Chapter 19

Temporal CSPs

Manolis Koubarakis

19.1 Introduction

Reasoning with temporal constraints has been a hot research topic for the last twenty years. The importance of this topic has been recognized in many areas of Computer Science and Artificial Intelligence e.g., planning [4], scheduling [23], natural language understanding [91], knowledge representation [79], spatio-temporal databases and geographical information systems [62], constraint databases [89], medical information systems [102], computer-aided verification [5], multimedia presentations [2] etc.

Temporal reasoning is an area that has greatly benefited by the application of techniques from constraint programming ever since the early papers by James Allen and others [3, 107, 31, 108, 34]. The CSP framework introduced in Chapter 2 of this handbook is immediately applicable for representing and reasoning about temporal information, and so are the algorithms of Chapters 3, 4 and 8. Temporal CSPs have been proved to be a robust framework where general CSP results such as the ones surveyed in Chapters 5 and 6 of this handbook could be applied profitably. Moreover, specific results about temporal CSPs have often provided the motivation for deriving general results about CSPs. Temporal CSPs have been studied in depth, not only because of intellectual curiosity, but mostly due to their importance for applications such as planning, scheduling, temporal databases and others mentioned above. In many cases, the problems studied come straight from the application front and developed solutions are immediately put into practical use.

In this chapter, we survey work on temporal CSPs starting from the papers that appeared in the early nineties [3, 107, 31, 108, 34] and continue with contributions that have been published as recently as last year. We have covered all of the influential works, but due to space, we have sometimes been brief in our presentation. Our presentation is sometimes historical; we hope this will turn out to be useful for the readers. For more information on temporal CSPs and temporal reasoning in general, the reader can read the Handbook of Temporal Reasoning in Artificial Intelligence [41] or the original papers that have appeared in the literature.

The rest of this chapter is organized as follows. Section 19.2 introduces some preliminary concepts of temporal reasoning and temporal CSPs. Section 19.3 introduces the most influential temporal reasoning formalisms based on constraint networks that have been proposed in the literature and relevant algorithmic problems. Then, Section 19.4 discusses efficient constraint satisfaction algorithms for these formalisms. Section 19.5 introduces the application need for more expressive queries over temporal constraint networks (especially queries combining temporal and non-temporal information) and surveys various proposals that address this need. Sections 19.6 and 19.7 introduce the scheme of indefinite constraint databases that is, up to today, the most comprehensive proposal for querying hybrid representations consisting of a relational database component and a constraint network component. In the case of temporal CSPs, the constraint network can be used to store temporal constraints on various temporal objects, and the relational database to store facts referring to these objects. Finally, Section 19.8 concludes the chapter and points out some open problems.

19.2 Preliminaries

In this section, we introduce the topic of representing and reasoning about temporal information, and discuss the representational choices that have been made in the temporal reasoning literature. We also introduce some basic concepts of CSPs that will subsequently be used throughout the chapter.

19.2.1 Temporal Representation and Reasoning: Basic Concepts

In everyday life, most people are able to communicate their knowledge and understanding of temporal phenomena without any major difficulties. However, quite different intuitions surface as soon as people undertake to construct a formal temporal representation. The literature distinguishes among three approaches for representing temporal phenomena: the *change-based* approach (exemplified by situation calculus [74] or event calculus [64]), the *time-based* approach (exemplified by various temporal logics [106]) or temporal database models [62]) and their combination [86]. Research on temporal CSPs adopts a time-based approach to temporal representation and inference. Time is introduced explicitly via an appropriate set of times (called the *time structure*) and change is manifested when propositions become true or false at different elements of this set. Once one adopts this approach, the time structure must be precisely defined. The relevant issues here are:

- What are the elements of the time structure? *Points, intervals* or *both*? Research in temporal CSPs has usually adopted some set of numbers P (e.g., the rationals) to be the set of points and pairs $(x, y) \in P^2$ such that $x < y$ to be the set of intervals. Conventional time unit systems have also been studied (e.g., see the TUS system of [70]).

- Is time *totally ordered, partially ordered, branching* or *cyclic*? Research in temporal CSPs usually assumes time to be totally ordered. There has recently been some interesting work on CSPs for other models of time e.g., partially ordered time, branching time etc. [16].

- Is time *discrete* or *dense*? The issue here is whether there exists a unit of time which cannot be decomposed. Discrete time is usually considered to be isomorphic to the integers ($\mathbb{Z}$). Proponents of dense time have a choice between rationals ($\mathbb{Q}$) and reals ($\mathbb{R}$). Various kinds of temporal CSPs have been studied that deal nicely with all three cases.

- Is time *bounded* or *unbounded*? Time is unbounded when for every element of the time structure there is a "previous" and a "next" element. Temporal CSPs can easily handle both cases.

Once one adopts an ontology and a structure for time, one usually turns to another, equally important, consideration: what are the kinds of temporal knowledge that must be represented? There are many kinds of temporal information that are useful in applications such as the ones mentioned in Section 19.1:

- *Definite temporal information.* We have definite temporal information when the time associated with an event or fact is known to be equal to an *absolute time* i.e., a point or interval on the time line. In other words, the time associated with an event or fact is known to full precision in the desired level of granularity. For example, the sentences "The car was on service throughout March 25th, 1993" and "The car has gone for service every March 25th for the years 1993-2000" give definite temporal information with respect to the time line of the Gregorian calendar. Note that the information in the second sentence is *periodic.*

- *Indefinite or indeterminate temporal information.* We have indefinite temporal information when the time associated with an event or fact is either unknown or has not been fully specified. The time associated with an event or fact can be under-specified in various ways [39]:

  – The time associated with an event or fact might be specified via a *qualitative relationship* (different than equality) to some absolute time. As an example, consider the sentence "John became manager *after* March, 1993".

  – The time associated with an event or fact might be specified via a *relationship* to the time associated with another event or fact. In this case, the two times can be related through a *qualitative, metric* (or *quantitative*) or *mixed temporal constraint.* For example, consider the statements "The explosion occurred *after* John left the scene" (qualitative temporal information), "The explosion occurred *5 to 10 minutes after* John left the scene" (metric temporal information), and "The explosion occurred *5 to 10 minutes after* John left the scene *while* he was getting into his car" (mixed temporal information).

  – The granularity of the system time line does not match the precision to which the time associated with an event or fact is known. As an example, consider storing the information "John was hired on January 25, 1993" in a system with time-stamps in the granularity of a second.

  – Dating techniques can be imperfect. All clocks have inherent imprecision.

Temporal CSPs are an expressive framework and they can represent all the above types of temporal information.

19.2.2 Background on CSPs

The area of temporal CSPs was initiated by James Allen in his seminal paper [3]. Allen proposed to represent qualitative temporal knowledge by interval constraint networks. An *interval constraint network* (see Figure 19.1) is a directed graph where nodes represents intervals and edges are labelled with vectors (i.e., disjunctions) of the thirteen binary qualitative interval relations presented in [3]. Following [3], many researchers concentrated on CSPs (or, equivalently, constraint networks) as a means for representing and reasoning about temporal knowledge. Their proposals are surveyed in Section 19.3 of this chapter.

In this chapter, the equivalent terms *CSP, constraint network* and *set (conjunction) of constraints* will be used interchangeably. We now define formally some of the concepts from the standard CSP literature that we will use in this chapter. We use $dom(x_i)$ to refer to the domain of variable x_i.

Definition 19.1. *Let C be a set of constraints in variables $x_1, \ldots, x_n$. The* solution set *of C, denoted by $Sol(C)$, is the following relation:*

$$\{(v_1, \ldots, v_n) \in dom(x_1) \times \cdots \times dom(x_n) : \text{ for every } c \in C, (v_1, \ldots, v_n) \text{ satisfies } c\}.$$

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

Definition 19.2. *A set of constraints is called* consistent *or* satisfiable *if and only if its solution set is non-empty.*

We now define the standard concepts of i-consistency, strong i-consistency and global consistency (or decomposability).

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

Definition 19.3. *Let C be a set of constraints in variables $x_1, \ldots, x_n$ and $1 \leq i \leq n$. C is called* i-consistent *iff for every $i-1$ distinct variables $x_1, \ldots, x_{i-1}$, every valuation $u = \{x_1 \leftarrow v_1, \ldots, x_{i-1} \leftarrow v_{i-1}\}$ such that $v_1 \in dom(x_1), \ldots, v_{i-1} \in dom(x_{i-1})$ and u satisfies the constraints $C(x_1, \ldots, x_{i-1})$, and every variable x_i different from $x_1, \ldots, x_{i-1}$, there exists a value $v_i \in dom(x_i)$ such that u can be extended to a valuation $u' = u \cup \{x_i \leftarrow v_i\}$ which satisfies the constraints $C(x_1, \ldots, x_{i-1}, x_i)$. C is called* strong i-consistent *if it is j-consistent for every j, $1 \leq j \leq i$. C is called* globally consistent *or* decomposable *iff it is i-consistent for every i, $1 \leq i \leq n$.*

We now define the standard concept of minimal set of constraints. Minimal sets of constraints are especially important in temporal CSPs because they make explicit all implied *binary* constraints (e.g., the strictest constraints between the endpoints of an interval or the constraints capturing the strictest qualitative relation between two points etc.). In a constraint network representation of binary constraints, the concept of minimal constraint set is equivalent to the concept of *minimal network*.

