Querying Indefinite Temporal and Spatial Information: A New Frontier

Manolis Koubarakis
Dept. of Informatics, University of Athens
Panepistimioupolis, TYPA Buildings
157 81 Athens, Greece
manolis@di.uoa.gr, www.di.uoa.gr/~manolis

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

Abstract

Temporal and spatial constraint networks do not live alone in the wilderness. In many cases they are components of larger systems e.g., temporal database systems, spatial database systems, knowledge representation systems, natural language systems, planning systems, scheduling systems, multimedia systems and so on. We believe that an interesting new frontier for temporal and spatial reasoning research is the formalisation, analysis and possible re-implementation of systems where temporal or spatial reasoners are an important component. In this paper we will make a first contribution to this exciting area of research. We will consider temporal constraint networks complemented by a database for storing the information typically used to label network nodes. We will then study the computational complexity of querying the combined system using a first order modal query language.

Introduction

Temporal and spatial constraint networks do not live alone in the wilderness. In many cases they are components of larger systems e.g., temporal database systems (Koubarakis 1997b; Brusoni *et al.* 1995; Dean & McDermott 1987; Schrag, Boddy, & Carciofini 1992), spatial database systems (Egenhofer 1994; Papadias *et al.* 1995), knowledge representation systems (Mylopoulos *et al.* 1990), natural language systems (Miller & Schubert 1988), planning systems (Penberthy & Weld 1994), scheduling systems (Cheng & Smith 1994), multimedia systems (Subrahmanian 1998) and so on.

We believe that temporal and spatial reasoning researchers now know enough about the intricacies of various classes of temporal and spatial constraints.[1] Therefore the time has come to move to a more interesting (and possibly more difficult) problem: *formalising, analysing and possibly re-implementing larger systems where temporal or spatial reasoners are an important component.* We do not claim to be the first to have raised this issue. It is explicit in (van Beek 1991), in most papers on the TMM system[2] (Dean & McDermott 1987; Schrag, Boddy, & Carciofini 1992) and possibly in other papers.

In this paper we will make a first contribution to this exciting area of research.[3] We will consider temporal constraint networks complemented by a *database* for storing the information typically used to label network nodes. We will then study the computational complexity of querying the combined system using a *first order modal query language.* This is similar to what is done in the TMM system (Dean & McDermott 1987; Schrag, Boddy, & Carciofini 1992) and the temporal relational database models of (Koubarakis 1997b; Brusoni *et al.* 1995). Of these two database proposals the most expressive one is the scheme of *indefinite constraint databases* proposed in (Koubarakis 1997b). In this paper we redefine the scheme of (Koubarakis 1997b) (using first order logic instead of relational database theory) and take it as the formalism in which we present our contributions.

We first point out that query evaluation in the presence of indefinite temporal information is a hard problem (it is NP-hard for possibility queries and co-NP-hard for certainty queries). Motivated by this negative fact, we try to discover tractable subclasses of the general query answering problem. To achieve this, we adopt the following approach. We start with the assumption that we have a class of constraints $\mathcal{C}$ with satisfiability and variable elimination problems that can be solved in PTIME.[4] Under this assumption, we demonstrate several general classes of indefinite constraint databases and queries for which query evaluation can be done with PTIME data complexity. Then we restate these results with $\mathcal{C}$ ranging over some interesting classes of temporal constraints. The tractable query

[1] We do not expect to find many people to disagree with this statement as far as temporal constraints are concerned. Spatial constraints are trickier and will probably require our attention for some more time until we know enough about them. Also, some important advances in this area (e.g., (Renz & Nebel 1999)) are very recent.

[2] Which has always been more than a constraint-network based temporal reasoner!

[3] The results in the rest of the paper are taken verbatim from (Koubarakis & Skiadopoulos 1999).

[4] So all previous results on various classes of temporal and spatial constraints will actually be crucial here!

answering problems identified in this way are bound to be interesting for temporal reasoning researchers. Two of them are significant extensions of tractable problems identified previously in (Brusoni, Console, & Terenziani 1995; van Beek 1991).

The organization of this paper is as follows. The next section presents some preliminaries. Then we present the model of indefinite constraint databases. The last two sections develop our contributions. All proofs are omitted.

