Reasoning About Time

MARTIN CHARLES GOLUMBIC
BAR-ILAN UNIVERSITY, RAMAT-GAN, ISRAEL
golumbic@cs.biu.ac.il

1. Introduction

Reasoning about time is essential for applications in artificial intelligence and in many other disciplines. Given certain explicit relationships between a set of events, we would like to have the ability to infer additional relationships which are implicit in those given. For example, the transitivity of "before" and "contains" may allow us to infer information regarding the sequence of events. Such inferences are essential in story understanding, planning and causal reasoning. There are a great number of practical problems in which one is interested in constructing a time line where each particular event or phenomenon corresponds to an interval representing its duration. These include seriation in archeology, behavioral psychology, temporal reasoning, scheduling, and combinatorics. Other applications arise in non-temporal contexts, for example, in molecular biology, where arrangement of DNA segments along a linear DNA chain involves similar problems.

A system to reason about time should have a knowledge base consisting of temporal and other information, a mechanism for determining consistency of temporal data, and routines for answering queries and discovering or inferring new facts. We will present several formal models to represent temporal knowledge which can be used in reasoning systems. Our interests are in the mathematical and computation properties of these models and the algorithms which can be applied to effectively solve temporal problems.

Example 1. *Fred was reading the Orlando Sentinel* while *eating breakfast in the hotel restaurant. Shortly* before he finished *Marty joined him. Fred put the paper down, and they discussed the plans for publishing the volume of papers from the AMS short course. The waitress took Marty's order. Glenn and Vic's papers* were taken care of *but* before *deciding who should referee Catherine's paper,* they saw *Helene enter the restaurant, and Fred asked her to join the discussion. He wondered if the others would be coming too, but Helene said* she saw *Monty sitting in the lobby at* 7 o'clock *when she went out jogging. He had eaten breakfast at* 6 am*, the first customer of the day. No one had heard from Jean-Louis, but he rarely eats breakfast. The three of them made the* final *decisions. It was a* half hour before *the first morning lectures* *would* begin. *Fred drank the last of his coffee and left to go for a walk* just as *Marty's bagel and lox was being served. Helene filled her plate at the buffet and saw Fred at the phone, probably calling Jean-Louis to be sure he was awake. (He was, and he had eaten breakfast with Monty!) At* 9 am *the lectures* started, *and everyone was* present. *That is, everyone except Catherine, Glenn and Vic, who were* still *snowed in up North.*

This example illustrates many of the issues that arise in temporal reasoning. First, there is the *chronology* of the story which takes place in the restaurant during a certain "window of time": Fred is eating, Marty enters, they discuss, Helene enters, they conclude the discussion, Fred leaves and Marty is served, Helene fills her plate and Fred phones, lectures start, etc. The narrative follows a time line reporting actions in the order in which they happen. Second, the story provides *temporal information* on other actions which have occurred outside its "window of time": Monty ate at 6 am, was in the lobby at 7am, and Jean-Louis was with him at breakfast. Third, the story gives *temporal clues* through the use of words and expressions: just as, before finishing, 7 o'clock, they saw, everyone was present, half hour before, etc.

All of these allow one to reconstruct a portion of a temporal scenario, that is, what happened, when or in what order, and by whom. It does not help us determine if Jean-Louis was preparing transparencies for his talk or sleeping or working on some research with Monty between breakfast and the lectures, but we can determine that he had no input on who would referee Catherine's paper. Detectives and puzzle enthusiasts, consultants and strategic planners, chefs and building contractors, politicians and college students all reason about time in this way.

There are at least three major approaches to reasoning about time. The first is *quantitative* where we treat the duration of an event in a numerical fashion. The temporal range of the starting point and finishing point, the length of time between disjoint events, and so forth, are numbers which may satisfy specified inequalities or various measurable constraints. Methods used in the quantitative approach may be based on constraint propagation, heuristic search, mathematical programming, and other techniques from artificial intelligence and operations research. The second is *qualititative* where events are represented by abstract time points and time intervals, and we process and deduce "relationships" between them, such as pairs intersecting each other, one preceding, following or containing another, etc. The techniques used in the qualitative approach often involve combinatorial and graph theoretic methods. The third is *logical* where the truth of predicates and propositions vary with time. This approach gives rise to a host of temporal, non-monotonic logics for reasoning about world states which change over time. We can think of other approaches as well, such as statistical or probabilistic temporal reasoning.

It is our belief that no single approach will ever be adequate to handle the diversity of all forms of temporal reasoning problems. Neither will an integrated solution method ever satisfy the virtually unbounded complexity that temporal situations can exhibit. Nevertheless, major strides have been made, and will continue to be made, to handle larger classes of useful problems and to give effective and efficient solutions in important application areas.

Mathematical and computational properties of the time line are intrinsically interwoven. We may not want to rely upon a program for updating and maintaining a database of financial transactions without providing a provably consistent representation for it at any point of time. We may find that the behavior of a hardware circuit depends on timing properties which can be optimized mathematically. Coordinating independent, distributed systems in a changing environment may call for communication protocols satisfying temporal conditions. Underlying mathematical structure of time intervals may give rise to algorithms for scheduling a sequence of actions to achieve a set of goals. The classical Yale Shooting Problem, to determine whether a gun is loaded at time T_n, given the fact that it was loaded at T_0 and a sequence of events (such as firing, reloading, jamming, various uncertainties, etc.), could mean the difference between life and death.

In this article, we provide background reading material for entering the field of temporal reasoning. It is not a complete survey, and it is biased by the author's particular interests. However, it should be a good (qualitative) starting point for anyone new to the field, and may even have some continuing value for the more familiar reader.

1.1. Events in Time: Points and Intervals. An *event* in time may be thought of as an interval $[s, t]$ on the time line where s is its starting endpoint and t is its terminal or finishing endpoint and having a *duration* given by $t - s \geq 0$. The special case where $s = t$ is called a *point event* since it has no duration, and otherwise it will be called an *interval event*.

