Applied Intelligence 17, 297–311, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

Querying Temporal Constraint Networks: A Unifying Approach

MANOLIS KOUBARAKIS
Department of Electronic and Computer Engineering, Technical University of Crete, University Campus, Kounoupidiana, 73 100 Chania, Crete, Greece
manolis@intelligence.tuc.gr
www.intelligence.tuc.gr/∼manolis

Abstract. We develop the scheme of indefinite constraint databases using first-order logic as our representation language. When this scheme is instantiated with temporal constraints, the resulting formalism is more expressive than standard temporal constraint networks. The extra representational power allows us to express temporal knowledge and queries that have been impossible to express before. To make our claim more persuasive, we survey previous works on querying temporal constraint networks and show that they can be viewed as an instance of the scheme of indefinite constraint databases.

Keywords: temporal knowledge, temporal constraints, constraint networks, indefinite constraint databases

1. Introduction

The last fifteen years have been very productive for research in temporal reasoning. Researchers have defined various formalisms, most notably *temporal constraint networks* [1], and studied algorithms for consistency checking, finding a solution and computing the minimal network [2–17]. There have also been various implementations of temporal reasoning systems based on the theoretical models [1, 11, 18–21]. All these implementations use a temporal constraint network as the underlying formalism for representing temporal information.

When temporal constraint networks are used to represent temporal information 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 [21–23]. 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 non-temporal as well as temporal information 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 [24] and TMM [25–28]. EPILOG uses the temporal reasoner Timegraph [11], Shocker uses TIMELOGIC, Telos uses a subclass of Allen's interval algebra [1] while TMM uses networks of difference constraints [5].

In parallel with these developments the state of the art in algorithms for temporal constraint networks has improved dramatically and our understanding of the theoretical and practical issues involved has matured. As a result, some researchers [10, 21–23, 29–32] have actively pursued the combination of these two strands of research to develop representational frameworks and systems that offer sophisticated query languages for temporal constraint networks. These efforts can be understood to proceed on the footsteps of TMM [25, 26] the first temporal reasoning system to augment a temporal constraint network with a Prolog-like language for representing other kinds of useful non-temporal knowledge.

This paper proposes the *scheme of indefinite constraint databases* as the formalism that can unify the proposals of [10, 21–23, 29–32]. The proposed formalism is a *scheme* because it can be instaniated with various kinds of constraints defined by a first-order language. When the constraints chosen are temporal, the resulting formalism is more expressive than the corresponding temporal constraint networks. To make our claim more persuasive, we show how previous research on querying temporal constraint networks [21–23, 31] can be viewed as an instance of the scheme of indefinite constraint databases. The same is true for previous research on querying temporal databases with relative and indefinite information [10, 29, 30, 32].

This paper shows that in order to achieve the required expressive power and functionality, we must be prepared to go from temporal constraint networks (or conjunctions of temporal constraints) to *first order theories of temporal constraints* as studied in [33, 34]. We identify *variable elimination* (and its logical analogue *quantifier elimination*) as the main technical tool needed by the proposed framework (theses concepts have been mostly ignored by temporal constraint network research). We show that query evaluation in the proposed formalism can be viewed as quantifier elimination in a first order language of temporal constraints.

Recently we have made the same arguments in the field of constraint-based extensions of relational databases [35]. In this paper we develop similar machinery in a first-order logic setting. In addition we show explicitly how the proposed scheme subsumes earlier proposals. We hope this effort will make our work more accessible (and hopefully more useful!) to Artificial Intelligence researchers.

The paper is organized as follows. Section 2 introduces the temporal constraint languages that we will study. Section 3 introduces the problems of deciding the satisfiability of a set of constraints, and performing variable or quantifier elimination. Section 4 introduces the proposed formalism: the scheme of indefinite constraint databases. Sections 5, 6 and 7 show that the formalisms of [10, 21–23, 29–32] are subsumed by the scheme of indefinite constraint databases. Finally, Section 8 presents our conclusions and discusses future work.

2. Constraint Languages

We start by introducing some concepts useful for the developments in forthcoming sections. We will deal with many-sorted first order languages [36]. 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.

The rest of this section defines several progressively more complex first order temporal constraint languages.

2.1. The Language PA

The language *PA* is a very simple language that we can use for talking about temporal phenomena. The logical symbols of *PA* include: parentheses, a countably infinite set of variables, the equality symbol $=$ and the standard sentential connectives. There is only one non-logical symbol: the predicate symbol $<$. The intended structure $\mathcal{M}_{PA}$ has the set of rational numbers $\mathcal{Q}$ as its domain, and interprets predicate symbol $<$ as the relationship "less than" over the rational numbers. We will freely use other defined predicates like $\leq$ and $\neq$.

PA-constraints are exactly the constraints of the well-known *Point Algebra* **PA** defined in [2, 4, 7, 8].

Example 1. The following is a set of *PA*-constraints:

$$e_1 < e_2, \quad e_2 \leq e_3, \quad e_3 = e_4, \quad e_4 \neq e_5$$

Researchers have also considered the subalgebra of **PA** which does not include the relation $\neq$. This algebra is called the *Convex Point Algebra* (**CPA**) [3].

2.2. The Language IA

The language *IA* is a first order language that allows us to make similar distinctions to the ones allowed by *PA*. The difference is that *IA* is a language for *intervals*. The logical symbols of *IA* include: parentheses, a countably infinite set of variables and the standard sentential connectives. *IA* has 13 predicate symbols inspired from [1]:

before, *after*, *meets*, *met-by*, *during*, *over*, *overlaps*, *overlapped-by*, *starts*, *started-by*, *finishes*, *finished-by*, *equal*

The intended structure $\mathcal{M}_{IA}$ has the set of intervals over $\mathcal{Q}$ as its domain [33]. Predicates are interpreted as binary relations over intervals in the obvious way [33].