Preliminaries

In this paper we consider first order constraint languages. For each such language $\mathcal{L}_C$, we assume an *intended structure* $\mathcal{M}_C$ which interprets formulas of $\mathcal{L}_C$. $Th(\mathcal{M}_C)$ will denote the *theory* of this structure. Finally, for each language $\mathcal{L}_C$ a class of formulas called $\mathcal{L}_C$*-constraints* will be defined.

For example let us consider $\mathcal{L}_{LIN}$: the first order language of *linear constraints over the rationals.* Its intended structure is $\mathcal{M}_{LIN} = (\mathcal{Q}, +, *, <)$ where $\mathcal{Q}$ is the set of rational numbers. We will deal with the following classes of $\mathcal{L}_{LIN}$-constraints:

- The class of *linear inequalities* LIN.
- The class of *Horn disjunctive linear constraints* HDL. An HDL constraint is a disjunction of an arbitrary number of linear disequations and *at most one* weak linear inequality.
- The class of *linear constraints with two variables per inequality and unit coefficients* UTVPI$^{\neq}$. A UTVPI$^{\neq}$ constraint is a linear constraint $ax + by \sim c$ where $a, b \in \{-1, 0, 1\}$ and $\sim$ is $\leq$ or $\neq$.

HDL constraints were defined in (Koubarakis 1996; Jonsson & Bäckström 1996). They subsume many interesting classes of temporal constraints. UTVPI$^{\neq}$ constraints (without disequations) have been considered in (Shostak 1981; Jaffar *et al.* 1994). UTVPI$^{\neq}$ constraints subsume the temporal constraints studied in (Koubarakis 1997a). They can also be used in spatial applications to define many kinds of polygons (e.g., arbitrary rectangles, several kinds of triangles and octagons and so on). The disequations allowed by this class give us more expressive power by allowing a limited form of negative information (e.g., they can be used to *remove* points and straight lines from polygons).

We assume that the reader is familiar with the concept of *satisfiability* of a set of constraints, and the concept of *variable elimination.*

Theorem 1 *The satisfiability of a set of HDL constraints can be decided in PTIME (Koubarakis 1996; Jonsson & Bäckström 1996).*

Unfortunately we do not have a nice theorem like the above concerning variable elimination for the class HDL. In fact variable elimination cannot be done in PTIME even for LIN. 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 (Yannakakis 1988).

The following result extends a similar result in (Koubarakis 1997a).

Theorem 2 *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.*

The Scheme of Indefinite Constraint Databases

In this section we present the scheme of indefinite constraint databases originally proposed in (Koubarakis 1997b). We follow the spirit (and details) of the original proposal but use first order logic instead of relational database theory.

Let $\mathcal{L}_C$ be a many-sorted first-order constraint language and $\mathcal{M}_C$ be its *intended structure.* We assume that the class of $\mathcal{L}_C$-constraints *admits variable elimination* and is *weakly closed under negation* (i.e., the negation of every $\mathcal{L}_C$-constraint is equivalent to a disjunction of $\mathcal{L}_C$-constraints). Many interesting constraint languages have this property e.g., the language $\mathcal{L}_{LIN}$ defined previously.

We start by introducing some formal tools that make our presentation easy to follow. Let $\mathcal{L}_=$ be a first order language with equality and a countably infinite set of constant symbols. The intended structure $\mathcal{M}_=$ for $\mathcal{L}_=$ interprets $=$ as equality and constants as "themselves". $\mathcal{L}_=$*-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. In the database formalism to be developed below constants of $\mathcal{L}_=$ will be used to represent real-world entities with no special semantics e.g., *John* or *breakfast.*

Now let $\mathcal{L}_C$ be any first order constraint language. We will use $\mathcal{L}_{C,=}$ to denote the union of $\mathcal{L}_C$ and $\mathcal{L}_=$. The intended structure for $\mathcal{L}_{C,=}$ is $\mathcal{M}_C \cup \mathcal{M}_=$ and will be denoted by $\mathcal{M}_{C,=}$. Finally let us define the first order language $\mathcal{L}^*_{C,=}$. $\mathcal{L}^*_{C,=}$ is obtained by augmenting $\mathcal{L}_C$ with the *new* sort symbol $\mathcal{D}$ (for real-world entities with no special semantics), the equality symbol $=$ for sort $\mathcal{D}$, a countably infinite set of *database predicate symbols* $p_1, p_2, \ldots$ of various arities, and a countably infinite set of variables $x_1, x_2, \ldots$ of sort $\mathcal{D}$. In other words $\mathcal{L}^*_{C,=}$ is obtained from $\mathcal{L}_{C,=}$ by introducing a countably infinite set of predicate symbols $p_1, p_2, \ldots$ of various arities.