Explosions, birth, and burial are typical examples of point events, at least from the point of view of an historian; however, they might be treated as interval events by the physicist, midwife or grave digger. Therefore, the granularity of the time line will depend on the context of the application. We will usually treat time as being linearly ordered discrete units, such as minutes, seconds, years, hours, millenia, nanoseconds, etc. Although we may use closed, half open or open intervals to represent temporal events, we generally use closed intervals.

Point events are easier to deal with than interval events, but the latter are more interesting. Consider the following statements:

S_1: Paul dived into the swimming pool before Ringo.

S_2: John finished swimming laps while George was still swimming laps.

We think of diving as a point event, so S_1 has the meaning $dive(Paul) < dive(Ringo)$. But swimming laps has a duration, so while S_2 tells us that $start\_laps(George) < finish\_laps(John) < finish\_laps(George)$, we have no information about who started swimming laps first.

With interval events we have twice as many points to reason about. But that is *not* the reason for the higher complexity. When a statement relates two point events, it is a binary relation. When a statement relates two interval events, it can be a 4-way constraint. For example, S_2 gave us two binary constraints, but S_3 (below) gives a relation among 4 points which cannot be expressed using simple binary relations.

> S_3: Carole and Carly each drove the school bus around town for a few hours today.

The relationship in S_3 requires the 4-way constraint:

$$finish\_bus(Carole) < start\_bus(Carly) \vee finish\_bus(Carly) < start\_bus(Carole)$$

since a bus can have only one driver at a time.

We shall see later that as we add more expressiveness to our relations, we pay for it by increased computational complexity.

1.2. Quantitative Approaches. In Section 2 we will present a model for metric temporal constraint problems (MTCPs) consisting of variables representing time points and sets of inequalities on their differences. Example 2 tells a story about Allan and Judy and Figure 1.1 shows the corresponding temporal constraint network whose edges are labelled by the intervals of possible values for the pairwise differences. Thus, an edge $i \to j$ labelled by $[a, b]$ represents the inequality $a \le x_j - x_i \le b$. It is worthwhile noting that two of the edges in the network are labelled by more than one interval indicating a disjunction, that is, the difference for such a pair can be in either interval. As an exercise, the reader may analyse the temporal constraints, ad hoc, and find a consistent solution, i.e., values assigned to the variables x_i such that all the inequalities are satisfied. A formal technique will be given in Section 2.

Example 2. *Allan spent two hours in the library solving the exercises at the end of his chapter. After finishing, he planned to go swimming at the pool for at least 45 minutes but no more than one hour. It is 25 minutes walking from the library to the pool, but by bus it takes only 5 to 15 minutes. Afterwards, he would meet Judy at her house and go to a party. Since Allan lives next door to the pool, he can go to her house by bicycle which is more fun, and takes 45 to 55 minutes. Going by taxi is more expensive, but faster, no more than 10 minutes. Judy had called Allan on his cellular telephone to make the arrangements, and he told her that he was already* *in the library solving the exercises for the past 45 minutes. She told him that she was catching the next train which would get her home in one hour. Then she would change clothes and watch a video movie taking 135 to 140 minutes altogether, but she would wait for him no more than 10 minutes after it finished. Could Allan ride his bicycle to Judy's house and be sure not to miss her?*

The variables and their constraints, as extracted from the story, are the following:

x_0 - Allan started solving the Exercises
x_1 - Allan finished the Exercises

$$120 \leq x_1 - x_0 \leq 120$$

x_2 - Allan started swimming

$$5 \leq x_2 - x_1 \leq 15 \quad \text{or} \quad 25 \leq x_2 - x_1 \leq 25$$

x_3 - Allan finished swimming

$$45 \leq x_3 - x_2 \leq 60$$

x_4 - Allan arrived at Judy's home

$$45 \leq x_4 - x_3 \leq 55 \quad \text{or} \quad 0 \leq x_4 - x_3 \leq 10$$

x_5 - Judy called Allan

$$45 \leq x_5 - x_0 \leq 45$$

x_6 - Judy arrived home

$$60 \leq x_6 - x_5 \leq 60$$

x_7 - Judy finished watching the movies

$$135 \leq x_7 - x_6 \leq 140$$

x_8 - Judy left her house

$$0 \leq x_8 - x_7 \leq 10$$

Allan's goal:

$$0 \leq x_8 - x_4 \leq \infty$$

Remark 1. *Presumably Allan can control the swimming time, but not the bus time. This model does not make such a distinction.*

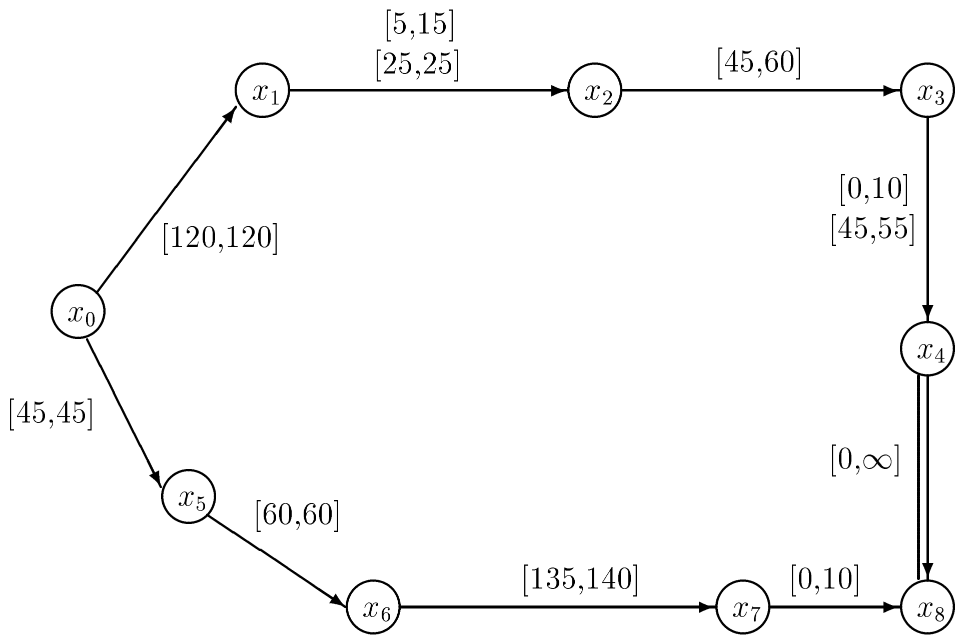


Figure 1.1. The temporal constraint network for Example 2.