IA-constraints are exactly the constraints of the *Interval Algebra* **IA** defined in [1] and subsequently studied by [4, 8, 9, 33] and others.

Example 2.2. Let us consider the following example from [23]:

"Fred was reading the paper while eating his breakfast. He put the paper down and drank the last of his coffee. After breakfast he went for a walk."

The above paragraph asserts the following *IA*-constraints among events *breakfast*, *paper*, *coffee* and *walk*:

breakfast before walk,
coffee during breakfast,
paper overlaps breakfast $\vee$
paper overlapped-by breakfast $\vee$
paper starts breakfast $\vee$ *paper started-by breakfast* $\vee$
paper during breakfast $\vee$ *paper over breakfast* $\vee$
paper finishes breakfast $\vee$
paper finished-by breakfast $\vee$ *paper equals breakfast*,
paper overlaps coffee $\vee$ *paper starts coffee* $\vee$
paper during coffee

Interval algebra researchers have also considered a subalgebra of **IA** called **SIA**. **SIA** includes only relations which translate into conjunctions of endpoint relations in **PA** [4].

2.3. *The Language LIN*

The language *LIN* is also a first order language (*LIN* comes from *lin*ear). The logical symbols of $\mathcal{L}_{LIN}$ include: parentheses, a countably infinite set of variables, the equality symbol $=$ and the standard sentential connectives. The non-logical symbols of $\mathcal{L}_{LIN}$ 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 $\mathcal{M}_{LIN}$ has the set of rational numbers $\mathcal{Q}$ as its domain. $\mathcal{M}_{LIN}$ assigns to each constant symbol an element of $\mathcal{Q}$, 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 $\mathcal{Q}$.

LIN-constraints are the well-known class of linear constraints known from linear programming [37]. *LIN* constraints are useful for temporal reasoning because they allow the representation of *quantitative* temporal information (e.g., the duration of interval I is less than 5 minutes, event A lasts at least 5 hours more than event B etc.).

We will pay particular attention to a special subclass of *LIN*-constraints called HDL-constraints. *HDL-constraints* or *Horn disjunctive linear constraints* have been defined originally in [14–16, 38].

Example 2.3. The following is a set of HDL-constraints:

$$x_1 - x_2 \leq 5, \quad 3x_1 + x_2 \leq 3,$$
$$x_4 + 4x_5 \neq 6, \quad x_1 - x_5 \neq 2 \vee x_6 \neq 7,$$
$$4x_1 + 3x_2 - 5x_5 \leq 5 \vee x_1 - x_2 \neq 4 \vee x_3 + 3x_4 \neq 5$$

Interval algebra researchers have also considered a subalgebra of **IA** called **ORD-Horn**. **ORD-Horn** includes only relations which translate into conjunctions of endpoint relations that are *HDL*-constraints [9].

2.4. *The Language LATER*

The language *LATER* is a first-order language inspired by the temporal reasoning system LATER [21, 22, 39]. It has three sorts: $\mathcal{P}$ for time points, $\mathcal{I}$ for time intervals and $\mathcal{DUR}$ for durations. The constant symbols of *LATER* include *dates* and *times* of the form

$$month/day/year\ hour : minute$$

and *durations* of the form

$$days : hours : minutes$$

(followed by the word *days*, *hours* or *minutes*). Times with the smallest duration are of sort $\mathcal{P}$ while everything else is of sort $\mathcal{I}$. Durations are of sort $\mathcal{DUR}$.

LATER has two function symbols *start* and *end* with sort $\mathcal{I} \rightarrow \mathcal{P}$.

The predicate symbols of *LATER* have been defined in detail in [21, 22]. They include the *convex* predicates of **PA** ($<, \leq, >, \geq, =$) [2], the 13 basic predicates of **IA** [1] and the 10 basic point-to-interval predicates of [6]. There are also functions (e.g., *start*, *end*) and predicates (e.g., *lasting*, *lasting at least*, *since*, *until*, *at*) that can be used to assert durations of intervals and locations of points on the time line.

The intended structure $\mathcal{M}_{LATER}$ interprets dates as integer elements of $\mathcal{Q}$ and durations as positive integers. The interpretation of function and predicate symbols is the obvious one.

LATER-constraints have been defined in detail in [21, 22]. They offer a nice temporal reasoning framework since they include many useful classes of qualitative and metric temporal constraints. However, because disjunctive relations are *carefully controlled*, the expressive power of *LATER*-constraints is not greater than the expressive power of *difference constraints* as studied in [31, 40]. The complete set of functions and predicates can be found in [21. 22].

Example 2.4. The following set of *LATER*-constraints provides information about the working hours of Tom, Mary and Ann:

> *TomWork since* 1/1/1995 14:15,
> *TomWork until* 1/1/1995 18:30
> *TomWork before MaryWork*,
> *MaryWork lasting at least* 4:40 *hours*
> *start*(*AnnWork*) *at* 1/1/1995,
> *AnnWork lasting* 3:00 *hours*,
> *end*(*AnnWork*) *before* 1/1/1995 18:00

2.5. Other Languages

Temporal reasoning researchers have studied other languages of temporal constraints. The following languages deserve being mentioned here even though they are not defined in detail; the careful reader will probably have no difficulty in doing so after consulting the relevant publications.

Dechter et al. [40] have studied the language *DIFF* of *difference constraints*. *DIFF* deals only with points, and allows us to express constraints on the location of points on the rational line (e.g., $x < 2$) or on the distance of one point from another (e.g., $5 \leq x - y \leq 8$). [12, 13, 41] have extended the work of Dechter et al. to consider disequations of the form $x - y \neq r$ (r is a rational constant) as basic constraints. Our definition of *DIFF-constraints* will not include such disequations.