Definition 19.4. *A set of constraints C will be called* minimal *if any instantiation of two variables, which satisfies the constraints involving these variables only, can be extended to a solution of C.*

In temporal CSPs, the variables are used to represent time elements (points or intervals), the domains are time structures (usually $\mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$ for time points, and the set of intervals over $\mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$ for time intervals), and the constraints represent temporal relationships. Section 19.3 presents various temporal CSP frameworks with appropriate choices for variables, domains and temporal constraints.

The following reasoning problems have been traditionally associated with CSPs:

- Deciding whether a set of constraints is consistent.
- Finding a solution or all the solutions of a consistent constraint set.
- Computing the minimal set of constraints equivalent to a given one.
- Determining if a set of constraints is i-consistent, strong i-consistent or globally consistent.

The above reasoning problems have also been the main focus of algorithms for temporal CSPs proposed in the literature. These algorithms are surveyed in Section 19.4 of this chapter.

19.3 Constraint-Based Formalisms for Reasoning About Time

In this section we initiate our survey of temporal representation and reasoning formalisms based on constraint networks. We distinguish the proposed formalisms depending on the kind of temporal information they allow: qualitative, metric or mixed temporal information.

19.3.1 Qualitative Temporal Reasoning

As we already said earlier, the first important paper that proposed to represent qualitative temporal information by CSPs was [3] by James Allen. In [3], Allen introduced a formalism for reasoning about intervals in time. An *interval* i is a pair (i^-, i^+) where i^- and i^+ are *endpoints* on the real line and $i^- < i^+$ holds. Allen's formalism is based on thirteen mutually exclusive *binary relations* which can capture all the possible ways two intervals can be related. These *atomic* relations are

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

and their inverses ($equals$ is its own inverse). Figure 19.2 defines these relations in terms of endpoint constraints, and gives a shorthand notation and pictorial representation for them.

Allen's formalism has received a lot of attention and has been the formalism of choice for representing qualitative interval information. Whenever the interval information to be represented is indefinite, a disjunction of some of the thirteen atomic relations can be used to represent what is known. There are 2^{13} such disjunctions representing qualitative relations between two intervals. Each one of these relations will be denoted by the set of its constituent basic relations e.g., $\{b, bi, d, m\}$. The *empty* relation will be denoted by $\perp$, and the *universal* relation will be denoted by $\top$. The set of all 2^{13} relations expressible in

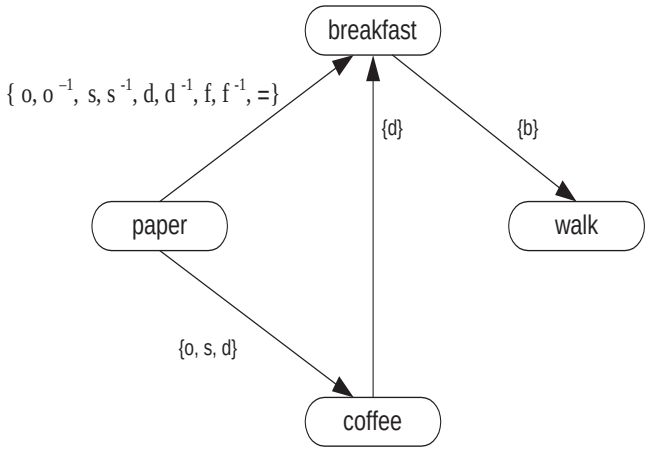

Figure 19.1: An $\mathcal{IA}$ network

Allen's formalism will be denoted by $\mathcal{IA}$. The operations of intersection ($\cap$), complement ($\cdot^{-1}$) and composition ($\circ$) can be defined on $\mathcal{IA}$ as follows:

$$(\forall x, y)(x\ r^{-1}\ y \Leftrightarrow y\ r\ x)$$

$$(\forall x, y)(x\ (r \cap r')\ y \Leftrightarrow (x\ r\ y\ \wedge\ x\ r'\ y))$$

$$(\forall x, y)(x\ (r \circ r')\ y \Leftrightarrow (\exists z)(x\ r\ z\ \wedge\ z\ r'\ y))$$

The set $\mathcal{IA}$ equipped with these operations forms an algebra [82], called the *interval algebra*.

Example 19.5. Let us consider the following text [103]:

> 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.

If we use $breakfast, paper, walk$ and $coffee$ to stand for appropriate time intervals, the information included in the above sentences is captured by the $\mathcal{IA}$ network of Figure 19.1.

In [3], Allen presented a constraint propagation algorithm for $\mathcal{IA}$ networks based on path consistency which runs in $O(n^3)$ time where n is the number of intervals in the network. When constraints are propagated, some temporal knowledge that has been implicit before is made explicit. Later on, Vilain and Kautz showed that Allen's constraint propagation algorithm is not complete because deciding the consistency of a set of $\mathcal{IA}$ constraints is an NP-complete problem and so is computing the minimal network [107].

In the same paper, Vilain and Kautz introduced the *point algebra* $\mathcal{PA}$ which allows one to relate two time points using the binary qualitative relations $<$, $>$ and $=$ and their disjunctive combinations (see Figure 19.2). [107] also identified the *pointisable subclass* $\mathcal{PIA}$ of $\mathcal{IA}$ which consists of all elements of $\mathcal{IA}$ that can be expressed as a conjunction of binary constraints using only elements of $\mathcal{PA}$.

Basic Relation	Symbol	Pictorial Representation	Endpoint Constraints
$i\ before\ j$	b	iiiiiiii jjjjjjjj	$i^- < j^-,\ i^- < j^+,$ $i^+ < j^-,\ i^+ < j^+$
$j\ after\ i$	b^{-1}		
$i\ meets\ j$	m	iiiiiiii jjjjjjjjjj	$i^- < j^-,\ i^- < j^+,$ $i^+ = j^-,\ i^+ < j^+$
j *met-by* i	m^{-1}		
$i\ overlaps\ j$	o	iiiiiiii jjjjjjjjjjj	$i^- < j^-,\ i^- < j^+,$ $i^+ > j^-,\ i^+ < j^+$
j *overlapped-by* i	o^{-1}		
$i\ during\ j$	d	iiiiiiii jjjjjjjjjjjjjjjj	$i^- > j^-,\ i^- < j^+,$ $i^+ > j^-,\ i^+ < j^+$
$j\ includes\ i$	d^{-1}		
$i\ starts\ j$	s	iiiiiiiii jjjjjjjjjjjjjj	$i^- = j^-,\ i^- < j^+,$ $i^+ > j^-,\ i^+ < j^+$
j *started-by* i	s^{-1}		
$i\ finishes\ j$	f	iiiiiiiiii jjjjjjjjjjjjjjjj	$i^- > j^-,\ i^- < j^+,$ $i^+ > j^-,\ i^+ = j^+$
$j\ finished\text{-}by\ i$	f^{-1}		
$i\ equals\ j$	$=$	iiiiiiiiiii jjjjjjjjjj	$i^- = j^-,\ i^- < j^+,$ $i^+ > j^-,\ i^+ = j^+$

Basic Relation	Symbol	Pictorial Representation	Point Constraints
$p\ before\ i$	b	p iiiiiiii	$p < i^-$
$i\ after\ p$	b^{-1}		
$p\ starts\ i$	s	p iiiiiiii	$p = i^-$
i *started-by* p	s^{-1}		
$p\ during\ i$	d	p iiiiiiii	$i^- < p < i^+$
$i\ includes\ p$	d^{-1}		
$p\ after\ i$	a	p iiiiiiii	$i^+ < p$
$i\ before\ p$	a^{-1}		

Basic Relation	Symbol	Pictorial Representation
$p\ before\ q$	$<$	p q
$q\ after\ p$	$>$	
$p\ equals\ q$	$=$	p q
$p\ after\ q$	$>$	q p
$q\ before\ p$	$<$	

Figure 19.2: Interval-to-interval, point-to-interval and point-to-point relations

In [107], Vilain and Kautz claimed that Allen's constraint propagation algorithm computes the minimal network for $\mathcal{PA}$. Subsequently, van Beek pointed out that this result is true only for the subset of $\mathcal{PA}$ which does not include the *disequality* relation $\neq$; this is the *convex point algebra* $\mathcal{CPA}$ [101, 108, 104]. The same result is true for the *continuous endpoint subclass* $\mathcal{CEIA}$ of $\mathcal{IA}$ which consists of all elements of $\mathcal{IA}$ that can be expressed as a conjunction of binary constraints using only elements of $\mathcal{CPA}$ [101, 108, 104]. Van Beek also pointed out that enforcing strong 4-consistency in an $\mathcal{PIA}$ or $\mathcal{PA}$ network results in an equivalent minimal network [101, 108, 104]. However, enforcing strong 4-consistency does not result in global consistency for these networks. As shown by Koubarakis in [57], strong 5-consistency is necessary and sufficient for achieving global consistency in $\mathcal{PIA}$ and $\mathcal{PA}$. Van Beek has also presented two efficient algorithms for $\mathcal{PA}$ and $\mathcal{PIA}$ networks: an $O(n^2)$ algorithm for consistency checking and finding a solution, and an $O(\max mn^2, n^3)$ for computing the minimal network [103]. The parameter n here is again the number of nodes in the network, while m is the number of edges labelled with $\neq$.