1.3. Qualitative Approaches. Some temporal reasoning problems make no mention of numbers, clock times, dates, durations, etc. Rather, only qualitative relations such as *before*, *during*, *after* or *not after* are given between pairs of events. This is often the case in story telling, where we get clues from the natural language text about temporal relationships without specific times stated.

The *seriation problem* asks for a mapping of the events onto the time line such that all the given relations are satisfied, that is, a consistent scenerio. Unlike the metric temporal constraint problems (MTCPs) where numerical methods, including shortest path algorithms and constraint based methods are useful, here a more combinatorial, graph theoretic approach is needed. This reconstruction of the events on the time line from qualitative data is similar to DNA mapping problems in molecular biology where the true DNA sequence of the gene must be built up using overlap data of clone fragments.

One of the techniques most useful with qualitative problems is the propagation of constraints between pairs of events. We will illustrate this on a simple example and postpone a full development of the method to Section 3.

Example 3. *Once upon a time there were four bears, Papa bear, Mama bear, Baby bear and Teddy Bear. Each bear sat at the table to eat his morning porridge, but since there were only two chairs (the third chair was broken in a previous story), the bears had to take turns eating. Baby and Teddy always ate together sharing the* *same chair, and on this day Mama was seated for part of Baby bear's meal when the door opened and in walked their new neighbor, Goldie.*

"What a great aroma," she said. "Can I join for a bowl?" Mama replied, "Sure, but you will have to wait *for a chair!" "Yeah, I know all about chairs," said Goldie. So Goldie sat down when Baby and Teddy got up.*

Papa entered the kitchen. Looking at Mama he said, "I wouldn't sit at the same table with that girl." Mama answered, "Then it's good you ate already."

In this story, we assume that each character sat at the table for an interval of time without interruption. Let's call these intervals I_B, I_G, I_M, I_P, I_T, subscripts indicating the first initial of the character's name. Since Baby and Teddy always eat together, we have $I_B = I_T$, so there are only four intervals that we have to place on the time line.

Here are some questions to ask:

1. Could Papa and Baby both be at the table together?

2. Could Papa and Mama both be at the table together?

3. Could Papa have spent some time at the table with both Baby and Mama?

4. Did anyone sit at the table with Goldie?

Consider the three relations $\prec$, $\cap$, $\succ$ between intervals, where $I \prec J$ (equiv. $J \succ I$) denotes that I and J are disjoint intervals with I occurring before J, and $I \cap J$ denotes that I and J intersect. We can represent the interval events of our story as the vertices of an oriented graph with edges labelled by the possible relations between pairs of events. For example, if $I \prec J$ or $I \cap J$ are both possible, and the edge $e = (I, J)$ is oriented from I to J, then e will be labelled $\prec \cap$. The graph for our example appears in Figure 1.2.

Our story provides the following facts:

(a) Only two chairs (This is not temporal but resource information.)

(b) $I_B \cap I_M$ Mama and Baby seated when the door opened.

(c) $I_B \prec I_G$ Goldie sat down when Baby got up.

(d) $I_P \prec I_G$ Papa ate before Goldie.

(e) $I_M \cap I_G$ Language clue Papa to Mama, "I wouldn't sit ... with that girl."

Although no relation is stated between I_P and I_B or I_M, we could "reason" that $I_P \prec \cap\, I_M$, that is, Papa could not eat *after* Mama since that would contradict the last paragraph of the story. Let's formalize that reasoning process using constraint propagation.

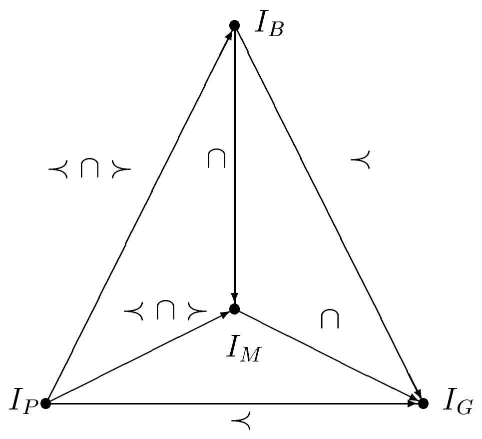


Figure 1.2. The constraint graph before propagation.

The simplest form of propagation is ordinary transitivity, namely if $I \prec J$ and $J \prec K$, then $I \prec K$. But we can define other implications as well between three intervals, for example, $I \prec J$ and $J \cap K$ implies $I \prec \cap K$, which can easily be seen by trying to draw the intervals I, J, K on the time line. This is precisely the rule we used combining facts (d) and (e) to deduce $I_P \prec \cap I_M$. These generalized transitivity rules are given in the multiplication table in Figure 1.3, and illustrates one of the *interval algebras*, namely A_3, which will be studied further in Section 3. This language is limited, as it cannot express the idea that one interval contains another. More complex interval algebras A_6, A_7, A_{13} and ord-Horn allow containment and other primitive relations.

	$\prec$	$\cap$	$\succ$
$\prec$	$\prec$	$\prec \cap$	$\prec \cap \succ$
$\cap$	$\prec \cap$	$\prec \cap \succ$	$\cap \succ$
$\succ$	$\prec \cap \succ$	$\cap \succ$	$\succ$

Figure 1.3. The multiplication table for A_3 constraint propagation.

1.4. Logical Approaches and Related Topics. Logic and logic programming are used widely to express temporal information and to automate reasoning about time. In Section 4, we briefly introduce temporal logic and provide references. Temporal reasoning issues also arise in distributed artificial intelligence with the coordination of multiple agents, and in the area of temporal databases which must maintain a repository of non-static facts and knowledge and answer temporal queries. We provide some motivation and references in Section 5 for these topics.

2. Metric Temporal Constraint Problems - A Qualitative Approach

Constraint networks are frequently used in artificial intelligence to represent sets of values which may be assigned to variables, subject to binary restrictions that may apply between pairs of variables. The temporal constraint networks presented in this section are an important example of this, and we shall use them to formalize methods for solving quantitative problems.