Meiri [6] and Kautz and Ladkin [42] have studied the language *QMPIA* (Meiri's term) that deals with points and intervals and mixes qualitative and metric constraints between points and intervals. More precisely, *QMPIA* allows *IA*-constraints between intervals, *PA*-constraints between points and *DIFF*-constraints between points or interval endpoints. Our class of *QMPIA-constraints* includes all the *qualitative/metric point-to-point/interval-to-interval/point-to-interval* constraints considered in [6]. Meiri [6] studies *QMPIA*-constraints using *general temporal constraint networks.*

2.6. Relationships Between Classes of Temporal Constraints

Several relationships hold between the classes of temporal constraints defined in the above sections. They are captured in Fig. 1 as stated in the following theorem.

Theorem 2.1. *The subsumption relations of Fig.* 1 *hold. Subsumption relations between a class of interval constraints* (*e.g.*, **SIA**) *and a class of point constraints* (*e.g.*, *PA*) *mean that each constraint in the first class can be expressed by a conjunction of constraints in the second class. Classes not connected by an arrow are incomparable.*

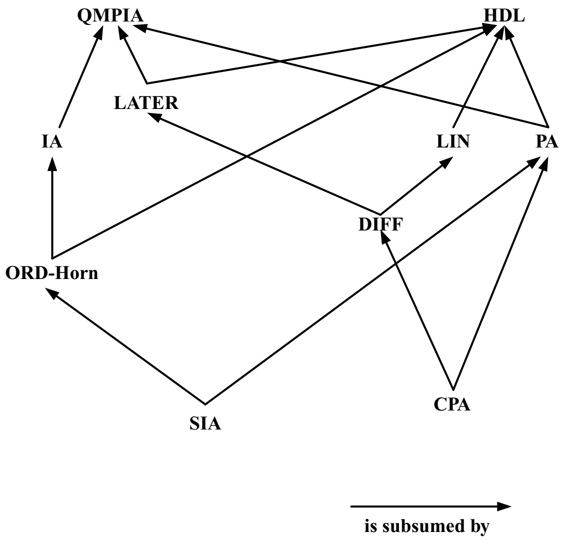

Figure 1. Subsumption relations between temporal constraint classes.

This section has defined several temporal constraint languages and constraint classes. We now turn to some interesting related problems in constraint-based reasoning.

3. Satisfiability, Variable Elimination and Quantifier Elimination

In the framework presented in this paper, two problems are important: deciding the satisfiability of a set of constraints, and performing variable or quantifier elimination. Satisfiability of temporal constraints has been studied by the research community continuously since [1]. Unfortunately, variable and quantifier elimination have not been paid the attention they deserve except in the work of the present author [12–14, 34, 35]. This section is an introduction to these important problems.

Definition 3.1. Let C be a set of $\mathcal{L}$-constraints in variables $x_1, \ldots, x_n$. The *solution set* of C, denoted by $Sol(C)$, is the following relation:

$$\big\{\big(x_1^0, \ldots, x_n^0\big) : \big(x_1^0, \ldots, x_n^0\big) \in \mathrm{domain}(\mathcal{M}_{\mathcal{L}})^n \quad \textit{and} \\ \textit{for every } c \in C, \big(x_1^0, \ldots, x_n^0\big) \textit{ satisfies } c\big\}.$$

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

Definition 3.2. A set of constraints (in some language $\mathcal{L}$) is called *satisfiable* or *consistent* if and only if its solution set is nonempty.

Example 3.1. The set of constraints of Example 2.1 is satisfiable. Tuple (1, 2, 3, 3, 5) is one of its solutions.

A lot of previous research has concentrated on the complexity of checking the satisfiability of a set of temporal constraints, and has identified tractable and possibly intractable constraint classes (e.g., see [3, 7, 9, 14–16, 40, 43]). The following theorem summarises the core results.

Theorem 3.1

1. *Deciding the satisfiability of a set of HDL-constraints can be done in PTIME* [14–16, 38]. *As a result, the same is true for all temporal constraint classes of Fig.* 1 *that are subsumed by HDL-constraints.*
2. *Deciding the consistency of a set of IA-constraints is NP-hard* (*so the same is true for QMPIA-constraints*) [3].

Let us now define the operations of quantifier and variable elimination. Quantifier elimination is an operation from mathematical logic [36]. Variable elimination is an algebraic operation [37]. As we will see below, quantifier elimination algorithms utilize variable elimination algorithms as subroutines. In the scheme of indefinite constraint databases introduced in Section 4, the operation of quantifier elimination is very useful because it can be used for query evaluation.

Definition 3.3. 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 [36]. 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 [36]:

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 3.4. 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 [37]. 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 3.2. Let C be the following set of linear constraints:

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

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

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

The following theorem is easy.

Theorem 3.2. *Let $\mathcal{L}$ be any of the languages defined in Section* 2. *The theory $Th(\mathcal{M}_\mathcal{L})$ admits quantifier elimination.*

Proof: Algorithms can be developed that eliminate variables from sets of *PA*, *IA*, *HDL*, *LATER*- and *QMPIA*-constraints. For *PA* and *HDL* the algorithms are given in [12, 13] and [14]. For the rest of the classes variable elimination algorithms can be readily developed using similar techniques. The existence of quantifier elimination algorithms follows easily (see also [34]). □

It is not difficult to see that the above quantifier elimination algorithm has exponential complexity even for theories with polynomial time variable elimination algorithms. Luckily more sophisticated quantifier elimination algorithms exist and have been studied by computational complexity theorists in the 70s and 80s [44–53] and more recently by constraint database researchers [35, 54].

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

4. The Scheme of Indefinite Constraint Databases

In this section we present the scheme of indefinite constraint databases originally proposed in [35]. 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 3 has defined this concept precisely). For the purposes of this paper $\mathcal{L}$ can be any of the languages of Section 2 e.g., the language *LIN*.

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 [1, 5, 7] 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 information like the above is represented by utilising a first-order temporal language like *LIN* and extending it to represent non-temporal 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.