The work by Vilain, Kautz and van Beek [108] motivated the search for new subclasses of $\mathcal{IA}$ that are tractable. The most widely studied subclass discovered so far is the *Ord-Horn subclass* $\mathcal{H}$ introduced by Nebel and Bürckert in [82]. $\mathcal{H}$ consists of all relations $r \in \mathcal{A}$ which satisfy the following condition. If i and j are intervals, $i\ r\ j$ can be equivalently expressed as a conjunction of Ord-Horn constraints on the endpoints of i and j. An *Ord-Horn constraint* is a disjunction $d_1 \vee \cdots \vee d_n$ where at most one of the d_i's is an inequality of the form $x \leq y$, the rest of the d_i's are disequations of the form $x \neq y$, and x and y are variables ranging over the real numbers.

It is interesting to notice that $\mathcal{H}$, the most expressive tractable subclass of $\mathcal{IA}$ among the ones introduced above, consists of 868 relations i.e., it covers more than 10% of $\mathcal{A}$. $\mathcal{H}$ is *maximal* i.e., it cannot be extended without losing tractability [82].

Recently, Krokhin, Jeavons and Jonsson showed that there are *exactly* 18 maximal tractable subclasses of $\mathcal{IA}$; reasoning in any subset of $\mathcal{IA}$ not included in these subclasses is NP-complete [65]. This is an important *dichotomy* result: it classifies all subproblems of an NP-complete problem as either tractable or NP-complete. It is important to point out that this result is proved analytically while previous work had resorted to systematic computerized analysis (see e.g., [38]).

Koubarakis [59] has demonstrated that, in general, there is no low level of local consistency that can achieve global consistency of $\mathcal{H}$ constraints. Earlier, Bessière, Isli and Ligozat [12] had presented some subclasses of $\mathcal{H}$ for which path consistency achieves global consistency.

Gerevini [42] considers $\mathcal{PA}$ and $\mathcal{H}$ and studies *incremental* algorithms for checking consistency, maintaining a solution and maintaining the minimal network. The algorithms of [42] improve the static algorithms for these problems by a factor of $O(n)$ or $O(n^2)$ when a sequence of $O(n^2)$ operations (assertions or relaxations of constraints) are processed. In related work, Delgrande and Gupta [36] consider the problem of updating chains of $\leq$ or $<$ relations.

In [75], Meiri defines the *qualitative algebra* $\mathcal{QA}$, an expressive formalism for qualitative temporal reasoning on points and intervals. In $\mathcal{QA}$, one is able to express binary constraints of the form $o_i\ r_1\ o_j \vee \cdots \vee o_i\ r_k\ o_j$ where o_i, o_j are points or intervals and $r_1, \ldots, r_k$ are:

- interval-to-interval relations from $\mathcal{IA}$

- point-to-point relations from $\mathcal{PA}$
- *point-to-interval* or *interval-to-point* relations [109]. These five, mutually exclusive relations and their inverses can hold between a point and an interval. They are shown pictorially in Figure 19.2.

[75] presents several results on $\mathcal{QA}$ and its subclasses including how to combine it with metric information (see Section 19.3.3 below). Recently, [50] presented a dichotomy theorem which gives a *complete* classification of all subclasses of $\mathcal{QA}$ as either tractable or NP-complete.

The expressive power of the qualitative temporal reasoning algebras defined in this section can be summarized as follows (the symbol $\subset$ should be read as "contains" or "is less expressive than") :

$$\mathcal{CPA} \subset \mathcal{PA} \subset \mathcal{QA} \quad \text{and} \quad \mathcal{CEIA} \subset \mathcal{PIA} \subset \mathcal{H} \subset \mathcal{IA} \subset \mathcal{QA}$$

19.3.2 Metric Temporal Reasoning

Dechter, Meiri and Pearl studied metric temporal information using *disjunctive binary difference ($\mathcal{DBD}$) constraints*[1] of the form

$$a_1 \leq x_i - x_j \leq b_1 \vee \cdots \vee a_n \leq x_i - x_j \leq b_n$$

where x_i, x_j are real variables representing time points and $a_1, \ldots, a_n, b_1, \ldots, b_n$ are real numbers [34]. To deal with these constraints, [34] introduced *$\mathcal{DBD}$ networks* where nodes represent variables and arcs represent binary constraints.

Example 19.6. Let us consider the following text [34]:

> John goes to work either by car (30-40 minutes) or by bus (at least 60 minutes). Fred goes to work either by car (20-30 minutes) or in a car pool (40-50 minutes). Today John left home between 7:10 and 7:20, and Fred arrived at work between 8:00 and 8:10. We also know that John arrived at work about 10-20 minutes after Fred left home.

Let x_0 be a special time point (real variable) denoting the "beginning of time" (7:00 in our case). Let x_1, x_2, x_3, x_4 be real variables such that $[x_1, x_2]$ is the interval corresponding to John's travel to work, and $[x_3, x_4]$ is the interval corresponding to Fred's travel to work. The left part of Figure 19.3 shows a $\mathcal{DBD}$ network capturing the temporal relations in the above text.

Deciding consistency of $\mathcal{DBD}$ networks is NP-complete [34]. An important tractable subcase occurs when all constraints have a *single* disjunct i.e., they are of the form $a \leq x_i - x_j \leq b$. We will call these constraints simply *binary difference ($\mathcal{BD}$) constraints*. For the class of $\mathcal{BD}$ constraints, deciding consistency and at the same time computing the minimal

[1] In this section, we deviate from the usual terminology of the literature and name classes of metric temporal constraints by referring to what relationships they can express (e.g., difference, disjunctions etc.). In this way, we avoid using names formed with adjectives such as simple, complex etc. that do not say much about the expressivity of the particular constraint class they are used to name.

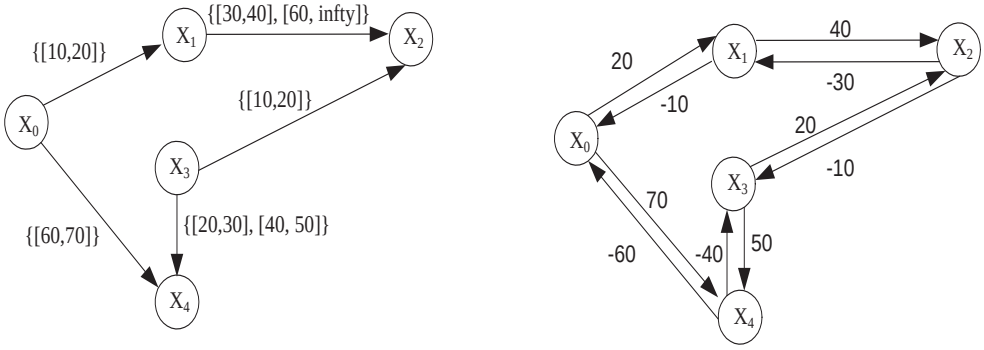

Figure 19.3: A $\mathcal{DBD}$ network (left) and a distance graph for a part of it (right)

network can be done in $O(n^3)$ time (where n is the number of variables) by running any all-pairs shortest-paths algorithm (e.g., Floyd-Warshall [29]) on an equivalent weighted, directed graph representation of the constraints called the *distance graph* [34]. The right part of Figure 19.3 shows the distance graph equivalent to the $\mathcal{BD}$ constraint network obtained from the $\mathcal{DBD}$ constraint network on the left part of the figure after dropping the interval $[60, infty]$ from the edge (x_1, x_2) and $[20, 30]$ from the edge (x_3, x_4).

For the class of $\mathcal{BD}$ constraints, computing the shortest-paths among all pairs of nodes in the distance graph is equivalent to enforcing path consistency in the original network. Notice also that path consistency is necessary and sufficient for achieving global consistency for the class of $\mathcal{BD}$ constraints [34]. Deciding consistency only can alternatively be achieved by a single-source shortest-paths algorithm (e.g., Bellman-Ford [29]) in $O(nE)$ time where n is the number of nodes and E the number of edges in the distance graph. Alternatively, one can use a *directional path consistency* algorithm on the given network which runs in $O(nW^*(d)^2)$ time where $W^*(d)$ is the maximum number of parents that a node possesses in the resulting network [34].

The framework of difference constraints of [34] has been influential in much future work in this area. For example, Koubarakis [57] and, independently, Gerevini and Christani [43] have introduced the class of *binary difference constraints with disequations* ($\mathcal{BD}^{\neq}$) by extending the class of $\mathcal{BD}$ constraints to include disequations of the form $x - y \neq r$ (r is a real constant). Deciding consistency in $\mathcal{BD}^{\neq}$ can be checked in $O(n^3)$ time by trivially modifying any all-pairs shortest path algorithm used for the class of $\mathcal{BD}$ constraints so that it reports inconsistencies resulting from any disequation $x - y \neq r$ and any (implied) equality of the form $x - y = r$. Computing the minimal network for $\mathcal{BD}^{\neq}$ constraints can be done in $O(\max mn^2, n^3)$ where n is the number of variables and m is the number of disequations [57, 43]. [57] has also shown that strong 5-consistency is the necessary and sufficient condition for achieving global consistency in the case of $\mathcal{BD}^{\neq}$ constraints. Recently, [63] extended this result to the class of *unit two-variable per inequality/disequation* ($UTVPI^{\neq}$) constraints. In addition to terms of the form $x - y$, this class allows terms of the form $x + y$ and the same comparison operators as $\mathcal{BD}^{\neq}$.