2.1. Metric Temporal Constraint Networks. Let $x_1, x_2, \ldots, x_n$ be variables which represent time *points* and let x_0 be an arbitrary fixed point in time which we call the *origin*. A *constraint* between x_i and x_j consists of a set of inequalities on their difference, namely

$$C_{i,j} = \begin{cases} a_1 \leq x_j - x_i \leq b_1 \\ \vdots \\ a_r \leq x_j - x_i \leq b_r \end{cases}$$

A constraint $C_{i,j}$ is *satisfied* by an assignment $x_i = c_i$ and $x_j = c_j$ if *one* of its inequalities holds for this assignment. The bounds $a_1, b_1, \ldots, a_r, b_r$ may be negative, zero or positive.

A *metric temporal constraint problem* (MTCP) consists of the set of variables $X = \{x_0, x_1, \ldots, x_n\}$ and a constraint set $C_{i,j}$ (possibly empty) for each i and j ($i \neq j$). A *solution* to the MTCP is an assignment of real numbers to each of the variables $x_0, x_1, \ldots, x_n$ such that each of the constraints $C_{i,j}$ is satisfied. Unary constraints, like $a \leq x_i \leq b$, are easily handled by expressing them as binary constraints with the origin, namely, $a \leq x_i - x_0 \leq b$.

Example 4.

$$C_{1,2} = \begin{cases} -3 \leq x_2 - x_1 \leq 2 \\ 10 \leq x_2 - x_1 \leq 12 \end{cases}$$

The first inequality is interpreted as "either x_2 occurs after x_1 with a distance of at most 2 apart or they are equal or x_2 occurs before x_1 at a distance of at most 3 apart." The second inequality adds the additional possibility that "x_2 occurs after x_1 separated by a distance of between 10 and 12." For example, $x_1 = 40$, $x_2 = 60$ does not satisfy $C_{1,2}$, however, $x_1 = 40$, $x_2 = 51$, and $x_1 = 40$, $x_2 = 39$ both satisfy $C_{1,2}$.

Example 5. *By setting the origin $x_0 = 0$ to mean 8:00 am, we can express other time points. For example, $30 \le x_5 - x_0 \le 30$, or more simply $x_5 = 30$, is interpreted as "x_5 occurs at exactly 8:30 am." We can also express unary constraints such as "x_6 occurs between 7:45 am - 8:15 am or between 10:00 am - 10:30 am" by the two inequalities: $-15 \le x_6 - x_0 \le 15$ or $120 \le x_6 - x_0 \le 150$.*

Example 6. *We* cannot *express ternary constraints (on three variables) such as in the statement, "Peter finished 15 minutes after Paul and Mary had both finished", since we don't know which of Paul or Mary finished second.*

Let $D_{i,j}$ denote the union of the constraint intervals for $x_j - x_i$ given by $C_{i,j}$, namely

$$D_{i,j} = [a_1, b_1] \cup \cdots \cup [a_r, b_r]$$

Thus, $D_{i,j}$ represents the *domain* of the difference $x_j - x_i$ over all possible permitted values according to the *input* of the problem. However, in a solution the pair x_i and x_j must satisfy both $x_j - x_i \in D_{i,j}$ and $x_i - x_j \in D_{j,i}$, so that for simplicity we combine $D_{i,j}$ and $D_{j,i}$ as follows:

Noting that $a \le x_j - x_i \le b$ if and only if $-b \le x_i - x_j \le -a$, we define the *reversal* of a constraint set as

$$D_{i,j}^{-1} = [-b_r, -a_r] \cup \cdots \cup [-b_1, -a_1]$$

We define the intersection sets

$$T_{i,j} = D_{i,j} \cap D_{j,i}^{-1} \text{ (for all } i \ne j).$$

It is easy to show that $T_{i,j} = T_{j,i}^{-1}$, so with these sets it is sufficient to have only one of them for each pair (i, j), the other being implicit. Furthermore, one can calculate $T_{i,j}$ from $C_{i,j}$ and $C_{j,i}$ in linear time. Doing this for each pair (i, j), combining intervals that overlap, we construct an equivalent version of the problem which we call its canonical form.

A temporal constraint problem is in *canonical* form if we are given the constraints as the union of disjoint intervals

$$T_{i,j} = [a_1, b_1] \cup \cdots \cup [a_r, b_r]$$

satisfying $T_{i,j} = T_{j,i}^{-1}$. Thus, a tuple $< c_0, c_1, \ldots, c_n >$ is a solution if and only if $c_j - c_i \in T_{i,j}$ for all $i < j$. We call $T_{i,j}$ the *canonical constraint set* for i and j.

Example 7.

$$C_{2,1} = \begin{cases} 2 \le x_1 - x_2 \le 5 \\ -1 \le x_1 - x_2 \le 0 \\ -6 \le x_1 - x_2 \le -3 \\ -11 \le x_1 - x_2 \le -11 \end{cases}$$

Combining $C_{2,1}$ with $C_{1,2}$ from Example 4, we have

$$D_{1,2} = [-3, 2] \cup [10, 12]$$
$$D_{2,1}^{-1} = [-5, -2] \cup [0, 1] \cup [3, 6] \cup [11, 11]$$

whose intersection gives the canonical form

$$T_{1,2} = [-3, -2] \cup [0, 1] \cup [11, 11]$$

Thus, $x_1 = 40$, $x_2 = 51$ is still a possible solution, but $x_1 = 40$, $x_2 = 39$ is no longer possible.

Temporal constraint problems are often represented in a graphical form called a *temporal constraint network*. The network has vertices corresponding to the variables $x_0, x_1, \ldots, x_n$ and a directed arc from i to j labelled by its constraint set $D_{i,j}$. If there is no constraint for a pair, the arc is usually omitted.[1] The *canonical network* labels arc (i, j) with the set $T_{i,j}$ and omits the arc (j, i). We denote by $T_e = \{I_e^1, \ldots, I_e^r\}$ the set of disjoint intervals labeling the arc $e = (i, j)$. From our experience, we always recommend using the canonical network.