4.1. 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 $\mathcal{M}_{\mathcal{L}\cup\mathcal{EQ}} = \mathcal{M}_{\mathcal{L}} \cup \mathcal{M}_{\mathcal{EQ}}$.

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 $(\mathcal{L} \cup \mathcal{EQ})^*$.

Example 4.1. Let $\mathcal{L}$ be the language *LIN* 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 $(LIN \cup \mathcal{EQ})^*$ capturing the fact that somebody took a walk during some unknown interval of time:

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

4.2. Databases and Queries

In this section the symbols $\bar{\mathcal{T}}$ and $\bar{\mathcal{T}}_i$ will denote vectors of sorts of $\mathcal{L}$. Similarly, the symbol $\bar{\mathcal{D}}$ 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 4.1. An *indefinite constraint database* is a formula $DB(\bar{\omega})$ of $(\mathcal{L} \cup \mathcal{EQ})^*$ of the following form:

$$\bigwedge_{i=1}^{m} (\forall \bar{x}_i/\bar{\mathcal{D}})(\forall \bar{t}_i/\bar{\mathcal{T}}_i)\left(\bigvee_{j=1}^{l_i} Local_j(\bar{x}_i, \bar{t}_i, \bar{\omega}) \equiv p_i(\bar{x}_i, \bar{t}_i)\right) \wedge ConstraintStore(\bar{\omega})$$

where

- $Local_j(\bar{x}_i, \bar{t}_i, \bar{\omega})$ is a conjunction of $\mathcal{L}$-constraints in variables $\bar{t}_i$ and Skolem constants $\bar{\omega}$, and $\mathcal{EQ}$-constraints in variables $\bar{x}_i$.
- $ConstraintStore(\bar{\omega})$ is a conjunction of $\mathcal{L}$-constraints in Skolem constants $\bar{\omega}$.

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. $\bar{\omega}$ 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 [55]).

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 [54], and the marked null values proposal of [56, 57]. Similar ideas can also be found in the first order databases of [55].

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

Example 4.2. The following is an indefinite constraint database which formalises the information in the paragraph considered at the beginning of this section.

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

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

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

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 $\omega_3 - \omega_2 \geq 60$.
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 define queries. The concept of query defined here is more expressive than the query languages for temporal constraint networks proposed in [21–23], and it is similar to the concept of query in TMM [27].

Definition 4.2. A *first order modal query* over an indefinite constraint database is an expression of the form $\bar{x}/\bar{\mathcal{D}}, \bar{t}/\bar{\mathcal{T}} : OP\phi(\bar{x}, \bar{t})$ 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 in *certainty* or *necessity* queries ($\Box$) and *possibility* queries ($\Diamond$).

Example 4.4. The following query refers to the database of Example 4.2 and asks "Who was the person who possibly had a conversation with Fred during this person's walk in the park?":

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

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 [58–60]. 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 4.3. Let q be the query $\bar{x}/\bar{\mathcal{D}}, \bar{t}/\bar{\mathcal{T}} : \diamond\, \phi(\bar{x}, \bar{t})$ over an indefinite constraint database DB. The answer to q is a pair $(answer(\bar{x}, \bar{t}),\ \emptyset)$ such that

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

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

   where $Local_j(\bar{x}, \bar{t})$ is a conjunction of $\mathcal{L}$-constraints in variables $\bar{t}$ and $\mathcal{EQ}$-constraints in variables $\bar{x}$.
2. Let V be a variable assignment for variables $\bar{x}$ and $\bar{t}$. 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 $\phi(\bar{x}, \bar{t})$ under V then V satisfies $answer(\bar{x}, \bar{t})$ and vice versa.

We have chosen the notation $(answer(\bar{x}, \bar{t}),\ \emptyset)$ to signify that *an answer is also a database* which consists of a single predicate defined by the formula $answer(\bar{x}, \bar{t})$ 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 4.5. The answer to the query of Example 4.4 is $(x = Mary,\ \emptyset)$.

The definition of answer in the case of certainty queries is the same as Definition 4.3 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 $\bar{x}$ and $\bar{t}$. If M satisfies $\phi(\bar{x}, \bar{t})$ under V then V satisfies $answer(\bar{x}, \bar{t})$ and vice versa.

Definition 4.4. A query is called *closed* or *yes/no* if it does not have any free variables. Queries with free variables are called *open*.

Example 4.6. The query of Example 4.4 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 4.7. The answer to the query of Example 4.6 is $(true, \emptyset)$ i.e., yes.

Let us now give some more examples of queries.

Example 4.8. Let us consider the database of Example 4.3 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:

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

Example 4.9. The following query refers to the database of Example 4.3 and asks "Is it certain that the first Chemotherapy appointment for Smith is scheduled to take place in the first month of 1996?":

$$: \Box\, (\exists t_1, t_2/\mathcal{Q})(treatment(Smith, Chem1, t_1, t_2) \wedge 0 \leq t_1 < t_2 \leq 31)$$

The answer to this query is no.

4.3. 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 [61]). The following theorem is essentially from [35].

Theorem 4.1. *Let DB be the indefinite constraint database*

$$\bigwedge_{i=1}^{m}(\forall \bar{x}_i/\bar{\mathcal{D}})(\forall \bar{t}_i/\bar{\mathcal{T}}_i)\left(\bigvee_{j=1}^{l_i} Local_j(\bar{x}_i, \bar{t}_i, \bar{\omega}) \equiv p_i(\bar{x}_i, \bar{t}_i)\right) \wedge ConstraintStore(\bar{\omega})$$

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

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

In this formula the vector of Skolem constants $\bar{\omega}$ *has been substituted by a vector of appropriately quantified variables with the same name* ($\bar{\mathcal{T}}'$ *is a vector of sorts of* $\mathcal{L}$). $\psi(\bar{y}, \bar{z}, \bar{\omega})$ *is obtained from* $\phi(\bar{y}, \bar{z})$ *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*

$$\bigvee_{j=1}^{l_i} Local_j(\bar{x}_i, \bar{t}_i, \bar{\omega}) \equiv p_i(\bar{x}_i, \bar{t}_i)$$

of predicate p_i *in the database* DB.