Databases And Queries

Let $\mathcal{L}_C$ be a first order constraint language. In this section the symbols $\overline{\mathcal{T}}$ and $\overline{\mathcal{T}}_i$ will denote vectors of sorts of $\mathcal{L}_C$. The symbol $\overline{\mathcal{D}}$ will denote a vector with all its components being the sort $\mathcal{D}$.

Indefinite $\mathcal{L}_C$-constraint databases and queries are special formulas of $\mathcal{L}^*_{C,=}$ and are defined as follows.

Definition 1 *Let $\mathcal{L}_C$ be a first order constraint language. An* indefinite $\mathcal{L}_C$-constraint database *is a formula $DB(\overline{\omega})$ of $\mathcal{L}^*_{C,=}$ of the following form:*

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

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

where

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

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

The second part of the database formula is a set of equivalences *defining* the database predicates p_i. These equivalences may refer to the Skolem constants of the constraint store. For temporal reasoning applications, the constraint store can be used instead of a constraint network while the predicates p_i can be used to encode the events or facts usually associated with the nodes of temporal constraint networks.

For a given database DB the first conjunct of the database formula will be denoted by $ConstraintStore(DB)$, and the second one by $EventsAndFacts(DB)$. 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:

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

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

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 (Kanellakis, Kuper, & Revesz 1990), and the marked nulls proposal of (Imielinski & Lipski 1984; Grahne 1991).

Example 1 *The following is an indefinite $\mathcal{L}_{LIN}$-constraint database:*

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

This database contains information about the events walk (taking a walk) and paper (reading a paper) in which Mary participates. The temporal information is indefinite since we do not know the exact constraint between Skolem constants ω_2 and ω_3.

Let us now define queries. The concept of query defined here is more expressive than the query languages for temporal constraint networks proposed in (Brusoni *et al.* 1994; van Beek 1991). It is very similar to the query language used in TMM (Schrag, Boddy, & Carciofini 1992).

Definition 2 *A* first order modal query *over an indefinite $\mathcal{L}_C$-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 ϕ is a formula of $\mathcal{L}^*_{C,=}$ and OP is the modal operator $\Diamond$ or $\Box$.*

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

Example 2 *The following query asks "Who is possibly having a walk before reading the paper?":*

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

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

Definition 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 $\mathcal{L}_C$-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 $\mathcal{L}_C$-constraints in variables $\overline{t}$ and $\mathcal{L}_=$-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 $\mathcal{M}_{C,=}$ on the interpretation of the symbols of $\mathcal{L}_{C,=}$, and M satisfies $\phi(\overline{x}, \overline{t})$ under V then V satisfies $answer(\overline{x}, \overline{t})$ and vice versa.*

The definition of answer in the case of certainty queries is defined accordingly. 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 *The answer to the query of Example 2 is $(x = mary,\ \emptyset)$.*

Query evaluation over indefinite $\mathcal{L}_C$-constraint databases can be viewed as quantifier elimination in the theory $Th(\mathcal{M}_{C,=})$. $Th(\mathcal{M}_{C,=})$ admits quantifier elimination as a consequence of the assumptions given at the beginning of this section. The following theorem is essentially from (Koubarakis 1997b).

Theorem 3 *Let DB be the indefinite $\mathcal{L}_C$-constraint database*

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

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

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 $\mathcal{L}_=$-constraints in variables $\overline{y}$ and $\mathcal{L}_C$-constraints in variables $\overline{z}$ obtained by eliminating quantifiers from the following formula of $\mathcal{L}_{i,=}$:

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

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 $\mathcal{L}_C$). $\psi(\overline{y}, \overline{z}, \overline{\omega})$ 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 $\mathcal{L}_C$-constraints. This equivalent disjunction is obtained by consulting the definition

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

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.

Due to lack of space we do not give any examples of this procedure. Examples can be found in (Koubarakis 1997b).