In general, metric temporal constraint problems are intractible, that is, their computational complexity is NP-hard, as is the case with all general forms of constraint satisfiability problems. However, in practice, certain heuristic algorithms work well with real temporal data, and one simplified version, the simple temporal problems where each constraint $T_{i,j}$ consists of a single interval, have a well known polynomial time algorithm. These computational aspects are presented in Sections 2.2-2.3.

The quantitative approach to temporal reasoning offers convenient mechanisms for expressing and dealing with metric information which are important and useful in managing and understanding the timing of events. They form the basis for several implemented systems, and may be combined with other approaches.

2.2. Simple Temporal Problems and Distance Constraints. A metric temporal constraint problem (MTCP) in which at most one inequality $x_j - x_i \leq c_{i,j}$ is specified for each *ordered* pair (i, j) is called a *simple temporal problem* (STP). Alternatively, the constraint can be expressed on the *unordered* pair (i, j) by the inequality $-c_{j,i} \leq x_j - x_i \leq c_{i,j}$ or as the single canonical interval $T_{i,j} = [-c_{j,i}, c_{i,j}]$. Simple temporal problems can be solved using the efficient techniques available for finding shortest paths in a directed graph with weighted edges. These have been studied for many years as simple distance or difference constraints.

A system S of *simple distance* (or *difference*) *constraints* on a set of variables $\{x_i\}$ is a collection of weak and strict inequalities of the form

$$x_j - x_i \leq c_{i,j} \quad \text{or} \quad x_j - x_i < c_{i,j}$$

[1] Although we could include it with the label $[-\infty, \infty]$.

with at most one inequality for each ordered pair (i, j). By convention, we assign $c_{i,i} = 0$, and if no constraint is specified for a pair (i, j), then we may assign $c_{i,j} = \infty$. Clearly, any STP can be regarded as a system of simple difference constraints.

The problems that are studied here are concerned with the existence of a solution to the system of constraints; i.e., is S feasible? Bellman's notion of a *distance graph* is used to solve these problems[2]:

Construct a directed weighted graph G_S with directed edge $i \to j$ labelled by the *weight* $c_{i,j}$ and marked *solid* or *dotted* depending upon whether the inequality is weak or strict. Figure 2.1 shows the distance graph for a set of inequalities, with edges labelled ∞ omitted, and its corresponding edge distance matrix. Besides the immediate constraint $x_j - x_i \leq c_{i,j}$, each path $\pi = [i = i_0, i_1, \ldots, i_k = j]$ in G_S from i to j induces a constraint on the distance $x_j - x_i$ by summing the weights on the path, namely

$$x_j - x_i \leq weight(\pi) = \sum_{t=1}^{k} c_{i_{t-1}, i_t}$$

Bellman [1958] was the first to show that when all inequalities are weak, S is feasible if and only if G_S has no negative cycle. The more general result is the following.

Theorem 8. *A system S of simple distance constraints is feasible if and only if its distance graph G_S has no negative weight cycle and any zero weight cycle consists of $\leq$ edges only.*

The proof of Thereom 8 can be found in several references including Pe'er and Shamir [1995]. For completeness, we provide a proof which will also have the added benefit of leading us to the algorithms for finding a solution or a forbidden cycle.

Proof. Suppose there was a negative weight cycle $C = [i_0, i_1, \ldots, i_k = i_0]$ in G_S. Then $x_{i_0} - x_{i_0} \leq weight(C) < 0$ which cannot be satisfied. Next, suppose C is a zero weight cycle with at least one strict inequality. Then

$$x_{i_0} - x_{i_0} = \sum_{t=1}^{k} (x_{i_t} - x_{i_{t-1}}) < \sum_{t=1}^{k} c_{i_{t-1}, i_t} = weight(C) = 0$$

which is not feasible.

Conversely, if G_S has no negative cycle, then the minimum weight $d_{i,j}$ over all paths from i to j is well defined for every pair of vertices. However, this alone will not distinguish between weak and strict inequalities, which must be taken into account when constructing a feasible solution. Therefore, we consider two cases:

[2]Simple difference constraints are a special case of linear programming (LP); however, they can be solved by algorithms which are more efficient than a general LP algorithm.

- *All edges have labels* $\leq$. Choose a variable x_0 to be the origin. Since $d_{0,j} \leq d_{0,i} + d_{i,j}$ and $d_{i,j} \leq c_{i,j}$, we have $d_{0,j} - d_{0,j} \leq c_{i,j}$. Therefore, by assigning $x_k = d_{0,k}$, we obtain a feasible solution.

Remark 2. *Most books assume all inequalities are weak* ($\leq$) *and conclude the proof here.*

- *Some edges are labelled* $<$. We solve this case by constructing a slightly perturbed but equivalent system S' on the same set of variables as S. Let n be the number of variables and fix ε such that $0 < \varepsilon < c/n$ where c is the granularity of the weights, (e.g., if all weights are integers, then $c = 1$, or if weights are decimals with 4 significant digits to the right of the decimal point, then $c = 0.0001$, etc.) Define S' to include (1) every weak inequality $x_j - x_i \leq c_{i,j}$ in S, and (2) a weak equality $x_j - x_i \leq c_{i,j} - \varepsilon$ for every strict inequality $x_j - x_i < c_{i,j}$ in S. Thus, all edges in $G_{S'}$ have labels $\leq$, since all constraints of S' are weak inequalities.

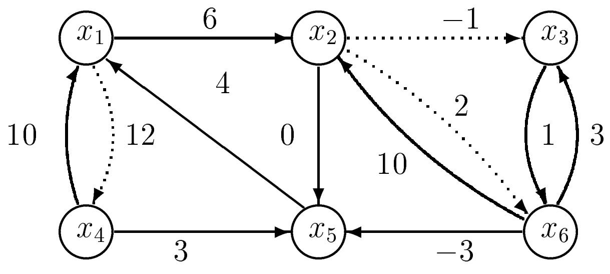
Claim: S is feasible iff S' is feasible.