Extensions to the framework of [33] also explored more practical directions. For ex-

ample, [13] has shown how to extend this framework so that *multiple time granularities* are supported.

A related but more expressive class of temporal constraints which has also been studied widely in the literature is the class of n-ary disjunctive difference constraints. An *n-ary disjunctive difference ($\mathcal{NDD}$) constraint* is a formula of the form

$$a_1 \leq x_1 - y_1 \leq b_1 \vee \cdots \vee a_n \leq x_n - y_n \leq b_n$$

where $x_1, y_1, \ldots, x_n, y_n$ are real variables representing time points and $a_1, \ldots, a_n, b_1, \ldots, b_n$ are real numbers [97, 6, 96].

Example 19.7. The following are examples of $\mathcal{NDD}$ constraints:

$$x_1 - y_1 \leq 2, \;\; x_1 - y_1 \leq 5 \;\vee -2 \leq x_2 - y_2 \leq 2 \vee x_3 - y_3 \leq 4,$$

$$0 \leq x_4 - y_4 \vee 2 \leq x_5 \leq 5$$

Disjunctive constraints with disjuncts having different pairs of variables cannot be expressed in the $\mathcal{DBD}$ constraints framework of [34].

Example 19.8. Let I, J be intervals, I^-, J^- their beginning points and I^+, J^+ their ending points. The following $\mathcal{NDD}$ constraints express the fact that intervals I and J have duration between 5 and 10 minutes and they cannot overlap.

$$5 \leq I^+ - I^- \leq 10, \;\; 5 \leq J^+ - J^- \leq 10, \;\; I^+ - J^- \leq 0 \vee J^+ - I^- \leq 0$$

Example 19.9. Let I and J be intervals corresponding to the execution of operations O_I and O_J. O_I and O_J will be executed on a machine that can handle only one operation at a time and has a set up time of 2 minutes. Let I^-, J^- be the beginning points of I and J and I^+, J^+ their ending points.

The following is an appropriate constraint on the scheduling of operations O_I and O_J:

$$I^+ - J^- \leq -2 \vee J^+ - I^- \leq -2$$

Deciding the consistency of a set of $\mathcal{NDD}$ constraints is also NP-complete. *Boolean combinations of binary difference ($\mathcal{BCBD}$) constraints* have also been studied recently [98].

The quest for tractability of metric temporal CSPs received a big push forward when Koubarakis [59] and Jonsson and Bäckström [49] independently introduced the class of Horn-disjunctive linear constraints. A *linear constraint* ($\mathcal{LIN}$) is a formula of the form $(\sum_{i=1}^{n} a_i x_i)\ \theta\ r$ where a_1, ..., a_n, r are rational constants, x_1, ..., x_n are variables and θ is $\leq$ or $<$. We freely use $\geq, >$ and $=$ as well. A *Horn-disjunctive linear ($\mathcal{HDL}$) constraint* is a disjunction $d_1 \vee \cdots \vee d_n$ where each d_i is a weak linear inequality or a linear disequation, and the number of inequalities among $d_1, \ldots, d_n$ does not exceed one.

Example 19.10. The following are examples of $\mathcal{HDL}$ constraints:

$$3x_1 + x_5 - 3x_4 \leq 10, \;\; x_1 + x_3 + x_5 \neq 7,$$

$$3x_1 + x_5 - 4x_3 \leq 7 \vee 2x_1 + 3x_2 - 4x_3 \neq 4 \vee x_2 + x_3 + x_5 \neq \frac{5}{2},$$

$$4x_1 + x_3 \neq 3 \vee 5x_2 - 3x_5 + x_4 \neq 6$$

Deciding the consistency of a set of $\mathcal{HDL}$ constraints can be done in PTIME [56, 59, 49]. The main intuition behind this result is that disequations can be dealt with independently from one another for the purposes of consistency checking.

There are currently no relevant maximality results regarding the tractability of $\mathcal{HDL}$ constraints. [16] give two such maximal tractable subclasses of the class of disjunctions of $\mathcal{PA}$ relations.

[15] demonstrates how to implement efficiently consistency checking for $\mathcal{HDL}$ constraints when disjuncts are constrained to be of the form $x_i - x_j \leq a$ or $x_i - x_j \neq a$.

The properties of the class of $\mathcal{HDL}$ constraints have partly motivated Cohen et al. [27] to study questions of tractability for constraints that are obtained as *disjunctions of simpler constraints* with certain useful properties (e.g., independence, guaranteed satisfaction etc.). The importance of such results is that they are obtained in an *abstract CSP framework* and turn out to be useful for many kinds of specific CSPs e.g., temporal, spatial, etc. For some of the results in this area, the reader should see [26, 17] and Chapter 6 of this handbook

Recently, Kumar [68] pioneered the use of *randomized* algorithms for temporal CSPs. [68] initially presents a randomized algorithm for $\mathcal{BD}$ constraints. Then, the intuitions derived from this class are used to develop a strongly-polynomial deterministic algorithm, and a simple randomized algorithm for a restricted class of $\mathcal{NDD}$ constraints, denoted by $\mathcal{RNDD}$, which includes the following three types of constraints:

$$l_{ij} \leq x_i - y_j \leq u_{ij},\ \ a_1 \leq x_i \leq b_1 \vee \cdots \vee a_n \leq x_i \leq b_n,\ \ l_i \leq x_i \leq u_i \vee l_j \leq x_j \leq u_j$$

The expressive power of the metric temporal CSPs defined in this section can be summarized as follows:

$$\mathcal{LIN} \subset \mathcal{HDL}, \quad \mathcal{BD} \subset \mathcal{BD}^{\neq} \subset \mathcal{DBD} \subset \mathcal{NDD} \subset \mathcal{BCBD}, \quad \mathcal{BD}^{\neq} \subset \mathcal{HDL},$$

$$\text{and} \quad \mathcal{BD} \subset \mathcal{RNDD} \subset \mathcal{NDD}$$

19.3.3 Qualitative and Metric Temporal Reasoning Combined

Meiri [75] has combined the expressive power of the qualitative algebra $\mathcal{QA}$ and the $\mathcal{DBD}$ constraint framework of [34] to come up with a framework of binary mixed temporal constraint networks where nodes are points or intervals and constraints are qualitative from $\mathcal{QA}$ or quantitative from $\mathcal{DBD}$. Independently, Kautz and Ladkin [53] have proposed a very similar framework that combines qualitative constraints from the interval algebra $\mathcal{IA}$ and the $\mathcal{BD}$ constraints of [34].

Example 19.11. Let us consider the following text [75]:

> John and Fred work for a company that has local and main offices in Los Angeles. They usually work at the local office, in which case it takes John less than 20 minutes and Fred 15-20 minutes to get to work. Twice a week, John works at the main office, in which case his commute to work takes at least 60 minutes. Today John left home between 7:00-7:05 a.m. and Fred arrived at work 7:50-7:55 a.m. We also know that Fred and John met at a traffic light on their way to work.

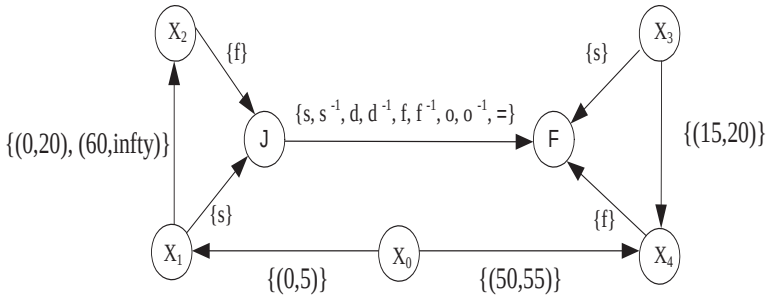

Figure 19.4: A network with qualitative and metric temporal constraints

Let x_0 be the real variable denoting the "beginning of time" (7:00 again). Let $J = [x_1, x_2]$ be the time interval corresponding to John's travel to work, and $F = [x_3, x_4]$ be the time interval corresponding to Fred's travel to work where x_1, x_2, x_3, x_4 are real variables representing the interval endpoints. Figure 19.4 shows a constraint network capturing the temporal relations in the above text.

More recently, Krokhin et al. presented another framework that combines qualitative and metric temporal reasoning [66]. In this case, the objects of interest are intervals and qualitative information is expressed in $\mathcal{IA}$. In addition, metric temporal information on interval endpoints can be expressed using $\mathcal{HDL}$ constraints. The important result of [66] is a dichotomy theorem that settles the standard tractability question for the proposed framework by completely characterizing all subproblems that are tractable; all the remaining ones are shown to be NP-complete. Since the framework of [66] subsumes the framework of [53], the tractability question for this framework has also been settled. The exact characterization of all tractable classes of the framework of [75] remains an open problem.

Qualitative reasoning about *durations* has also been considered in [80] and other papers and a formalism called *point-duration* networks has been defined. Point-duration networks start from $\mathcal{PA}$ networks and enrich them with binary comparisons of the times elapsed between pairs of points (i.e., durations of intervals). The comparison of these durations is also done using the relations of $\mathcal{PA}$. Similarly, Pujari et al. [84] have defined a similar framework called $\mathcal{INDU}$ for reasoning about intervals using $\mathcal{IA}$ and interval durations using $\mathcal{PA}$. It is not clear up to now, how far one can go with these two duration frameworks since, as pointed out in [10], the framework is not closed under the composition operation. Recently, Renz and Ligozat [88] discussed this issue in a general CSP context and differentiated between composition as defined in Section 19.3.1 for $\mathcal{IA}$ (and most other frameworks studied here) and *weak* composition. In general, the frameworks of [80, 84] are not as well developed currently as the rest of the frameworks surveyed in this chapter so we will not deal with them any further in this chapter.