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

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

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

Example 4.10. Using the above theorem, the query of Example 4.4 can be answered by eliminating quantifiers from the formula:

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

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

Answering queries by the above method is mostly of theoretical interest. For implementations of this scheme more efficient alternatives have to be considered.

Let us close this section by pointing out that what we have defined is a *database scheme*. Given various choices for $\mathcal{L}$ (e.g., $\mathcal{L} = LIN$), one gets a *model* of indefinite constraint databases (e.g., the model of indefinite *LIN*-constraint databases). Examples of such instantiations will be seen repeatedly in the forthcoming Sections 5, 6 and 7 where we demonstrate that the proposals of [10, 21–23, 29–32] are subsumed by the scheme of indefinite constraint databases.

5. The LATER System

In [21, 22, 31] sets of *LATER*-constraints are considered as knowledge bases with indefinite temporal knowledge, and are queried in sophisticated ways using a first-order modal query language. This section will show that query answering in the LATER system is really an instance of the scheme of indefinite constraint databases.

We will first specify a method for translating a LATER knowledge base KB (i.e., a set of *LATER*-constraints) to an *indefinite LATER-constraint database* DB. The translation is done in two steps. First, for each symbolic point or interval I in KB, we introduce a fact $happens_I(\omega_I)$ in $EventsAndFacts(DB)$ where $happens_I$ is a new database predicate and ω_I is a new Skolem constant of appropriate sort. Then, for each constraint c between symbolic intervals I and J in KB, we introduce the same constraint between Skolem constants ω_I and ω_J in $ConstraintStore(DB)$.

Example 5.1. The following is the indefinite *LATER*-constraint database which corresponds to the LATER knowledge base of Example 4.2.[3]

$(\{happens_{TomWork}(\omega_{TomWork}),$
$happens_{MaryWork}(\omega_{MaryWork}),$
$happens_{AnnWork}(\omega_{AnnWork})\},$
$\{\omega_{TomWork}$ *Since* 1/1/1995 14:15,
$\omega_{TomWork}$ *Until* 1/1/1995 18:30,
$\omega_{TomWork}$ *Before* $\omega_{MaryWork}$,
$\omega_{MaryWork}$ *Lasting At Least* 4:40 *hours*,
$start(\omega_{AnnWork})$ *At* 1/1/1995,
$\omega_{AnnWork}$ *Lasting* 3:00 *hours*,
$end(\omega_{AnnWork})$ *Before* 1/1/1995 18:00$\})$

Now it is easy to translate queries over a LATER knowledge base to first order modal queries over an indefinite *LATER*-constraint database. We will consider all types of queries presented in [21, 22, 31].

1. *WHEN queries.* A WHEN query is of the form

$$WHEN\ T\ ?$$

where T is a symbolic point or interval in the queried LATER knowledge base. For the case of intervals, the corresponding query in our framework is

$$x/\mathcal{I} : happens_T(x)$$

and similarly for points.

Example 5.2. The query

$$WHEN\ TomWork\ ?$$

is translated into

$$x/\mathcal{I} : happens_{TomWork}(x)$$

and has the following answer over the database of Example 4.2:

$$(\{x = \omega_{TomWork}\},$$
$$\{\omega_{TomWork}\ Since\ 1/1/1995\ 14{:}15,$$
$$\omega_{TomWork}\ Until\ 1/1/1995\ 18{:}30\})$$

2. *MUST queries.* A MUST query in its simplest form is

$$must\ c(I, J)\ ?$$

where I, J are symbolic time intervals and c is a temporal constraint in LATER (similarly for points). The corresponding query in our framework is

$$:\Box\,(\exists x, y/\mathcal{I})(happens_I(x) \wedge happens_J(y) \wedge c(x, y))$$

The extension to arbitrary MUST queries is straightforward.

Example 5.3. The query

$$MUST\ overlaps(AnnWork, MaryWork)\ ?$$

can be translated into

$$:\Box\,(\exists x, y/\mathcal{I})(happens_{AnnWork}(x) \wedge$$
$$happens_{MaryWork}(y) \wedge Overlaps(x, y))$$

The answer to this query over the LATER *KB* of Example 4.2 is

$$(false,\ \emptyset)$$

which means NO.

3. *MAY queries.* The translation is similar to MUST queries but now the modal operator $\Diamond$ is used.
4. *Hypothetical queries.* The query language of our framework does not support hypothetical queries. They can be simulated by updating the database with an appropriate set of constraints and then asking a query.

6. Van Beek's Proposal for Querying Interval Algebra Networks

In [23] van Beek went beyond the typical reasoning problems studied for **IA** networks and considered them as knowledge bases about events that can be queried in more sophisticated ways. This section will show that van Beek's efforts can also be subsumed by our framework.

In [23] an **IA** knowledge base is a set of Interval Algebra constraints among appropriately named event constants (see Example 2.2). We will first specify a method for translating an **IA** knowledge base *KB* to an indefinite *IA*-constraint database DB. The translation is done in two steps. First, for each event e in *KB*, we introduce the facts

$$event(e), \quad happens(e, \omega_e)$$

in $EventsAndFacts(DB)$ where *event* and *happens* are database predicates and ω_e is a new Skolem constant of sort $\mathcal{I}$.[4] Then, for each constraint c between events e_1 and e_2 in *KB*, we introduce the same constraint between events ω_{e_1} and ω_{e_2} in $ConstraintStore(DB)$.