Clearly, a feasible solution to S' also satisfies S. Suppose S' is not feasible but S is feasible. Then $G_{S'}$ must contain a negative weight cycle C. If all edges in C were labelled by $\leq$ in G_S, then the constraints on C would be identical in S and S', so C would also be a negative cycle in G_S, contradicting the feasibility of S. Therefore, we may assume that C does not consist of only $\leq$ edges in G_S, and so $weight_S(C) > 0$. On one hand, by the choice of c as the granularity of the edge weights, we must have $weight_S(C) > c$. On the other hand, the weight in G_S of each edge in C differs from the weight in $G_{S'}$ by no more than ε, and since C contains at most n edges, we have

$$weight_S(C) \leq weight_{S'}(C) + n\varepsilon < n\varepsilon < c$$

which is a contradiction. This proves the *Claim*.

Finally, we have shown that to solve a system S of weak and strict constraints, it is sufficient to solve the system S' of weak constraints, which can be done according to the first case above. The solution obtained for S' is also a solution for S, and the infeasibility of S' would imply the infeasibility of S. This concludes the proof of Theorem 8. ■



$$\begin{array}{ccc}
x_1 - x_4 \leq 10 & x_2 - x_1 \leq 6 & x_3 - x_2 < -1 \\
x_4 - x_1 < 12 & x_5 - x_4 \leq 3 & x_1 - x_5 \leq 4 \\
x_5 - x_2 \leq 0 & x_6 - x_2 < 2 & x_6 - x_3 \leq 1 \\
x_5 - x_6 \leq -3 & x_2 - x_6 \leq 10 & x_3 - x_6 \leq 3
\end{array}$$

	1	2	3	4	5	6
1	0	6	∞	**12**	∞	∞
2	∞	0	**−1**	∞	0	**2**
3	∞	∞	0	∞	∞	1
4	10	∞	∞	0	3	∞
5	4	∞	∞	∞	0	∞
6	∞	10	3	∞	−3	0

Figure 2.1. The distance graph for a set of inequalities, and its corresponding distance matrix. *Weak inequalities are indicated by solid arcs in the graph and normal font entries in the matrix and strict inequalities by dotted arcs and bold font entries.*

We have just seen that finding the minimum weight $d_{i,j}$ of a path from i to j, for all pairs i, j, allows us to construct a solution. This problem, often called the "all-pairs shortest path problem", can be solved by the Floyd-Warshall algorithm (Figure 2.2), the repeated matrix squaring algorithm or the Donald Johnson algorithm.

In the case where all edges represent weak inequalities, the solution $x_k = d_{0,k}$, for all k and an arbitrary x_0 is obtained immediately as in the proof of the Theorem. In the case of a mix of strict and weak inequalities, the appropriate ε can be calculated and the solution will be obtained by slightly perturbing the distances $d_{0,k}$.

An alternative method for constructing a solution uses the Bellman-Ford algorithm for solving the "single-source shortest path" problem. Here the origin x_0 is a *new* variable added to the problem and the corresponding new vertex 0 added to G_S serves as the source node with an edge weighted zero directed from 0 to each other vertex.

The complexity of the Floyd-Warshall algorithm is $O(n^3)$ and that of the Bellman-Ford algorithm is $O(mn)$ where n is the number of variables and m is the number of constraints (or edges).[3] Repeated matrix squaring is $O(n^3 \log n)$ and Johnson's

[3]Dijkstra's algorithm could also be used if all edge weights were non-negative, a very unlikely situation in our temporal problems.

algorithm is $O(n^2 \log n + mn)$ but is rather complicated. For further reading on these well-known algorithms, see the book by Cormen, Leiserson and Rivest [1990]. Figure 2.2 gives our version of the Floyd-Warshall algorithm. $M^{(k)}$ indicates the distance matrix after iteration k, hence, $m_{i,j}^{(0)} = c_{i,j}$ and $m_{i,j}^{(n)} = d_{i,j}$. To apply the algorithm to the example in Figure 2.1, we could choose $\varepsilon = 0.1$ and modify the three boldface entries in the original distance matrix accordingly to start the algorithm with $M^{(0)}$ as input.

Floyd-Warshall (M, n) [Input: the $n \times n$ distance matrix M]
$M^{(0)} \leftarrow M$
for $k \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** n **do**
      $m_{i,j}^{(k)} \leftarrow \min(m_{i,j}^{(k-1)}, m_{i,k}^{(k-1)} + m_{k,j}^{(k-1)})$
    endfor
  endfor
endfor
for $i \leftarrow 1$ **to** n **do**
  if $m_{i,i}^{(n)} < 0$ **then**
    print (Not feasible. Negative weight cycle.)
    exit
  endif
endfor
return $M^{(n)}$ [Output: the shortest distance matrix $M^{(n)}$]

Figure 2.2. The Floyd-Warshall algorithm to compute all-pairs shortest paths and test feasibility.

2.3. Solving General Temporal Problems. The general metric temporal constraint problem (MTCP) is known to be a computationally hard problem (Dechter, Meiri and Pearl [1991]). It is NP-complete even if there are at most two constraints per edge (pair of variables). Thus, all known algorithms to find a solution will have exponential complexity in the worst case (unless $P = NP$). Nevertheless, there is good reason to be optimistic because in practice many real world MTCPs can be solved by an increasing number of heuristic search algorithms using constraint propagation and intelligent backtracking. (The same statement can be made for many NP-complete problems which are important for applications.)

In this section we describe several techniques for guiding an otherwise arbitrary backtracking algorithm, teaching it to become smarter and smarter. Among these, we mention backjumping, forward checking, variable ordering, value ordering and learning schemes. Additional methods include path consistency and topologically motivated algorithms for exploiting the constraint network as a preprocessing step before applying backtracking.

Brute Force Backtracking. Let $T = (X, \{T_e\}_{e \in E})$ be a general MTCP given in canonical form, and let $G = (X, E)$ be its associated constraint network. Each edge $e \in E$ is labelled by a set of disjoint intervals $T_e = \{I_e^1, \ldots, I_e^r\}$. Selecting exactly one of these intervals for each edge gives an instance of a simple temporal problem (STP) which can be solved in polynomial time using the Floyd-Warshall algorithm for example (Section 2.2). By iterating over all possible selections and solving each of the STPs, we could (in principle) find all possible solutions to T. A brute force backtracking algorithm for doing this is given in Figure 2.3.