Let us close this section by pointing out that what we have defined is a *database scheme*. Given various choices of $\mathcal{L}_C$ one can use the developed formalism to study any kind of databases with indefinite information (e.g., temporal or spatial). 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 constraints is given in the final section of the paper.

Tractable Query Evaluation in Indefinite Constraint Databases

In this paper the complexity of database query evaluation is measured using the notion of *data complexity* (Vardi 1982). 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 has also been made in previous work on querying constraint networks (van Beek 1991). In our case we also assume that the size of any integer constant in the database is *logarithmic* in the size of the database (Kanellakis, Kuper, & Revesz 1990).

Evaluating possibility queries over indefinite constraint databases can be NP-hard even when we only have equality and inequality constraints between atomic values (Abiteboul, Kanellakis, & Grahne 1991) (similarly evaluating certainty queries is co-NP-hard). It is therefore important to seek tractable instances of query evaluation.

In this paper we start with the assumption that we have classes of constraints with some nice 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. We will reach tractable cases of query evaluation by restricting the classes of constraints, databases and queries we allow in our framework. We introduce the concepts of query type and database type to allow us to make these distinctions.

Query Types

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

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

The first argument of a query type distinguishes between closed and open queries. A query is called *closed* or *yes/no* if it does not have any free variables. Queries with free variables are called *open*. The argument *Modality* can be $\Diamond$ or $\Box$ representing possibility or necessity queries respectively.

The third argument can be *PositiveExistential* or *SinglePredicate*. We use *PositiveExistential* to denote that the query is a *positive existential* one i.e., it is of the form $(\exists \overline{x}/\overline{s})\phi(\overline{x})$ where ϕ involves only the logical symbols $\wedge$ and $\vee$. We use *SinglePredicate* to denote that the query is of the form $\overline{u}/\overline{s}_1 : \ OP \ (\exists \overline{t}/\overline{s}_2)p(\overline{u}, \overline{t})$ where $\overline{u}$ and $\overline{t}$ are vectors of variables, $\overline{s}_1$, $\overline{s}_2$ are vectors of sorts, p is a database predicate symbol and OP is a modal operator.

The fourth argument *Constraints* denotes the class of constraints that is used as query conditions. For example the query

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

is in the class $Q(Closed, \Box, PositiveExistential, LIN)$.

Database Types

A *database type* is a tuple of the form

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

where *Arity* is the arity of the database predicates, *LocalCondition* is the constraint class used in the definition of these predicates, and *ConstraintStore* is the class of constraints in the constraint store. *Arity* can have values *N-ary* and *Monadic*.

In one of our results we consider the type of a database which consists of a *single* N-ary predicate defined by constraints in some class $\mathcal{C}$, and the constraints in the constraint store belong to the same class $\mathcal{C}$. This is a special database type and is represented by

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

Complexity Results

The following definitions are needed below.

Definition 4 *If $\mathcal{C}$ is a class of constraints then $\vee\overline{\mathcal{C}}$ is a new class of constraints defined as follows. A constraint c is in $\vee\overline{\mathcal{C}}$ iff c is a disjunction of negations of $\mathcal{C}$-constraints.*

Definition 5 *Let $\mathcal{C}$ be a class of constraints. The problem of deciding whether a given set of constraints from $\mathcal{C}$ is satisfiable will be denoted by $SAT(\mathcal{C})$. The problem of eliminating an arbitrary number of variables from a given set of constraints from class $\mathcal{C}$ will be denoted by $VAR\text{-}ELIM(\mathcal{C})$. If the number of variables is fixed then the problem will be denoted by $FVAR\text{-}ELIM(\mathcal{C})$.*

In the lemmas and theorems of this section we often assume that $SAT(\mathcal{C})$, $VAR\text{-}ELIM(\mathcal{C})$ or $FVAR\text{-}ELIM(\mathcal{C})$ can be done in PTIME. The implicit parameter of interest here is the size of the input constraint set.

Let us now present our results assuming the data complexity measure. We first consider closed queries.

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

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

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

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

where CS and θ_i are conjunctions of $\mathcal{C}$-constraints.

