates, and $\psi(x)$ is a conjunction of linear constraints. Without loss of generality, we can also assume that all arguments of predicates of ϕ are distinct variables.

A database in the class $DB(N\text{-}ary, RAT\text{-}EQUAL\text{-}CON, NONE)$ consists of completions (in the sense of [Rei84]) of atomic formulas of the form $p(\overline{r}, \overline{\omega})$ where $\overline{r}$

[11] The theorem has been proved in [AKG91] for equality constraints over any countably infinite domain thus it holds for the domain of rational numbers too.

is a vector of rational constants, $\overline{\omega}$ is a vector of Skolem constants, and all Skolem constants in the database are distinct (this is necessary because if two of them are equal then this is equivalent to having an equality constraint in the constraint store).

In this case, according to Theorem 3.1 query evaluation amounts to quantifier elimination from the formula

$$(\forall \overline{\omega})((\exists \overline{x})((\theta_1(\overline{x}, \overline{r}, \overline{\omega}) \ \vee \ \cdots \ \vee \ \theta_n(\overline{x}, \overline{r}, \overline{\omega})) \ \wedge \ \psi(x)))$$

or equivalently

$$(\forall \overline{\omega})((\exists \overline{x})((\theta_1(\overline{x}, \overline{r}, \overline{\omega}) \ \wedge \ \psi(x)) \ \vee \ \cdots \ \vee \ (\theta_n(\overline{x}, \overline{r}, \overline{\omega}) \ \wedge \ \psi(x)))).$$

In the above formulas the θ_i's are conjunctions of equalities of the form $x = a$ where x is a variable and a is a rational constant or a variable corresponding to a Skolem constant. In addition, each θ_i involves distinct variables from the vector $\overline{\omega}$.

If we eliminate quantifiers $(\exists \overline{x})$ then the resulting formula is

$$(\forall \overline{\omega})(\xi_1(\overline{r}, \overline{\omega}) \ \vee \ \cdots \ \vee \ \xi_n(\overline{r}, \overline{\omega}))$$

where the ξ_i's are conjunctions of linear constraints. Because each ξ_i involves distinct variables from vector $\overline{\omega}$, it is easy to decide the validity of the above sentence. If one of the ξ_i's is true for all values of its variables then the sentence is valid, and the answer to the query is "yes". Otherwise the sentence is not valid and the answer to the query is "no".

The query evaluation algorithm sketched above is obviously in PTIME. ■

A final issue that the careful reader might be wondering about is whether Parts (c) and (d) of Theorem 7.1 can be extended. Let us consider Part (c) first. Theorem 7.3 shows that we should not expect to stay within PTIME if we move away from positive existential queries. So the only way that this result could be improved is by discovering a class $\mathcal{C}$ such that UTVPI$^{\neq} \subset \mathcal{C} \subset HDL$ and $VAR\text{-}ELIM(\mathcal{C})$ is in PTIME. This is therefore an interesting open problem; its solution will also be very interesting to linear programming researchers [HN94, Gol97].

Let us now consider whether we can improve Theorem 7.1(d). The following result shows that this is *not* possible by extending the class of constraints allowed in the definitions of the database predicates so that *more than one* non UTVPI-EQ constraints are allowed in each conjunction.[12]

Theorem 7.7 *There exists a query in $Q(Closed, \Box, SinglePredicate, NONE)$ with co-NP-hard data complexity over*

$$DB(Monadic, RAT\text{-}EQUAL\text{-}CON \cup WORD_{\leq 2}, SORD)$$

databases.

Proof: We will use a reduction from the following co-NP-hard problem of Theorem 3 in [GS94] (let us call it EXCL):

> **Instance:** A set of point variables $p_1, \ldots, p_n$, a set of interval variables $i_1, \ldots, i_m$ and a set of constraints C of the form "point p_k is not included in interval i_l" (where $1 \leq k \leq n$, $1 \leq l \leq m$). Point and interval variables are interpreted over the rational numbers and rational intervals respectively.
>
> **Question:** Is C inconsistent?

[12] Since our result is negative, it is enough to consider closed queries.

Given an instance of EXCL, we construct a database in

$$DB(Monadic, RAT\text{-}EQUAL\text{-}CON \cup WORD_{\leq 2}, SORD)$$

as follows:

- For each point variable p_k, we introduce a Skolem constant λ_k.
- For each interval variable i_l, we introduce two Skolem constants ω_l^- and ω_l^+ and a SORD constraint $\omega_l^- < \omega_l^+$ in the constraint store.
- For each constraint $c \in C$ of the form "point p_k is not included in interval i_l", we introduce a local condition θ_c of the form $\lambda_k \geq \omega_l^- \ \wedge \ \lambda_k \leq \omega_l^+$.
- There is only one predicate p defined by

$$(\forall x)(p(x) \equiv \bigvee_{c \in C} (x = 1 \ \wedge \ \theta_c)).$$

The query $q \in Q(Closed, \Box, SinglePredicate, NONE)$ that we will use is

$$: \Box p(1).$$

This query is fixed (it does not depend on the instance of EXCL since our measure is data complexity).