Since the number of possibilities is potentially exponential, we must be clever in how we perform this iteration. Even though in the worst case we cannot avoid the exponential complexity, it is possible to *guide* the search for a solution using techniques developed for constraint satisfaction problems and other combinatorially explosive problems.

choose an ordering of the edges of $E : e_1, e_2, \cdots, e_m$
for all $I_{e_1} \in T_{e_1}$ **do**
    for all $I_{e_2} \in T_{e_2}$ **do**
        $\vdots$
            for all $I_{e_m} \in T_{e_m}$ **do**
                call Floyd-Warshall with labels $(I_{e_1}, I_{e_2}, \ldots, I_{e_m})$
            endfor
        $\vdots$
    endfor
endfor

Figure 2.3. Brute force backtrack algorithm

Variable ordering, value ordering and testing partial instantiations. The order of the edges chosen in the first line of Figure 2.3 is called the *instantiation order*. The instantiation order of the interval variables $I_{e_1}, \ldots, I_{e_m}$ was chosen arbitrarily in the brute force algorithm. However, if the algorithm were to be stopped as soon as Floyd-Warshall finds a solution, then some instantiation orders would lead us to a solution faster than others.

One of the simplest heuristics that is usually profitable is ordering the edges according to number of intervals in their canonical constraint domains, from fewest to most. Other more complicated rules for ordering could also be considered. This is known as *variable ordering*.

A second degree of freedom ignored by the brute force algorithm is the order in which the domain intervals $I^1_{e_i}, \ldots, I^r_{e_i}$ are selected for each edge e_i. Again there may be more favorable orders within T_{e_i}, namely, choosing first those intervals whose likelihood of being in a solution is greatest. This is known as *value ordering*, but unfortunately it is usually difficult to predict in advance which domain intervals are more favorable.

A third opportunity of speeding up the brute force algorithm is by noting that it is not necessary to generate a full instantiation $I_{e_1}, \ldots, I_{e_m}$ before running Floyd-Warshall. Rather, it can be called to test a partial instantiation to find inconsistencies in a subgraph, and thus may rule out a large number of full instantiations early on. After all, *adding* additional constraints as the instantiation proceeds never adds new solutions, but instead usually cuts off many dead branches in the search tree.

In summary, we want to intelligently order the variables, and if possible the values within the domains, and then test partial instantiations periodically - either after every instantiation or at certain points during the search. There may also be an interaction between the variable ordering and the frequency of testing. Both theoretical and experimental studies have been carried out on variable and value ordering and alternative search methods, and the interested reader should consult the constraint satisfiability literature, for example, Mackworth [1977, 1992], Mackworth and Freuder [1985], Mohr and Henderson [1986], Nadel [1989], Nudel [1983] and their forward citations.

3. Interval and Point Temporal Algebras - A Qualitative Approach

Interval consistency problems deal with events, each of which is assumed to be an interval on the real line or on any other linearly ordered set. Allen [1983] defined a fundamental model for temporal reasoning using the thirteen *primitive interval relations*, expressing the relative positions between two intervals, obtained by considering all possible relations between their four endpoints (see Figure 3.1). We call this 13-valued interval algebra A_{13}.

RELATION	NOTATION	INTERPRETATION
x before y	$\prec$	
y after x	$\succ$	
x meets y	m	
y met-by x	m^{-1}	
x overlaps y	o	
y overlapped-by x	o^{-1}	
x starts y	s	
y started-by x	s^{-1}	
x during y	d	
y includes x	d^{-1}	
x finishes y	f	
y finished-by x	f^{-1}	
x equals y	$\equiv$	

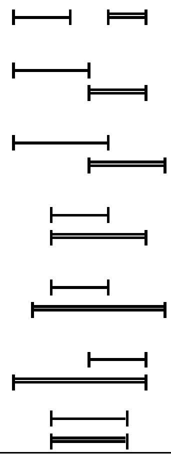


Figure 3.1. The 13-valued interval algebra A_{13}.
Thin line: x interval. Thick line: y interval.

These thirteen primitive relations represent all possible configurations between two intervals. However, in order to represent partial or nonexact information, we allow a disjunction of primitive relations to express the explicit knowledge we have about a pair of intervals.

Example 9. *$I_x\{\prec, m, d^{-1}\}I_y$ represents the disjunction*

$$I_x \textit{ before } I_y \quad \text{or} \quad I_x \textit{ meets } I_y \quad \text{or} \quad I_x \textit{ includes } I_y$$

Allen allows any nonempty subset of the 13 primitive relations to hold between a pair of intervals, that is, any of the 8191 possible disjunctions of A_{13}.

The approach of Golumbic and Shamir [1993] has been to simplify Allen's model in order to study its complexity using graph theoretic techniques. The first of the two lines of specialization which we study here is macro relations. Macro relations refers to partitioning the 13 primitive relations into more coarse relations by regarding a subset of primitive relations as a new basic relation. We let

$$\begin{array}{lll} \cap & = & \{m, m^{-1}, o, o^{-1}, s, s^{-1}, f, f^{-1}, d, d^{-1}, \equiv\} \\ \alpha & = & \{m, o\}, \qquad\qquad \alpha^{-1} = \{m^{-1}, o^{-1}\} \\ \subset & = & \{s, f, d\}, \qquad\qquad \subset^{-1} = \{s^{-1}, f^{-1}, d^{-1}\} \end{array}$$

From these we define the 3-valued and 7-valued interval algebras A_3 and A_7 whose elements are called its *atomic relations*, respectively,

$$A_3 : \{\prec, \succ, \cap\} \qquad\qquad A_7 : \{\prec, \succ, \alpha, \alpha^{-1}, \subset, \subset^{-1}, \equiv\}$$

For certain applications it is convenient to assume that all interval endpoints are distinct. This simplification generates the 6-valued algebra

$$A_6 : \{\prec, \succ, o, o^{-1}, d, d^{-1}\}$$

The choice of which of these algebras A_i to use depends on the nature of the application, data, constraints and on the type of complexity result being proved. We have already seen one example of using A_3 in the story of the Four Bears in Section 1.3. We shall present applications in archaeology and molecular biology later in this section.