19.4 Efficient Algorithms for Temporal CSPs

In the previous section, we have surveyed work on temporal CSPs, complexity results and algorithms for deciding consistency, computing the minimal network and enforcing global consistency. In typical temporal reasoning applications (e.g., planning and scheduling) the databases of temporal constraints to be handled are very large thus *scalability* of temporal reasoning algorithms becomes important. Unfortunately, the algorithms of Section 19.3 are *not* scalable. Even for the case of tractable temporal reasoning formalisms such as $\mathcal{PA}$, typical algorithms [103] require $O(n^2)$ space and $O(\max mn^2, n^3)$ time to answer queries. Researchers in temporal reasoning quickly recognized this problem and implemented efficient reasoners for various formalisms described in Section 19.3. This is the work that we survey in this section.

19.4.1 Efficient Algorithms for Qualitative Temporal CSPs

The work on efficient algorithms for qualitative temporal constraints can be distinguished into two categories: scalable algorithms for constraint classes with PTIME reasoning problems (especially $\mathcal{PA}$) and backtracking or local search algorithms for classes with NP-complete reasoning problems (especially $\mathcal{IA}$).

Efficient Algorithms for $\mathcal{PA}$

Len Schubert, Alfonso Gerevini and colleagues implemented and experimentally evaluated the temporal reasoners TimeGraph I and II for handling constraints expressed in $\mathcal{PA}$ [44]. The main idea in TimeGraph II, which is the most advanced version, is to represent sets of $\mathcal{PA}$ constraints by directed labelled graphs, partition these graphs into *chains* (i.e., linearly ordered points) where constant time reasoning is possible, and use a *meta-graph* to reason about points belonging to different chains. TimeGraph II also handles binary disjunctions of $\mathcal{PA}$ constraints using an intelligent backtracking algorithm [44]. TimeGraph I and II have been used in various planning and natural language understanding projects e.g., [91].

The work of [45] has also addressed scalability for $\mathcal{PA}$ networks using an approach which also relies heavily on an underlying directed graph structure. In this case, *spanning trees* are the basic data structure where efficient reasoning with respect to the $\leq$ relation is performed. The algorithms of [45] are incomplete for $\mathcal{PA}$ since they cannot handle cases involving the relation $\neq$ [44]. The work of [45] has been extended with metric constraints and has been utilized in the temporal reasoner of the IxTeT temporal planner [69].

[37] further extend the ideas of TimeGraph II by relying on *series-parallel graphs* (instead of chains) as their basic efficient data structure. [37] provides new intuitions regarding the techniques of TimeGraph II, and shows experimentally what improvements are possible when *series-parallel graphs* become the basic data structure.

Efficient Algorithms for $\mathcal{IA}$

Ladkin and Reinfeld [72] were the first to implement and evaluate experimentally backtracking algorithms for solving $\mathcal{IA}$ constraints. The backtracking algorithm of [72] has the following characteristics: a preprocessing step based on path consistency, instantiation of disjunctions by any set of $\mathcal{IA}$ relations for which path consistency is complete, chronological backtracking, and forward checking using path consistency. [105] improves [72]

with a more efficient version of path consistency and heuristics for dynamic variable ordering. [81] shows that performance improvements can be obtained if we use the class $\mathcal{H}$ for instantiating disjunctions in the backtracking algorithm of [72]. [72] and [81] also studied the phase transition of the problem of solving $\mathcal{IA}$ constraints.

[99] shows how to solve $\mathcal{IA}$ consistency checking problems using local search. In [99], a given $\mathcal{IA}$ problem with m interval variables is first translated into an equivalent (with respect to satisfiability) problem where the endpoints of the intervals are constrained to range over the integers $1, \ldots, 2m$. Then, this problem is solved using the discrete Langrangian method.

Let us now turn our attention to efficient algorithms for metric temporal CSPs. Here attention has been focused on $\mathcal{BD}$, $\mathcal{DBD}$ and $\mathcal{NDD}$ constraints. For $\mathcal{BD}$ constraints, the emphasis has been on improving existing polynomial time algorithms as well as devising incremental versions of such algorithms that are important in applications (e.g., planning or scheduling). Since the reasoning problems for classes $\mathcal{DBD}$ and $\mathcal{NDD}$ have exponential complexity, the emphasis there has been on backtracking algorithms and local search algorithms with influences from CSP and SAT solvers.

19.4.2 Efficient algorithms for $\mathcal{BD}$ and $\mathcal{DBD}$ constraints

[21] and [22] has considered incremental algorithms for networks of $\mathcal{BD}$ constraints. The idea in these algorithms is that when a new constraint is added or retracted, a constraint propagation algorithm is not run from scratch, but only some processing *local* to the insertion or deletion takes place. [21] concentrates on incremental arc-consistency algorithms for $\mathcal{BD}$ constraints while [22] presents an incremental version of the well-known Bellman-Ford algorithm for the single-source shortest-paths problem [29]. Similarly, [24] has presented an incremental version of the directional path consistency algorithm of [33].

Recently, Xu and Choueiry [111] presented an efficient algorithm for deciding the consistency of $\mathcal{BD}$ constraints. This algorithm essentially improves the partial path consistency algorithm of [14] (which operates on a triangulated constraint graph) and applies it to the case of $\mathcal{BD}$ constraints. [111] demonstrates experimentally that this algorithm improves on many of its competitors that have appeared in the literature [34] in the case of large and sparse constraint graphs.

[92, 32] consider checking the consistency of $\mathcal{DBD}$ constraints using backtracking algorithms that operate on the equivalent *meta-CSP* (i.e., the CSP with variables corresponding to disjunctions and values corresponding to $\mathcal{BD}$ disjuncts) and utilize local consistency algorithm like path consistency for preprocessing and forward checking. [92] points out that enforcing path consistency in networks of $\mathcal{DBD}$ constraints can result in the creation of an exponential number of intervals. Then, it develops alternative local processing algorithms that compute looser constraints than path consistency but do so in polynomial time. Finally, [92] demonstrates that significant savings are achieved when these local processing algorithms are combined with backtracking to check the consistency of sets of $\mathcal{DBD}$ constraints.

Xu and Choueiry [110] show alternative ways to improve on chronological backtracking algorithms for $\mathcal{DBD}$ constraints [92, 32]. Their techniques include utilizing the algorithm of [111] to check the consistency of the set of $\mathcal{BD}$ constraints considered at each node of the search tree, exploiting the constraint topology, having good variable-ordering heuristics, and reducing the domains with a special form of arc consistency [25]. More re-

cently, [95] have also investigated using the incremental all-pairs-shortest-path algorithm of [22] instead of [111] at each node of the search tree.

TMM (Time Map Manager) is another important temporal reasoning system with support for $\mathcal{BD}$ constraints [31]. The main contribution of TMM is *not* its CSP features but rather its querying facilities, its good support for *temporal persistence* and *causality*, and its sophisticated indexing algorithms for handling large databases of temporal propositions [30]. TMM will be again discussed in Section 19.5.

19.4.3 Efficient algorithms for $\mathcal{NDD}$ constraints and extensions

The papers [97, 6, 100, 83, 98, 7] have tackled the problem of checking the consistency of sets of $\mathcal{NDD}$ constraints efficiently. As it is explained nicely in [7], all these works propose algorithms that consist of the following basic steps:

- *Generation step:* Generate all possible sets of $\mathcal{BD}$ constraints that satisfy the disjunctions.
- *Consistency checking step:* Check consistency of these sets.

The papers [97, 100, 83] do the generation step by solving a meta-CSP with variables corresponding to disjunctions and values corresponding to $\mathcal{BD}$ disjuncts. The papers [6, 98, 7] do the generation step by solving the corresponding propositional satisfiability problem (where $\mathcal{BD}$ disjuncts are represented by propositional variables). The consistency checking step in both cases is carried out using various incremental algorithms for $\mathcal{BD}$ constraints e.g., incremental directional path consistency [24] or incremental full path consistency [77].

Stergiou and Koubarakis [97] were the first to discuss various backtracking algorithms (chronological backtracking, backjumping and forward checking with backjumping) and related heuristics for $\mathcal{NDD}$ constraints. [97] presents theoretical results that characterize these algorithms in terms of number of search tree nodes visited and consistency checks performed by extending [54] where backtracking algorithms for binary CSPs are compared ($\mathcal{NDD}$ constraints are n-ary). [97] also evaluate the performance of their algorithms experimentally using randomly generated hard problems.

Armando et al. [6] subsequently showed how to improve the results of [97] by an algorithm, called TSAT, which is built on top of a SAT solver that implements the Davis-Putnam procedure efficiently. The SAT solver produces the sets of $\mathcal{BD}$ constraints to be checked for consistency. In addition, TSAT has a preprocessing step that produces a more accurate SAT encoding than the obvious one, and a constraint propagation step as in the forward checking algorithm of [97].