Theorem 4 *Let $\mathcal{C}$ be a class of constraints such that $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{C})$ can be solved in PTIME. Let 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.*

Lemma 2 *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E} \subseteq \mathcal{C}$ and $\vee\overline{\mathcal{E}} \subseteq \mathcal{C}$. Evaluating a query of the class $Q(Closed, \Box, PositiveExistential, \mathcal{E})$ over an indefinite constraint database 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 and θ_i, $1 \leq i \leq m$ are $\mathcal{C}$-constraints.

Theorem 5 *Let $\mathcal{E}, \mathcal{C}$ be two classes of constraints such that $\mathcal{E} \subseteq \mathcal{C}$, $\vee\overline{\mathcal{E}} \subseteq \mathcal{C}$ and $SAT(\mathcal{C})$ and $FVAR\text{-}ELIM(\mathcal{E})$ can be solved in PTIME. Let 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 3 *Let $\mathcal{C}$ be a class of constraints. Evaluating a query of the class*

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

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

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

where CS and θ_i are conjunctions of $\mathcal{C}$-constraints and $\overline{u}$ is the vector of free variables of the query.

Theorem 6 *Let $\mathcal{C}$ be a class of constraints such that $VAR\text{-}ELIM(\mathcal{C})$ can be done 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.*

Lemma 4 *Let $\mathcal{C}$ be a class of constraints which is closed under negation. Evaluating a query of the class $Q(Open, \Box, SinglePredicate, None)$ over an indefinite constraint database of the class $SinglePredDB(N\text{-}ary, Single\text{-}\mathcal{C}, \mathcal{C})$ is equivalent to eliminating quantifiers from a formula of the form $CS(\overline{\omega}) \wedge \theta_1(\overline{u}, \overline{\omega}) \wedge \ldots \wedge \theta_l(\overline{u}, \overline{\omega})$ where CS is a conjunction of $\mathcal{C}$-constraints, θ_i, $1 \leq i \leq l$ are $\mathcal{C}$-constraints, $\overline{u}$ is the vector of free variables of the query and l is the number of disjuncts in the disjunction defining the single predicate referred in the query.*

Theorem 7 *Let $\mathcal{C}$ be a class of constraints such that $\mathcal{C}$ is closed under negation and $VAR\text{-}ELIM(\mathcal{C})$ can be done in PTIME. Let DB be an indefinite constraint database of the class $SinglePredDB(N\text{-}ary, Single\text{-}\mathcal{C}, \mathcal{C})$ 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.*

Applications to Temporal Reasoning

We will now specialize the general complexity results presented in the previous section to some interesting temporal and spatial domains. We will need 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 (Allen 1983).
- **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 (van Beek & Cohen 1990).

- **ORDHorn**. This is the subalgebra of **IA** which includes only interval relations that can be translated into conjunctions of ORD-Horn constraints on the endpoints of intervals (Nebel & Bürckert 1995). An *ORD-Horn constraint* is a disjunction of weak inequalities of the form $x \leq y$ and disequations of the form $x \neq y$ with the additional constraint that the number of inequalities should not exceed one (Nebel & Bürckert 1995).
- **PA** and **CPA**. **PA** is the Point Algebra of (Vilain, Kautz, & van Beek 1989). **CPA** is the subalgebra of **PA** which does not include the relation $\neq$.
- *None*. This is used when we only have the trivial constraints *true* and *false*.

Theorem 8 *Evaluation of*

1. $Q(Closed, \diamond, PositiveExistential, HDL)$ *queries over* $DB(N\text{-}ary, HDL, HDL)$ *databases,*
2. $Q(Closed, \square, PositiveExistential, LINEQ)$ *queries over* $DB(N\text{-}ary, LINEQ, HDL)$ *databases,*
3. $Q(Open, \diamond, PositiveExistential, \mathrm{UTVPI}^{\neq})$ *queries over* $DB(N\text{-}ary, \mathrm{UTVPI}^{\neq}, \mathrm{UTVPI}^{\neq})$ *databases and*
4. $Q(Open, \square, SinglePredicate, None)$ *queries over* $SinglePredDB(N\text{-}ary, Single\text{-}\mathrm{UTVPI}^{\neq}, \mathrm{UTVPI}^{\neq})$ *databases*

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

The first part of Theorem 8 is a significant extension of the PTIME result of (Brusoni, Console, & Terenziani 1995) on possibility queries over networks of temporal constraints of the form $x - y \leq c$.