Example 6.1. The following is the indefinite *IA*-constraint database corresponding to the *IA*-constraints of Example 2.2:

$(\{event(breakfast), event(paper), event(coffee),$
$\quad event(walk),$
$happens(breakfast, \omega_{breakfast}),\ happens(paper, \omega_{paper}),$
$happens(coffee, \omega_{coffee}),\ happens(walk, \omega_{walk})\ \},$

$\{\omega_{breakfast}$ *before* ω_{walk},
ω_{coffee} *during* $\omega_{breakfast}$,
ω_{paper} *overlaps* $\omega_{breakfast} \vee \cdots \vee \omega_{paper}$ *equals* $\omega_{breakfast}$,
ω_{paper} *overlaps* $\omega_{coffee} \vee \omega_{paper}$ *starts* $\omega_{coffee} \vee$
ω_{paper} *during* $\omega_{coffee}\})$

The first component of the above pair asserts the existence of four events and their times. The second component asserts "all we know" about these times in the form of *IA*-constraints.

It is easy to translate queries over an **IA** KB to first order modal queries over an indefinite *IA*-constraint database. We will consider all types of queries presented in [23].

1. *Possibility* and *certainty* queries. These are very similar to MAY and MUST queries in LATER. The translation to our framework is also very similar. A certainty (resp. possibility) query is a formula of the form

$$OP\ \phi(e_1, \ldots, e_n)?$$

where OP is $\Box$ (resp. $\Diamond$), and ϕ is a quantifier free formula of *IA* with free variables $e_1, \ldots, e_n$. In our framework the corresponding query is

$$: OP(\exists x_1, \ldots, x_N/\mathcal{D})(\exists t_1, \ldots, t_n/\mathcal{I})$$
$$(event(x_1) \wedge \cdots \wedge event(x_n) \wedge happens(x_1, t_1) \wedge$$
$$\cdots \wedge happens(x_2, t_2) \wedge \phi(t_1, \ldots, t_n))$$

2. *Aggregation questions.* An aggregation question is of the form

$$x_1, \ldots, x_n : x_1 \in E \wedge \cdots \wedge x_n \in E \wedge OP\phi(x_1, \ldots, x_n)$$

where E is the set of all events in the KB, OP is the modal operator $\Diamond$ or $\Box$ and ϕ is a quantifier free first order formula of *IA*.
The corresponding query in our framework is

$$x_1, \ldots, x_n/\mathcal{D} : OP\ (\exists t_1, \ldots, t_n/\mathcal{I})$$
$$(event(x_1) \wedge \cdots \wedge event(x_n) \wedge happens(x_1, t_1) \wedge$$
$$\cdots \wedge happens(x_2, t_2) \wedge \phi(t_1, \ldots, t_n))$$

Example 6.2. The following **IA** KB provides information about a patient's visits to the hospital during the period 1990–1991:

1990 *meets* 1991,
*visit*4 *during* 1990, *visit*5 *during* 1990,
*visit*6 *during* 1991, *visit*7 *during* 1991,
*visit*4 *before visit*5, *visit*5 *before visit*6,
*visit*6 *before visit*7

The aggregation query

$$x : x \in Visits \wedge \Box(x \ during\ 1991)$$

where *Visits* is the set of all events can be translated into the following query in our framework:

$$x/\mathcal{D} : (\exists t/\mathcal{I})(event(x) \wedge happens(x, t) \wedge$$
$$\Box(x \ during\ 1991))$$

Note that calendars are not part of *IA*. To deal with them we follow our approach for *LATER*: calendar primitives (e.g., years) can be introduced as terms of the language and interpreted accordingly.

If the above query is executed over the indefinite *IA*-constraint database which corresponds to KB (it is easy to construct this database as it was done in Example 6.1) then it has the following answer:

$$(\{x = visit1,\ x = visit7\},\ \emptyset)$$

7. Other Proposals

In [10, 32] the LATER team extended the relational model of data with the temporal reasoning facilities of LATER. In their proposal, a relational database stores non-temporal information about events and facts which times are constrained by a set of *LATER*-constraints.

Earlier (and independently) similar work had been done by Koubarakis in [29, 30] where the model of indefinite temporal constraint databases was first defined as an extension of the relational data model.

The above data models and query languages are essentially instantiations of the scheme of indefinite constraint databases presented in this paper. The model of [10, 32] is essentially the model of *indefinite LATER-constraint databases.* Similarly the model of [29, 30] is the model of *indefinite DIFF-constraint databases.* The only notable difference is that in this paper we have developed our framework using first-order logic while Koubarakis, Brusoni, Console, Pernici and Terenziani use the relational data model.

Another related effort is of course TMM [25, 27] that can be seen to be an *ancestor* of all of the above systems. TMM has a very expressive representation language so it cannot be presented under the umbrella of the proposed scheme. However, if we omit *persistence assumptions, projection rules* and *dependencies* from the TMM formalism then the resulting subset is subsumed by indefinite *DIFF*-constraint databases.

8. Conclusions

We presented the scheme of indefinite constraint databases using first-order logic as our representation language. We demonstrated that when this scheme is instantiated with temporal constraints, the resulting formalism is more expressive than the standard machinery of temporal constraint networks. Previous proposals by [23] and [21] served to validate our claims.

In recent research we have also studied the complexity of query evaluation in indefinite temporal constraint databases for various classes of temporal constraints [35, 62–65]. As it might be expected the problem of evaluating first-order possibility or certainty queries over indefinite temporal constraint databases turns out to be very hard (NP-complete for possibility queries and co-NP-complete for certainty queries if we use the data complexity measure). Fortunately, there are many useful cases when query evaluation is tractable [62].

Our current research concentrates in the application of similar ideas to spatial constraint databases and their use in querying multimedia databases (e.g., "Give me all the images where there is an olive tree *to the left* of a house"). The main technical challenge here is to develop variable and quantifier elimination algorithms for interesting classes of spatial constraints. To the best of our knowledge, none of these problems has been studied so far.

Finally, implementation techniques for models based on our proposal are urgently needed. Not much has been done in this area with the exception of work by the LATER and TMM groups [26, 32].