The above reduction is obviously polynomial. Also q has answer "yes" iff the formula

$$(\forall \lambda_1, \ldots, \lambda_n, \omega_1^-, \omega_1^+, \ldots, \omega_m^-, \omega_m^+)(\bigwedge_{l=1}^{m} \omega_l^- < \omega_l^+ \ \supset \ \bigvee_{c \in C} (\exists x)(x = 1 \ \wedge \ \theta_c))$$

of LC is true. Equivalently, after eliminating $\exists x$

$$\neg(\exists \lambda_1, \ldots, \lambda_n, \omega_1^-, \omega_1^+, \ldots, \omega_m^-, \omega_m^+)(\bigwedge_{l=1}^{m} \omega_l^- < \omega_l^+ \ \wedge \ \bigwedge_{c \in C} \neg\theta_c)$$

is true. Now observe that $\neg\theta_c$ is of the form $\lambda_k < \omega_l^- \ \vee \ \lambda_k > \omega_l^+$ i.e., it means that l_k is excluded from the rational interval $[\omega_l^-, \omega_l^+]$. So, C is inconsistent iff q has answer "yes". The proof is now complete. ■

The following theorem complements the previous one by showing that the query answering problem considered in Theorem 7.1 (d) becomes co-NP-hard if we slightly extend the class of queries considered (more precisely, if we consider conjunctive queries with *two* conjuncts that are database predicates and no constraints).

Theorem 7.8 *There exists a query q in $Q(Closed, \Box, Conjunctive, NONE)$ with co-NP-hard data complexity over databases in the class*

$$DB(Monadic, RAT\text{-}EQUAL\text{-}CON \cup WORD_{\leq 1}, SORD).$$

The query q has exactly two conjuncts that are database predicates.

Proof: We will use a reduction from the co-NP-hard problem EXCL [GS94] as we did in Theorem 7.7. Without loss of generality we assume that the constraints of EXCL are ordered (and their order is given by the numbers $1, 2, \ldots, n$).

Given an instance of EXCL, we construct a database in the class

$$DB(Monadic, RAT\text{-}EQUAL\text{-}CON \cup WORD_{\leq 1}, SORD)$$

as follows:

- There are two monadic predicates $left(x)$ and $right(x)$. The formulas defining these predicates will be built incrementally below.
- For each point variable p_k, we introduce a Skolem constant λ_k.
- For each interval variable i_f, we introduce two Skolem constants ω_f^- and ω_f^+ and a SORD constraint $\omega_f^- < \omega_f^+$ in the constraint store.
- For each constraint $c \in C$ of the form "point p_k is not included in interval i_m", we include a conjunction

  $$x = r \ \wedge \ \lambda_k \geq \omega_f^-$$

  in the definition of predicate $left(x)$ (where r is the natural number giving us the order of constraint c).

  Similarly we include a conjunction

  $$x = r \ \wedge \ \lambda_k \leq \omega_f^+$$

  in the definition of predicate $right(x)$.

The query $q \in Q(Closed, \Box, Conjunctive, NONE)$ that we will use is

$$: \Box(\exists x)(left(x) \ \wedge \ right(x)).$$

This query is fixed (it does not depend on the instance of EXCL since our measure is data complexity).

The above reduction is obviously polynomial. As in Theorem 7.7 we can also easily see that C is inconsistent iff q has answer "yes". ■

We can now conclude that it is unlikely that Theorem 7.1(d) can be improved except with the discovery of a class of constraints $\mathcal{C}$ such that UTVPI$^{\neq} \subset \mathcal{C} \subset HDL$ and $VAR\text{-}ELIM(\mathcal{C})$ is in PTIME (this is similar to what we concluded for Theorem 7.1(c)).

Let us close this section by summarising what we have achieved. The main tractability result of this section is Theorem 7.1. The rest of this section has focused on establishing that this theorem outlines very precisely the frontier between tractable and intractable query processing problems in indefinite constraint databases with Horn disjunctive linear constraints. The two cases left open by our results can only be resolved after answering an important open question in the area of linear programming (i.e., whether there exists a class of constraints $\mathcal{C}$ such that UTVPI$^{\neq} \subset \mathcal{C} \subset HDL$ and $VAR\text{-}ELIM(\mathcal{C})$ is in PTIME [HN94, Gol97]).

8 Future Work

This paper presented the scheme of indefinite constraint databases and concentrated on the problem of tractable query evaluation in this scheme. We discovered several new tractable query evaluation problems by utilising old and new results about satisfiability and variable elimination for subclasses of HDL constraints. We also outlined carefully the frontier between tractable and possibly intractable query evaluation problems.