Theorem 8 does not mention constraints on higher-order objects (e.g., intervals) explicitly so one might think that it useful for temporal reasoning problems involving only points. Luckily this is not true. For example, results for interval constraint databases can be deduced immediately by taking into account the subsumption relations between classes of interval and point constraints. For example, the first part of Theorem 8 implies that evaluating

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

queries over $DB(N\text{-}ary, None, \mathbf{ORDHorn})$ databases can be done with PTIME data complexity. This is a significant extension of the PTIME result of (van Beek 1991) on possibility queries over networks of **SIA** constraints. We can also derive an extension of the PTIME results of (van Beek 1991; Brusoni, Console, & Terenziani 1995) for certainty queries with a simple modification of our definitions of database and query. The details will be given in the long version of this paper.

Theorem 8 does not constrain the arity of database predicates. (van der Meyden 1992) has shown that evaluating $Q(Closed, \square, PositiveExistential, \mathbf{CPA})$ queries over $DB(Monadic, None, \mathbf{CPA})$ databases can be done in PTIME (compare with Part 2 of Theorem 8). If we move to databases with binary predicates, the corresponding query answering problem becomes NP-complete (van der Meyden 1992). Unlike (van der Meyden 1992), we have not proven any lower bound results in this paper. More research is necessary for drawing an informative picture of tractable vs. intractable query answering problems for indefinite constraint databases.

Acknowledgements

This research has been partially supported by project CHOROCHRONOS funded by EU's 4th Framework Programme. Spiros Skiadopoulos has also been supported by a postgraduate fellowship from NATO. Most of this work was performed while both authors were with the Dept. of Computation, UMIST, U.K.

References

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

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

Brusoni, V.; Console, L.; Pernici, B.; and Terenziani, P. 1994. 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.

Brusoni, V.; Console, L.; Pernici, B.; and Terenziani, P. 1995. Extending temporal relational databases to deal with imprecise and qualitative temporal information. In Clifford, J., and Tuzhilin, A., eds., *Recent Advances in Temporal Databases (Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, September 1995)*, Workshops in Computing. Springer.

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

Cheng, C., and Smith, S. 1994. Generating feasible schedules under complex metric constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

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

Egenhofer, M. 1994. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering* 6:86–95.

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

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

Jaffar, J.; Maher, M. J.; Stuckey, P.; and Yap, R. 1994. Beyond Finite Domains. In Borning, A., ed., *Proceedings of PPCP'94*, volume 874 of *Lecture Notes in Computer Science*, 86–94. Springer Verlag.

Jonsson, P., and Bäckström, C. 1996. A linear programming approach to temporal reasoning. In *Proceedings of AAAI-96*.

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

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

Koubarakis, M. 1997a. From Local to Global Consistency in Temporal Constraint Networks. *Theoretical Computer Science* 173:89–112. 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.

Koubarakis, M. 1997b. The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases. *Theoretical Computer Science* 171:25–60. Special Issue on Uncertainty in Databases and Deductive Systems, Editor: L.V.S. Lakshmanan.

Koubarakis, M., and Skiadopoulos, S. 1999. Querying Temporal Constraint Networks in PTIME. In *Proceedings of AAAI-99*. Forthcoming.

Miller, S., and Schubert, L. 1988. Using Specialists to Accelerate General Reasoning. In *Proceedings of AAAI-88*, 161–165.

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

Nebel, B., and Bürckert, H.-J. 1995. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM* 42(1):43–66.

Papadias, D.; Theodoridis, Y.; Sellis, T.; and Egenhofer, M. 1995. 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*, 92–103.

Penberthy, J. S., and Weld, D. 1994. Temporal Planning with Continuous Change. In *Proceedings of AAAI-94*, 1010–1015.

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

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

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

Subrahmanian, V. 1998. *Principles of Multimedia Database Systems*. Morgan Kaufmann.

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

van Beek, P., and Cohen, R. 1990. Exact and Approximate Reasoning about Temporal Relations. *Computational Intelligence* 6:132–144.

van der Meyden, R. 1992. 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*, 331–345. Full version appears in JCSS, 54(1), pp. 113-135, 1997.

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

Vilain, M.; Kautz, H.; and van Beek, P. 1989. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In Weld, D., and de Kleer, J., eds., *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann. 373–381.

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