Acknowledgments

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. I would like to thank Spiros Skiadopoulos for his contributions on related work, and Paolo Terenziani for detailed comments on a previous version of this paper. I would also like to thank the reviewers for comments that improved my presentation.

Notes

1. See `www.cs.rochester.edu/research/epilog/`.
2. See `www.cs.rochester.edu/research/kr-tools.html`.
3. In this and the next section we do not follow Definition 4.1 precisely for reasons of clarity and prefer to write sets of conjuncts instead of conjunctions. Also, when it comes to *EventsAndFacts*(DB), we write positive atomic formulas of first order logic and mean the *completions* of these formulas [55].
4. Let $\mathcal{I}$ be the only sort of language *IA*.

References

1. J. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.
2. M. Vilain and H. Kautz, "Constraint propagation algorithms for temporal reasoning," in *Proceedings of AAAI-86*, 1986, pp. 377–382.
3. M. Vilain, H. Kautz, and P. van Beek, "Constraint propagation algorithms for temporal reasoning: A revised report," in *Readings in Qualitative Reasoning about Physical Systems*, edited by D. Weld and J. de Kleer, Morgan Kaufmann: San Mateo, CA, pp. 373–381, 1989.
4. P. van Beek and R. Cohen, "Exact and approximate reasoning about temporal relations," *Computational Intelligence*, vol. 6, pp. 132–144, 1990.
5. R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, nos. 1–3, pp. 61–95, 1991, Special volume on knowledge representation.
6. I. Meiri, "Combining qualitative and quantitative constraints in temporal reasoning," in *Proceedings of AAAI-91*, 1991, pp. 260–267.
7. P. van Beek, "Reasoning about qualitative temporal information," *Artificial Intelligence*, vol. 58, pp. 297–326, 1992.
8. P. Ladkin and R. Maddux, "On binary constraint problems," *Journal of the ACM*, vol. 41, no. 3, pp. 435–469, 1994.
9. B. Nebel and H.-J. Bürckert, "Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra," *Journal of the ACM*, vol. 42, no. 1, pp. 43–66, 1995.
10. V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "Extending temporal relational databases to deal with imprecise and qualitative temporal information," in *Recent Advances in Temporal Databases* (Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, September 1995), edited by J. Clifford and A. Tuzhilin. Workshops in Computing. Springer, 1995.
11. A. Gerevini and L. Schubert, "Efficient algorithms for qualitative reasoning about time," *Artificial Intelligence*, vol. 74, pp. 207–248, 1995.
12. 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), Cassis, France, vol. 976 of LNCS, pp. 53–69, 1995.

13. M. Koubarakis, "From local to global consistency in temporal constraint networks," *Theoretical Computer Science*, vol. 173, pp. 89–112, 1997. Invited submission to the special issue dedicated to the *1st International Conference on Principles and Practice of Constraint Programming (CP95)*, edited by U. Montanari and F. Rossi.
14. M. Koubarakis, "Tractable disjunctions of linear constraints," in *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming (CP'96)*, Boston, MA, 1996, pp. 297–307.
15. P. Jonsson and C. Bäckström, "A linear programming approach to temporal reasoning," in *Proceedings of AAAI-96*, 1996, AAAI Press/MIT Press, pp. 1235–1240.
16. P. Jonsson and C. Bäckström, "A unifying approach to temporal constraint reasoning," *Artificial Intelligence*, vol. 102, pp. 143–155, 1998.
17. S. Staab, "On non-binary temporal relations," in *Proceedings of ECAI-98*, 1998, pp. 567–571.
18. A. Gerevini, L. Schubert, and S. Schaeffer, "Temporal reasoning in timegraph I-II," *SIGART Bulletin*, vol. 4, no. 3, pp. 21–25, 1993.
19. E. Yampratoom and J. Allen, "Performance of temporal reasoning systems," *SIGART Bulletin*, vol. 4, no. 3, pp. 26–29, 1993.
20. J. Stillman, R. Arthur, and A. Deitsch, "Tachyon: A constraint-based temporal reasoning model and its implementation," *SIGART Bulletin*, vol. 4, no. 3, 1993.
21. V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "LaTeR: An efficient, general purpose manager of temporal information," *IEEE Expert*, vol. 12, no. 4, pp. 56–64, 1997.
22. 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*, vol. 869 of Lecture Notes in Computer Science, Springer-Verlag: Berlin, 1994.
23. P. van Beek, "Temporal query processing with indefinite information," *Artificial Intelligence in Medicine*, vol. 3, pp. 325–339, 1991.
24. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: A language for representing knowledge about information systems," *ACM Transactions on Information Systems*, vol. 8, no. 4, pp. 325–362, 1990.
25. T. Dean and D. McDermott, "Temporal data base management," *Artificial Intelligence*, vol. 32, pp. 1–55, 1987.
26. T. Dean, "Using temporal hierarchies to efficiently maintain large temporal databases," *Journal of ACM*, vol. 36, no. 4, pp. 687–718, 1989.
27. R. Schrag, M. Boddy, and J. Carciofini, "Managing disjunction for practical temporal reasoning," in *Proceedings of KR'92*, 1992, pp. 36–46.
28. M. Boddy, "Temporal reasoning for planning and scheduling," *SIGART Bulletin*, vol. 4, no. 3, pp. 17–20, 1993.
29. M. Koubarakis, "Representation and querying in temporal databases: The power of temporal constraints," in *Proceedings of the 9th International Conference on Data Engineering*, 1993, pp. 327–334.
30. M. Koubarakis, "Database models for infinite and indefinite temporal information," *Information Systems*, vol. 19, no. 2, pp. 141–173, 1994.
31. V. Brusoni, L. Console, and P. Terenziani, "On the computational complexity of querying bounds on differences constraints," *Artificial Intelligence*, vol. 74, no. 2, pp. 367–379, 1995.
32. V. Brusoni, L. Console, P. Terenziani, and B. Pernici, "Qualitative and quantitative temporal constraints and relational databases: Theory, architecture, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 6, pp. 948–968, 1999.
33. P. Ladkin, "Satisfying first-order constraints about time intervals," in *Proceedings of AAAI-88*, 1988, pp. 512–517.
34. M. Koubarakis, "Complexity results for first-order theories of temporal constraints," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, 1994, Morgan Kaufmann: San Francisco, CA, pp. 379–390.
35. M. Koubarakis, "The complexity of query evaluation in indefinite temporal constraint databases," *Theoretical Computer Science*, vol. 171, pp. 25–60, 1997. Special issue on Uncertainty in Databases and Deductive Systems, edited by L.V.S. Lakshmanan.
36. H. Enderton, *A Mathematical Introduction to Logic*, Academic Press: New York, 1972.
37. A. Schrijver (ed.), *Theory of Integer and Linear Programming*, Wiley: New York, 1986.
38. M. Koubarakis, "Tractable disjunctions of linear constraints: Basic results and applications to temporal reasoning," *Theoretical Computer Science*, vol. 266, pp. 311–339, 2001.
39. L. Console and P. Terenziani, "Efficient processing of queries and assertions about qualitative and quantitative temporal constraints," *Computational Intelligence*, vol. 15, no. 4, pp. 442–465, 1999.
40. R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," in *Proceedings of 1st International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ontario, edited by R. Brachman, H. Levesque, and R. Reiter, pp. 83–93, 1989.
41. 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 appeared in the *Proceedings of the Workshop on Spatial and Temporal Reasoning, IJCAI-95*.
42. H. Kautz and P. Ladkin, "Integrating metric and qualitative temporal reasoning," in *Proceedings of AAAI-91*, 1991, pp. 241–246.
43. A. Gerevini and L. Schubert, "On point-based temporal disjointness," *Artificial Intelligence*, vol. 70, pp. 347–361, 1994.
44. M. Fischer and M. Rabin, "Super-exponential complexity of presburger arithmetic," in *Proc. of AMS Symposium on Complexity of Real Computational Processes*, 1974, vol. III.
45. J. Ferrante and C. Rackoff, "A decision procedure for the first order theory of real addition with order," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 69–76, 1975.
46. J. Ferrante and J. Geiser, "An efficient decision procedure for the theory of rational order," *Theoretical Computer Science*, vol. 4, no. 2, pp. 227–233, 1977.
47. L. Stockmeyer, "The polynomial-time hierarchy," *Theoretical Computer Science*, vol. 3, pp. 1–22, 1977.
48. C. Reddy and D. Loveland, "Presburger arithmetic with bounded quantifier alternation," in *Proc. of ACM Symposium on the Theory of Computing*, 1978, pp. 320–325.