There are still several open questions regarding query answering in databases with indefinite information expressed by linear constraints. Currently, our research concentrates on the following:

- *More expressive classes of queries and databases.* We would like to consider a full-fledged modal query language and study the problem of query evaluation. It would also be interesting to consider more expressive classes of databases e.g., in the spirit of [DB88] who consider tractable temporal reasoning in a framework similar to ours. A detailed comparison of our work with [DB88] is not very easy at this stage because the formalism of [DB88] allows several kinds of deductive rules that cannot be encoded in our model.

- *Different complexity measures.* We have not dealt with *expression complexity* and *combined complexity* at all in this paper. Query anwering problems like the ones considered in Theorem 7.1 have been shown to be very hard in the combined complexity measure even if we only have order constraints [vdM92]. Meyden [vdM92] has introduced several additional restrictions on queries and databases to arrive at query answering problems with PTIME combined complexity (in the case of order constraints only). A similar detailed study of expression/combined complexity of query answering in our framework is an interesting topic for further research.

- *Complexity of integrity constraint checking in indefinite constraint databases.* This is a very interesting topic and there are very few nice papers in the area [Ple93, CP00].

- *Richer classes of spatial constraints.* The only spatial constraints we have considered in this paper are subclasses of Horn disjunctive linear constraints. Admittedly, the expressive power of these constraints is not enough for many spatial reasoning problems and one is often forced to turn to other interesting classes of spatial constraints e.g., topological [RN99], directional [Her94, Lig98] and their combinations. It would be very interesting to consider the introduction of these classes into the scheme of indefinite constraint databases. This will be a straightforward task once properties like the ones assumed in Section 4 are established for these classes of constraints (hence the usefullness of the results in Section 4).

Acknowledgements

This research has been partially supported by project CHOROCHRONOS funded by EU's 4th Framework Programme and by a grant from the Greek General Secretariat for Research and Technology. We thank Timos Sellis for his constant encouragement and support, and the reviewers for comments that improved the presentation of this paper.

References

[AKG91] 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.

[All83] J.F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[BCdC98] P. Balbiani, J.-F. Condotta, and L.F. del Cerro. Bidimensional Temporal Relations. In *Proceedings of KR'98*, June 1998.

[BCPT94] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. LaTeR: a general purpose manager of temporal information. In *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, volume 869 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[BCPT95] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. Extending temporal relational databases to deal with imprecise and qualitative temporal information. 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. Springer, 1995.

[BCPT97] V. Brusoni, L. Console, B. Pernici, and P. Terenziani. Later: an efficient, general purpose manager of temporal information. *IEEE Expert*, 12(4):56–64, August 1997.

[BCT95] V. Brusoni, L. Console, and P. Terenziani. On the computational complexity of querying bounds on differences constraints. *Artificial Intelligence*, 74(2):367–379, 1995.

[CJK97] D. Cohen, P. Jeavons, and M. Koubarakis. Tractable disjunctive constraints. In *Proceedings of CP'97 (LNCS, Vol. 1330)*, pages 478–490, Linz, Austria, 1997.

[CP00] W. Cowley and D. Plexousakis. Temporal Integrity Constraints with Indeterminacy. In *Proceedings of VLDB 2000*, Cairo, Egypt, September 2000.

[DB88] T. Dean and M. Boddy. Reasoning About Partially Ordered Events. *Artificial Intelligence*, 36:375–399, 1988.

[Dec92] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–107, 1992.

[DM87] T. Dean and D.V. McDermott. Temporal Data Base Management. *Artificial Intelligence*, 32:1–55, 1987.

[DMP89] 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.

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. Special Volume on Knowledge Representation.

[DP88] R. Dechter and J. Pearl. Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[Fre82] E. Freuder. A Sufficient Condition For Backtrack-Free Search. *Journal of ACM*, 29(1):24–32, 1982.

[GC95] A. Gerevini and M. Cristani. Reasoning with Inequations in Temporal Constraint Networks. Technical report, IRST - Instituto per la Ricerca Scientifica e Tecnologica, Povo TN, Italy, 1995. A shorter version appears in the Proceedings of the Workshop on Spatial and Temporal Reasoning, IJCAI-95.

[GC97] A. Gerevini and M. Cristani. On Finding a Solution in Temporal Constraint Satisfaction Problems. In *Proceedings of IJCAI-97*, pages 1460–1465, 1997.

[GK97] D.Q. Goldin and P. Kanellakis. Constraint Query Algebras. *Constraints*, 1(1):45–83, 1997.

[Gol97] D.Q. Goldin. *Constraint Query Algebras*. PhD thesis, Dept. of Computer Science, Brown University, 1997.