[83] presents CSPi, an extension of the forward checking algorithm of [97] with a semantic branching step and a heuristic method for reducing the number of forward checks performed. The semantic branching step, which is available for free in SAT methods such as [6], is as follows. If the current valuation (set of $\mathcal{BD}$ disjuncts) $\{c_1, c_2, \ldots, c_i\}$ cannot be extended by another disjunct c_{i+1} so that we reach a satisfying valuation, then CSPi adds $\neg c_i$ to the current valuation and proceeds to choose another literal from the $(i+1)$-th disjunction. [83] shows that CSPi improves on [97] and is competitive with TSAT.

[100] adopts the CSP framework of [97] and improves it by introducing no-good recording as well as the other pruning techniques introduced by earlier literature (a different form of the backjumping used in [97], semantic branching as used in [83] and removal

of subsumed variables as used in [83]). The resulting system, called Epilitis, is shown to dominate all earlier algorithms [97, 6, 83].

[98] was the earliest paper to deal with deciding $\mathcal{BCBD}$ constraints. The approach of [98] is to transform a given $\mathcal{BCBD}$ formula ϕ into a propositional logic formula and then use the SAT solver Chaff [78] to decide it. The transformation involves essentially the following two steps:

- Introduce a new propositional variable for each $\mathcal{BD}$ constraint in ϕ, and transform ϕ into a new propositional logic formula ϕ'.
- Conjoin to ϕ' a new propositional logic formula that encodes transitive relations among variables derived from the original $\mathcal{BD}$ constraints.

Finally, [7] presents the system TSAT++ which is able to deal with Boolean combinations of difference constraints using a SAT-based approach (in particular, the SIMO solver [46]) and a powerful combination of preprocessing, constraint propagation, branching and intelligent backtracking techniques. [7] demonstrates that TSAT++ is more efficient than the systems of [97, 6, 100, 83, 98] presented above, but also MathSAT [8] which is able to deal with Boolean combinations of linear constraints. The performance analysis of [7] is based on randomly generated hard problems and instances of real-world applications.

Recently, Schwartz and Pollack studied incremental algorithms for $\mathcal{NDD}$ constraints [93]. They consider three update operations (tightening the bound of a $\mathcal{BD}$ constraint, add a $\mathcal{BD}$ constraint or add an $\mathcal{NDD}$ constraint) and present incremental algorithms to handle these updates using techniques from dynamic CSPs such as no-good recording and oracles.

Finally, [76] shows how to solve $\mathcal{NDD}$ constraints using local search. Contrary to earlier complete algorithms using a meta-CSP approach [97, 100, 83], the algorithm of [76] searches over the space defined by the original CSP using an algorithm which derives from GSAT [94] and Tabu search [47].

19.5 More Expressive Queries for Temporal CSPs

When constraint networks are used to represent temporal information (see Section 19.3), 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 $walk$ in Example 19.5) and arcs by temporal relations, temporal constraint networks can be queried in useful ways. The typical query targeted by most of the algorithms discussed in Sections 19.3 and 19.4 is: "What is the strictest temporal relationship between intervals (or points) A and B?". This query is typically answered by consulting the minimal network corresponding to the given temporal constraints.

Van Beek [102] was the first to consider more expressive queries for databases with temporal constraints. In [102], a database is a set of $\mathcal{IA}$ constraints among appropriately named interval constants (representing *events*). The first class of queries considered by [102] is *modal* (possibility or certainty) queries. 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 Boolean combination of $\mathcal{IA}$ constraints that use event constants $e_1, \ldots, e_n$. As an example, consider the query "Is it possible that event $walk$ happened after event $breakfast$?".

The second class of queries considered by [102] is aggregation queries. An *aggregation query* 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 database, OP is the modal operator $\Diamond$ or $\Box$, and ϕ is a Boolean combination of $\mathcal{IA}$ constraints that use variables $x_1, \ldots, x_n$. As an example, consider the query "What are the known events that come after event $breakfast$?".

The temporal reasoning system LATER [18, 28] is another proposal for querying temporal CSPs in sophisticated ways. LATER allows users to define symbolic time points and time intervals and assert temporal constraints relating them with other symbolic objects, or time constants representing conventional *dates, times* and *durations*. LATER offers a practical temporal reasoning framework that includes vocabulary for expressing many useful qualitative and metric temporal constraints. Only certain kinds of disjunctive relations are allowed so that the expressive power of LATER does not become greater than the expressive power of $\mathcal{BD}$ constraints [19]. The complete set of LATER functions and predicates can be found in [18].

The following types of queries are supported by LATER [18]:

1. Queries *extracting temporal information* (e.g., when, how long, duration and relation queries).

2. Modal queries as in [102].

3. *Hypothetical queries.* These queries allow one to query the database using queries of types 1 and 2 under the assumption that certain additional temporal constraints hold.

Although [102] and LATER offer expressive languages for querying databases of temporal constraints, queries combining *non-temporal* as well as temporal information (e.g., "*Who* is certainly having breakfast before taking a walk?") cannot be asked in these systems, even though the knowledge required to answer them might have been available in the first place. This problem arises because temporal CSPs 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 general-purpose data and knowledge representation frameworks have been proposed (and in most cases implemented). These proposals include EPILOG[2], Shocker[3], TMM [31], Telos [79], and the relational temporal constraint databases of [55] and [20]. EPILOG and Shocker use the temporal reasoners Timegraph I and II, Telos uses a subclass of $\mathcal{IA}$, TMM uses $\mathcal{BD}$ constraints, the proposal of [55] uses $\mathcal{BD}$ constraints and the system of [20] uses LATER.

In the rest of this chapter, we study the *scheme of indefinite constraint databases* proposed by Koubarakis [58, 63], as the formalism that unifies the proposals of [102, 55,

[2] See `http://www.cs.rochester.edu/research/epilog/`.

[3] See `http://www.cs.rochester.edu/research/cisd/projects/kr-tools/`.

18, 20]. This formalism is a *scheme* because it can be instantiated with various kinds of constraints defined by a first-order language (e.g., temporal, spatial etc. [63]). When the constraints chosen are temporal, the resulting formalism can be used to represent temporal constraints on various temporal objects, and the relational database can be used to store facts referring to these objects.

Sections 19.6 and 19.7 show that in order to be able to answer queries in this scheme, we must be prepared to go from temporal CSPs to *first-order theories of temporal constraints* as studied in [71, 58]. We identify *variable elimination* (and its logical analogue *quantifier elimination*) as the main technical tool needed by the proposed framework. We then show that query evaluation in the proposed formalism can be viewed as quantifier elimination in a first-order language of temporal constraints.

The indefinite constraint database scheme has been presented in the past as a constraint-based extension of the relational data model [58] or as a constraint-based extension of an equivalent subset of first-order logic [63]. We follow the second approach in this chapter using material directly from [63].

19.6 First-Order Temporal Constraint Languages

We start by introducing some concepts useful for the developments in forthcoming sections. We will deal with many-sorted first-order languages [40]. 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.

Ladkin [71] and Koubarakis [58, 63] have defined various first-order temporal constraint languages where the atomic formulas come from the temporal CSP frameworks defined in Section 19.3. As an example, we define below the first-order languages PA, IA and LIN that are based on the classes of $\mathcal{PA}, \mathcal{IA}$ and $\mathcal{LIN}$ constraints respectively.[4]

19.6.1 The languages PA and IA

The language PA is a simple first-order language that we can use for talking about points in time. 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 $\mathbb{Q}$ 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 $\neq$. We define *PA-constraints* to be exactly the constraints of the class $\mathcal{PA}$.

In a similar way, we can define the first-order language IA which has as atomic formulas the interval constraints expressible in the class $\mathcal{IA}$ (see [60, 61] for a precise definition).

[4] We use the calligraphic type style to write classes of constraints and italic type style to write the corresponding first-order language.

19.6.2 The language LIN

The language LIN is the first order language of linear constraints. The logical symbols of LIN include: parentheses, a countably infinite set of variables, the equality symbol $=$ and the standard sentential connectives. The non-logical symbols of 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 $\mathbb{Q}$ as its domain. $\mathcal{M}_{LIN}$ assigns to each constant symbol an element of $\mathbb{Q}$, 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 $\mathbb{Q}$. We define LIN-*constraints* to be the constraints of the class $\mathcal{LIN}$.

19.6.3 Quantifier and Variable Elimination

In this section we define the operations of quantifier and variable elimination. Quantifier elimination is an operation from mathematical logic [40]. Variable elimination is an algebraic operation [90]. As we will see below, quantifier elimination algorithms utilize variable elimination algorithms as subroutines. In the scheme of indefinite constraint databases to be introduced in Section 19.7, the operation of quantifier elimination is very useful because it can be used for query evaluation. [35] discuss variable elimination and related concepts for arbitrary CSPs.

Definition 19.12. *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 [40]. 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 [40]:

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 19.13. *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 LIN-constraints defined above variable elimination can be performed using Fourier's algorithm. Fourier's algorithm can be summarized as follows [90]. 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 19.14. Let us consider the following set of LIN-constraints:

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

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

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

The following theorem will be useful below. The result for PA and IA are due to [71].

Theorem 19.15. *The theories $Th(\mathcal{M}_{PA})$, $Th(\mathcal{M}_{IA})$ and $Th(\mathcal{M}_{LIN})$ admit quantifier elimination.*

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

19.7 The Scheme of Indefinite Constraint Databases

In this section, we present the scheme of indefinite constraint databases originally proposed in [58]. 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 19.6.3 has defined this concept precisely). For the purposes of this section, $\mathcal{L}$ can be a language like PA, IA and LIN that can be used to talk about temporal objects (i.e., points or intervals).