49. J. Ferrante and C. Rackoff, *The Computational Complexity of Logical Theories*, Lecture Notes in Mathematics, Springer Verlag: Berlin, 1979.
50. L. Berman, "The complexity of logical theories," *Theoretical Computer Science*, vol. 11, pp. 71–78, 1980.
51. A. Bruss and A. Meyer, "On time-space classes and their relation to the theory of real addition," *Theoretical Computer Science*, vol. 11, pp. 59–69, 1980.
52. M. Furer, "The complexity of presburger arithmetic with bounded quantifier alternation depth," *Theoretical Computer Science*, vol. 18, pp. 105–111, 1982.
53. E. Sontag, "Real addition and the polynomial time hierarchy," *Information Processing Letters*, vol. 20, pp. 115–120, 1985.
54. P. Kanellakis, G. Kuper, and P. Revesz, "Constraint query languages," in *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1990, pp. 299–313.
55. R. Reiter, "Towards a logical reconstruction of relational database theory," in *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, edited by M. Brodie, J. Mylopoulos, and J. Schmidt, Springer Verlag: Berlin, pp. 191–233, 1984.
56. T. Imielinski and W. Lipski, "Incomplete information in relational databases," *Journal of ACM*, vol. 31, no. 4, pp. 761–791, 1984.
57. G. Grahne, *The Problem of Incomplete Information in Relational Databases*, vol. 554 of Lecture Notes in Computer Science, Springer Verlag: Berlin, 1991.
58. H. Levesque, "Foundations of a functional approach to knowledge representation," *Artificial Intelligence*, vol. 23, pp. 155–212, 1984.
59. W.J. Lipski, "On semantic issues connected with incomplete information databases," *ACM Transactions on Database Systems*, vol. 4, no. 3, pp. 262–296, 1979.
60. R. Reiter, "On integrity constraints," in *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge*. Asilomar, CA, 1988, pp. 97–111.
61. P. Kanellakis, G. Kuper, and P. Revesz, "Constraint query languages," *Journal of Computer and System Sciences*, vol. 51, pp. 26–52, 1995.
62. M. Koubarakis and S. Skiadopoulos, "Querying temporal constraint networks in PTIME," in *Proceedings of AAAI-99*, 1999. pp. 745–750.
63. M. Koubarakis and S. Skiadopoulos, "Tractable query answering in indefinite constraint databases: Basic results and applications to querying spatio-temporal information," in *Spatio-Temporal Database Management (Proceedings of the International Workshop STDBM'99)*, vol. 1678 of LNCS, Springer: Berlin, pp. 204–223, 1999.
64. M. Koubarakis and S. Skiadopoulos, "Querying indefinite temporal and spatial information: A new frontier," in *Proceedings of the IJCAI-99 Workshop on Hot Topics in Temporal and Spatial Reasoning*, 1999.
65. M. Koubarakis and S. Skiadopoulos, "Querying temporal and spatial constraint networks in PTIME," *Artificial Intelligence*, vol. 123, nos. 1/2, pp. 223–263, 2000.