[Gra91] G. Grahne. *The Problem of Incomplete Information in Relational Databases*, volume 554 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.

[GS94] A. Gerevini and L. Schubert. On Point-Based Temporal Disjointness. *Artificial Intelligence*, 70:347–361, 1994.

[GS95] A. Gerevini and L. Schubert. Efficient Algorithms for Qualitative Reasoning about Time. *Artificial Intelligence*, 74:207–248, 1995.

[Gue89] H.-W. Guesgen. Spatial reasoning based on Allen's temporal logic. Technical Report TR-89-094, ICSI, 1989.

[Güt94] R.H. Güting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4), 1994. Special Issue on Spatial Databases.

[Her94] Daniel Hernandez. *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Inteligence*. Springer, 1994.

[HN94] D.S. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6):1179–1192, 1994.

[HS97] 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.

[IL84] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984.

[JB96] P. Jonsson and C. Bäckström. A linear programming approach to temporal reasoning. In *Proceedings of AAAI-96*, pages 1235–1240. AAAI Press/MIT Press, 1996.

[JB98] P. Jonsson and C. Bäckström. A unifying approach to temporal constraint reasoning. *Artificial Intelligence*, 102:143–155, 1998.

[JMSY94] 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.

[Kac79] L.G. Kachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[KKR90] 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.

[KKR95] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[Kou92] M. Koubarakis. Dense Time and Temporal Constraints with $\neq$. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 24–35. Morgan Kaufmann, San Mateo, CA, October 1992.

[Kou95] 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.

[Kou96] M. Koubarakis. Tractable Disjunctions of Linear Constraints. In *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming (CP'96)*, pages 297–307, Boston, MA, August 1996.

[Kou97a] 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.

[Kou97b] 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.

[Kou99] M. Koubarakis. Tractable Disjunctions of Linear Constraints: Basic Results and Applications to Temporal Reasoning, 1999. Long version of [Kou96]. Submitted for publication.

[Lev84] H.J. Levesque. Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, 23:155–212, 1984.

[Lig98] G. Ligozat. Reasoning About Cardinal Directions. *Journal of Visual Languages and Computing*, 9, 1998.

[Lip79] Witold Jr. Lipski. On Semantic Issues Connected with Incomplete Information Databases. *ACM Transcactions on Database Systems*, 4(3):262–296, September 1979.

[LM92] Jean-Louis Lassez and Ken McAloon. A Canonical Form for Generalized Linear Constraints. *Journal of Symbolic Computation*, 13(1):1–24, 1992.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: A Language for Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[NB95] 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.

[Ple93] D. Plexousakis. Integrity Constraint and Rule Maintenance in Temporal Deductive Knowledge Bases. In *Proceedings of VLDB'93*, pages 146–157, 1993.

[PTSE95] 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.

[Rei84] R. Reiter. Towards a logical reconstruction of relational database theory. In Michael Brodie, John Mylopoulos, and Joachim Schmidt, editors, *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, pages 191–233. Springer Verlag, 1984.

[Rei88] R. Reiter. On Integrity Constraints. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 97–111, Asilomar, CA, 1988.

[RN99] J. Renz and B. Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *Artificial Intelligence*, 1-2:95–149, 1999.

[SBC92] R. Schrag, M. Boddy, and J. Carciofini. Managing Disjunction for Practical Temporal Reasoning. In *Proceedings of KR'92*, pages 36–46, October 1992.

[Sch86] A. Schrijver, editor. *Theory of Integer and Linear Programming.* Wiley, 1986.

[Sho81] Robert Shostak. Deciding Linear Inequalities by Computing Loop Residues. *Journal of the ACM*, 28(4):769–779, 1981.

[Sno95] R.T. Snodgrass, editor. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[Stu97] P. Stuckey. Constraint search trees. In *Proceedings of ICLP'97*, pages 301–315, 1997.

[Var82] M. Vardi. The Complexity of Relational Query Languages. In *Proceedings of ACM SIGACT/SIGMOD Symposium on Principles of Database Systems*, pages 137–146, 1982.

[vB91] Peter van Beek. Temporal Query Processing with Indefinite Information. *Artificial Intelligence in Medicine*, 3:325–339, 1991.

[vB92] Peter van Beek. Reasoning About Qualitative Temporal Information. *Artificial Intelligence*, 58:297–326, 1992.

[vBC90] Peter van Beek and Robin Cohen. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence*, 6:132–144, 1990.

[vdM92] 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.

[VK86] Marc Vilain and Henry Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of AAAI-86*, pages 377–382, 1986.

[VKvB89] 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.

[Yan88] M. Yannakakis. Expressing Combinatorial Optimization Problems by Linear Programs. In *Proc. of ACM Symposium on the Theory of Computing*, pages 223–288, 1988.