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 occurrence (e.g., after) and, finally, other information about real-world entities (e.g., names of persons). Temporal CSPs as discussed in Section 19.3 can be used to represent such information.

In the scheme of indefinite constraint databases (and in similar formalisms like [31, 18]) information like the above is represented by utilizing 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.

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

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

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

The following lemma is straightforward.

Lemma 19.16. *If theory $Th(\mathcal{M}_{\mathcal{L}})$ admits quantifier elimination then the same holds for $Th(\mathcal{M}_{\mathcal{L} \cup \mathcal{EQ}})$.*

Finally, we define a new first-order language $(\mathcal{L} \cup \mathcal{EQ})^*$ by augmenting $\mathcal{L} \cup \mathcal{EQ}$ with a countably infinite set of *database predicate symbols* $p_1, p_2, \ldots$ of various arities. These predicate symbols can be used to represent information about our application domain. The arguments of these predicates will be constants and variables constrained by formulas of $\mathcal{L} \cup \mathcal{EQ}$. The indefinite constraint databases and queries defined below are formulas of $(\mathcal{L} \cup \mathcal{EQ})^*$.

In the following example and all the examples of subsequent sections, we assume $\mathcal{L}$ to be the language LIN defined in Section 19.6. The language $LIN \cup \mathcal{EQ}$ is now multi-sorted with sorts $\mathcal{D}$ (for the constants of $\mathcal{EQ}$) and $\mathcal{Q}$ (for the rational constants of LIN).

Example 19.17. 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:

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

19.7.2 Databases And Queries

In this section, the symbols $\bar{x}, \bar{y}, \bar{x}_i, \bar{y}_i$, etc. will denote vectors of variables while $\bar{\omega}$ will stand for a vector of Skolem constants. In addition, 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 [63].

Definition 19.18. *An* indefinite constraint database *is a formula $DB(\bar{\omega})$ of $(\mathcal{L} \cup \mathcal{EQ})^*$ of the following form:*

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

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 i.e., a CSP. $\bar{\omega}$ is a vector of *Skolem constants* denoting time entities (e.g., points and intervals) about which *only partial knowledge* is available. This partial knowledge has been coded in the constraint store as a CSP 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 [85].

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 CSP, while the predicates p_i will encode, in a flexible way, the events or facts usually associated with the variables of this CSP.

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 section.

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 [52], and the marked null values proposal of [48, 1]. Similar ideas can also be found in the first-order databases of [85].

Let us now give some examples of indefinite constraint databases. The constraint language used is LIN but the constraints are simpler than full linear: rational order constraints, difference constraints and bounds on variables.

Example 19.19. The following is an indefinite constraint database which formalises the information in the paragraph considered at the beginning of this section.

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

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

Example 19.20. Let us consider the following planning database used by a medical laboratory for keeping track of patient appointments for the year 2006.

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

Since we use LIN, the set of rationals $\mathbb{Q}$ is our time line. The year 2006 is assumed to start at time 0 and every interval $[i, i + 1)$ represents a day (for $i \in \mathbb{Z}$ 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 2006 (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 2006 (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 CSPs discussed in Section 19.5 above, and it is similar to the concept of query in TMM [31].

Definition 19.21. *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 queries ($\Box$) and possibility queries ($\Diamond$) as in [102].

Example 19.22. The following query refers to the database of Example 19.19 and asks "Who was the person who possibly had a conversation with Fred during this person's walk in the park?":

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

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. Such a query language can be interesting in a formal framework dealing with indefinite information, but we will not consider this issue further in this chapter.

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

Definition 19.23. *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 19.24. The answer to the query of Example 19.22 is $(x = Mary,\ \emptyset)$.

The definition of answer in the case of certainty queries is the same as Definition 19.23 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 19.25. *A query is called* closed *or* yes/no *if it does not have any free variables. Queries with free variables are called* open.

Example 19.26. The query of Example 19.22 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 19.27. The answer to the query of Example 19.26 is $(true,\emptyset)$ i.e., yes.

Example 19.28. Let us consider the database of Example 19.20 and the query "Find all appointments for patients that can possibly start at the 92th day of 2006". This query can be expressed as follows:

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

The answer to this query is the following:

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

19.7.3 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 [52]). The following theorem is essentially from [58] and [63].

Theorem 19.29. *Let DB be the indefinite constraint database*

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

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 $\mathcal{L}\cup\mathcal{EQ}$:

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

In this formula the vector of Skolem constants $\bar{\omega}$ has been substituted by a vector of appropriately quantified variables with the same name ($\bar{\mathcal{D}}'$ 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

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

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

Example 19.30. Using the above theorem, the query of Example 19.22 can be answered by eliminating quantifiers from the formula:

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

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

Koubarakis and Skiadopoulos [58, 63] have studied the complexity of query answering in the scheme of indefinite constraint databases for various temporal and spatial constraint languages $\mathcal{L}$. Their results precisely outline the frontier between tractable and possibly intractable query answering problem. [63] shows that if one wants to be able to answer modal queries in PTIME, it is no longer sufficient to have a constraint class (e.g., $\mathcal{BD}$) with PTIME reasoning problems (e.g., consistency checking for $\mathcal{BD}$ can be done in $O(n^3)$ time); further conditions should be imposed on queries and databases.

19.8 Conclusions

We have surveyed work on temporal CSPs starting from early papers such as [3, 107, 31, 108, 34] and continuing with influential contributions that have been published as recently as last year. There are certain topics of work in temporal CSPs that we did not cover due to limited space. These include:

- Temporal CSPs for non-totally-ordered time e.g., partially ordered time, branching time etc. [16].
- Representing *periodic* temporal information by constraints [51].
- *Non-convex* intervals and their CSPs [9].
- *Soft constraints* or *preferences* in temporal CSPs [67].
- *Overconstrained* temporal CSPs [73].
- Connections with *spatial* CSPs [87].

We expect research on temporal CSPs to continue healthily in the years to come due to their importance in applications. In our opinion, the following topics are likely to be in the front line of future developments:

- New algorithmic techniques for temporal constraint solving e.g., randomized algorithms [68] or local search [76, 11].
- Theory and algorithms for combining temporal CSPs and optimization concepts [67, 73].
- Theory and algorithms for quantified formulas with temporal constraints [71, 58, 63].
- Tractability results for the classes where this question has not been answered completely e.g., [75].
- Integration with spatial CSPs to deal with spatio-temporal scenarios [62].

Acknowledgements

I would like to thank Peter van Beek, Kostas Stergiou, Spiros Skiadopoulos, Peter Jonsson and Berthe Choueiry for comments on various versions of this chapter.

19.9 Bibliography

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

[2] S. Adali, L. Console, M. L. Sapino, M. Schenone, and P. Terenziani. Representing and reasoning with temporal constraints in multimedia presentations. In *TIME*, pages 3–12, 2000.

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

[4] J. Allen, H. Kautz, and R. Pelavin, editors. *Reasoning About Plans*. Morgan-Kaufmann, 1991.

[5] R. Alur. Timed automata. In *CAV*, pages 8–22, 1999.

[6] A. Armando, C. Castellini, and E. Giunchiglia. SAT-based procedures for temporal reasoning. In *ECP*, pages 97–108, 1999.

[7] A. Armando, C. Castellini, E. Giunchiglia, and M. Maratea. A SAT-based decision procedure for the Boolean combination of difference constraints. In *SAT*, 2004.

[8] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani. A SAT based approach for solving formulas over Boolean and linear mathematical propositions. In *CADE*, pages 195–210, 2002.

[9] P. Balbiani, J.-F. Condotta, and G. Ligozat. Reasoning about generalized intervals: Horn representability and tractability. In *TIME*, pages 23–30, 2000.

[10] P. Balbiani, J.-F. Condotta, and G. Ligozat. On the consistency problem for the INDU calculus. In *TIME*, pages 203–211, 2003.

[11] M. Beaumont, J. Thornton, A. Sattar, and M. J. Maher. Solving over-constrained temporal reasoning problems using local search. In *PRICAI*, pages 134–143, 2004.

[12] C. Bessière, A. Isli, and G. Ligozat. Global consistency in Interval Algebra networks: Tractable subclasses. In *ECAI*, pages 3–7, 1996.

[13] C. Bettini, X. S. Wang, and S. Jajodia. Solving multi-granularity temporal constraint networks. *Artificial Intelligence*, 140(1/2):107–152, 2002.

[14] C. Bliek and D. Sam-Haroud. Path consistency on triangulated constraint graphs. In *IJCAI*, pages 456–461, 1999.

[15] M. Broxvall. A method for metric temporal reasoning. In *AAAI/IAAI*, pages 513–518, 2002.

[16] M. Broxvall and P. Jonsson. Point algebras for temporal reasoning: Algorithms and complexity. *Artificial Intelligence*, 149(2):179–220, 2003.

[17] M. Broxvall, P. Jonsson, and J. Renz. Disjunctions, independence, refinements. *Artificial Intelligence*, 140(1/2):153–173, 2002.

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

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

[20] 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*, 1(6):948–968, 1999.

[21] R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks. In *AIPS*, pages 13–18, 1994.

[22] A. Cesta and A. Oddi. Gaining efficiency and flexibility in the simple temporal problem. In *TIME*, 1996.

[23] A. Cesta, A. Oddi, and S. F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136, 2002.

[24] N. Chleq. Efficient algorithms for networks of quantitative temporal constraints. In *Proceedings of CONSTRAINTS-95*, pages 40–45, Melbourne Beach, Florida, USA, April 1995.

[25] B. Y. Choueiry and L. Xu. An efficient consistency algorithm for the temporal constraint satisfaction problem. *AI Communications*, 17(4):213–221, 2004.

[26] D. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis. Building tractable disjunctive constraints. *Journal of the ACM*, 47(5):826–853, 2000.

[27] D. A. Cohen, P. Jeavons, and M. Koubarakis. Tractable disjunctive constraints. In *CP*, pages 478–490, 1997.

[28] L. Console and P. Terenziani. Efficient processing of queries and assertions about qualitative and quantitative temporal constraints. *Computational Intelligence*, 15 (4):442–465, 1999.

[29] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[30] T. Dean. Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

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

[32] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[33] R. Dechter, I. Meiri, and J. Pearl. Temporal Constraint Networks. In *KR*, pages

83–93, 1989.

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

[35] R. Dechter and P. van Beek. Local and global relational consistency. *Theoretical Computer Science*, 173(1):283–308, 1997.

[36] J. P. Delgrande and A. Gupta. Updating $<=$, $<$-chains. *Information Processing Letters*, 82(5):261–268, 2002.

[37] J. P. Delgrande, A. Gupta, and T. V. Allen. A comparison of point-based approaches to qualitative temporal reasoning. *Artificial Intelligence*, 131(1-2):135–170, 2001.

[38] T. Drakengren and P. Jonsson. A complete classification of tractability in allen's algebra relative to subsets of basic relations. *Artificial Intelligence*, 106(2):205–219, 1998.

[39] C. Dyreson and R. Snodgrass. Valid-time Indeterminacy. In *ICDE*, pages 335–343, 1993.

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

[41] M. Fisher, D. Gabbay, and L. Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[42] A. Gerevini. Incremental qualitative temporal reasoning: Algorithms for the Point Algebra and the ORD-Horn class. *Artificial Intelligence*, 166(1-2):37–80, 2005.

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

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

[45] M. Ghallab and M. Alaoui. Managing Efficiently Temporal Relations through Indexed Spanning Trees. In *IJCAI*, pages 1297–1303, 1989.

[46] E. Giunchiglia, M. Maratea, and A. Tacchella. Look-ahead vs. look-back techniques in a modern SAT solver. In *SAT*, 2003.

[47] F. Glover and M. Laguna. *Tabu Search*. Dordrecht, 1997.

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

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

[50] P. Jonsson and A. A. Krokhin. Complexity classification in qualitative temporal constraint reasoning. *Artificial Intelligence*, 160(1-2):35–51, 2004.

[51] F. Kabanza, J.-M. Stevenne, and P. Wolper. Handling Infinite Temporal Data. *Journal of Computer and System Sciences*, 51(1):3–17, 1995.

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

[53] H. Kautz and P. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *AAAI*, pages 241–246, 1991.

[54] G. Kondrak and P. van Beek. A theoretical evaluation of selected backtracking algorithms. *Artificial Intelligence*, 89(1-2):365–387, 1997.

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

[56] M. Koubarakis. Tractable Disjunctions of Linear Constraints. In *CP*, pages 297–

307, Boston, MA, August 1996.

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

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

[59] M. Koubarakis. Tractable disjunctions of linear constraints: basic results and applications to temporal reasoning. *Theoretical Computer Science*, 266(1-2):311–339, 2001.

[60] M. Koubarakis. Querying temporal constraint networks: A unifying approach. *Applied Intelligence*, 17(3):297–311, 2002.

[61] M. Koubarakis. Indefinite temporal databases with temporal information: Representational power and computational complexity. In M. Fisher, D. Gabbay, and L. Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[62] M. Koubarakis, T. K. Sellis, A. U. Frank, S. Grumbach, R. H. Güting, C. S. Jensen, N. A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H.-J. Schek, M. Scholl, B. Theodoulidis, and N. Tryfona, editors. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *Lecture Notes in Computer Science*, 2003. Springer.

[63] M. Koubarakis and S. Skiadopoulos. Querying Temporal and Spatial Constraint Networks in PTIME. *Artificial Intelligence*, 123(1-2):223–263, 2000.

[64] R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation Computing*, 1(4):67–95, 1986.

[65] A. A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, 50(5): 591–640, 2003.

[66] A. A. Krokhin, P. Jeavons, and P. Jonsson. Constraint satisfaction problems on intervals and lengths. *SIAM Journal on Discrete Mathematics*, 17(3):453–477, 2004.

[67] T. K. S. Kumar. A polynomial-time algorithm for simple temporal problems with piecewise constant domain preference functions. In *AAAI*, pages 67–72, 2004.

[68] T. K. S. Kumar. On the tractability of restricted disjunctive temporal problems. In *ICAPS*, pages 110–119, 2005.

[69] P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *IJCAI*, pages 1643–1649, 1995.

[70] P. Ladkin. Primitives and Units for Time Specification. In *AAAI*, pages 354–359, 1986.

[71] P. Ladkin. Satisfying First-Order Constraints About Time Intervals. In *AAAI*, pages 512–517, 1988.

[72] P. B. Ladkin and A. Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, 1992.

[73] M. H. Liffiton, M. D. Moffitt, M. E. Pollack, and K. A. Sakallah. Identifying conflicts in overconstrained temporal problems. In *IJCAI*, pages 205–211, 2005.

[74] J. McCarthy and P. J. Hayes. Some Philosophical Problems From the Standpoint of Artificial Intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, pages 463–502. Edinburg University Press, 1969.

[75] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning.

Artificial Intelligence, 87(1-2):343–385, 1996.

[76] M. D. Moffitt and M. E. Pollack. Applying local search to disjunctive temporal problems. In *IJCAI*, pages 242–247, 2005.

[77] R. Mohr and T. C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28(2):225–233, 1986.

[78] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *39th Design Automation Conference (DAC)*, 2001.

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

[80] I. Navarrete, A. Sattar, R. Wetprasit, and R. Marín. On point-duration networks for temporal reasoning. *Artificial Intelligence*, 140(1/2):39–70, 2002.

[81] B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints*, 1(3):175–190, 1997.

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

[83] A. Oddi and A. Cesta. Incremental forward checking for the disjunctive temporal problem. In *ECAI*, pages 108–112, 2000.

[84] A. K. Pujari, G. V. Kumari, and A. Sattar. INDU: An interval and duration network. In *Australian Joint Conference on Artificial Intelligence*, pages 291–303, 1999.

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

[86] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[87] J. Renz. A Spatial Odyssey of the Interval Algebra: 1. Directed Intervals. In *IJCAI*, pages 51–56, 2001.

[88] J. Renz and G. Ligozat. Weak composition for qualitative spatial and temporal reasoning. In *CP*, pages 534–548, 2005.

[89] P. Revesz. *Introduction to Constraint Databases*. Springer, 2002.

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

[91] L. Schubert and C. Hwang. Episodic logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding. In L. Iwanska and S. Shapiro, editors, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pages 111–174. MIT/AAAI Press, 2000.

[92] E. Schwalb and R. Dechter. Processing disjunctions in temporal constraint networks. *Artificial Intelligence*, 93:29–61, 1997.

[93] P. Schwartz and M. E. Pollack. Two approaches to semi-dynamic disjunctive temporal problems. In *ICAPS Workshop on Constraint Programming for Planning and Scheduling*, 2005.

[94] B. Selman, H. J. Levesque, and D. G. Mitchell. A new method for solving hard satisfiability problems. In *AAAI*, pages 440–446, 1992.

[95] Y. Shi, A. Lal, and B. Y. Choueiry. Evaluating consistency algorithms for temporal metric constraints. In *AAAI*, pages 970–971, 2004.

[96] S. Staab. From binary temporal relations to non-binary ones and back. *Artificial Intelligence*, 128(1-2):1–29, 2001.

[97] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence*, 120(1):81–117, 2000.

[98] O. Strichman, S. A. Seshia, and R. E. Bryant. Deciding separation formulas with SAT. In *CAV*, pages 209–222, 2002.

[99] J. Thornton, M. Beaumont, A. Sattar, and M. J. Maher. A local search approach to modelling and solving Interval Algebra problems. *Journal of Logic and Computation*, 14(1):93–112, 2004.

[100] I. Tsamardinos and M. E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151(1-2):43–89, 2003.

[101] P. van Beek. Approximation Algorithms for Temporal Reasoning. In *IJCAI*, pages 1291–1296, 1989.

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

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

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

[105] P. van Beek and D. W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.

[106] J. van Benthem. *The Logic of Time*. D. Reidel Publishing Company, 1983.

[107] M. Vilain and H. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *AAAI*, pages 377–382, 1986.

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

[109] M. B. Vilain. A system for reasoning about time. In *AAAI*, pages 197–201, 1982.

[110] L. Xu and B. Y. Choueiry. Improving backtrack search for solving the TCSP. In *CP*, pages 754–768, 2003.

[111] L. Xu and B. Y. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *TIME*, pages 212–222, 2003.

Index