We use the term 'algebra' here since the set of atomic relations in each A_i forms a Boolean algebra, and in certain cases, together with an additional composition operation, forms a *relation algebra*, in the sense defined by Tarski (see Ladkin & Maddux [1989]).

An even simpler specialization of Allen's model is the point algebra introduced by Vilain and Kautz [1986], in which each event is regarded as a single point rather than an interval. We denote the point algebra by P_3; its primitive relations are simply $\{<, =, >\}$. The point algebra has much more efficient algorithmic properties than the interval algebras, but has much less expressive power for modeling a temporal scenario. We study the point algebra P_3 in Section 3.3. **We cannot emphasize too strongly that P_3 and A_3 are very different**, stemming from the fact that equality between points is not at all like intersection between intervals.

3.1. Interval Satisfiability Problems. The input to all temporal reasoning problems which will be studied here is a set of events, and for each pair of events x and y, a set $D(x, y)$ of atomic relations in a fixed algebra A_i. The semantics here is that we do not know precisely the relationship between x and y, but it must be one of those in the set $D(x, y)$. For example, we read $D(x, y) = \{\prec, \subset\}$ (or, equivalently, $x\{\prec, \subset\}y$) as 'x is either before or contained in y'; and $D(x, y) = \{\prec, d^{-1}, \equiv\}$ and $x\{\prec, d^{-1}, \equiv\}y$ both mean 'either x ends before y starts, or y is during x, or the two are equal'. We also call a set of atomic relations a *relation set*. We omit braces when there is no ambiguity, e.g., $x \prec y$ or $x \cap y$. Without loss of generality, we assume that all relation sets are non-empty, and that for each pair of elements x and y, the relation sets $D(x, y)$ and $D(y, x)$ given as input are consistent, i.e., for each atomic relation R, $R \in D(x, y) \iff R^{-1} \in D(y, x)$. This corresponds to our canonical form for the MTCP in Section 2.1.

An *interval realization* (or *representation*) is an assignment of intervals on the real line to events. Since all the algebras discussed here are concerned with topological properties only, a realization can be viewed also as a complete weak order of the interval endpoints, thereby identifying all realizations which differ in metric only. A realization is *consistent* with the input $\{D(x,y)\}$ if for each pair of events, one of the atomic relations in their relation set holds. Two realizations are *distinct* if for at least one pair of events, the atomic relations which hold between them differ in the two realizations. The input data $\{D(x,y)\}$ is *consistent* (or *satisfiable*) if it admits a consistent realization.

The *interval satisfiability problem* (ISAT) is to determine if the input data $\{D(x,y)\}$ is consistent, and to find one consistent realization in case it is. The *minimal labeling problem* (MLP) is to determine the minimal sets $D'(x,y) \subseteq D(x,y)$ such that the set of consistent interval realizations is unchanged, and every remaining atomic relation holds in some realization.

Example 10. *For the input $x\{\prec, m, o\}y$, $y\{\prec, \equiv, \succ\}z$, $z\{f, s\}x$, two consistent realizations are shown in Figure 3.2. The relations $y \prec z$ and $z \equiv y$ are impossible. The minimal labeling for this problem is $x\{\prec, m, o\}y$, $y \succ z$, $z\{f, s\}x$.*

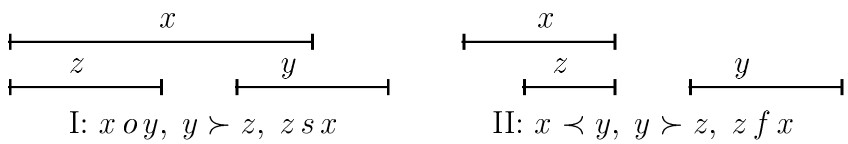


Figure 3.2. Two interval realizations for Example 10.

The *all consistent solutions problem* (ACSP) is that of enumerating all the combinations of atomic relations which are consistent with the given data. This problem arises since not all tuples of the cross product of a minimal labeling are consistent. The number of combinations may, of course, be exponential, and the goal here is to construct a *polynomial representation structure* $\sum$ requiring $O(p(n))$ space and from which k distinct solutions can be produced in $O(q(n,k))$ time, where n is the number of variables, p and q are polynomial functions, and k is any number less than or equal to the number of solutions. The closely related *endpoint sequence problem* (ESP) is that of enumerating all the distinct interval realizations which are consistent with the given data. For A_{13}, the last two problems are equivalent, but in the smaller algebras there may be several (or many) distinct endpoint sequences which realize the same combination of atomic relations.

Allen [1983] originally provided a heuristic approach for solving the MLP in A_{13}. It is a constraint propagation approach using a 13×13 composition or multiplication table for atomic relations, analogous to what we presented in Section 1.3 for A_3. That algorithm is polynomial but does not always provide a minimal solution, and may give a false positive answer to ISAT. Vilain and Kautz [1986] have shown that MLP is in fact NP-complete for A_{13}. Golumbic and Shamir [1993] obtain a stronger result using macro relations to reduce the number of atomic relations from thirteen to three. Our first main result is that even in A_3, the interval satisfiability problem is NP-complete. Consequently, all four problems ISAT, MLP, ACSP and ESP are intractable for all four interval algebras A_i, $i = 3, 6, 7, 13$.

To prove that ISAT is NP-complete for A_3, we introduced a new combinatorial problem, the interval graph sandwich problem (IGSP), which we proved to be NP-complete. The IGSP is shown to be a special case of ISAT. Recall that an undirected graph $G = (V, E)$ is called an *interval graph* if its vertices can be represented by intervals on the real line such that two vertices are adjacent if and only if their intervals intersect (see Golumbic [1980]). The interval graph sandwich problem (IGSP) is the following:

INTERVAL GRAPH SANDWICH PROBLEM:

INPUT: Two disjoint edge-sets, E^1 and E^2 on the same vertex set V.

QUESTION: Is there a graph $G = (V, E)$ satisfying $E^1 \subseteq E \subseteq E^1 \cup E^2$ which is an interval graph?

Denote $E^3 = V \times V - \{E^1 \cup E^2\}$. When $E^1 = \phi$ or $E^3 = \phi$, the answer is trivially yes. When $E^2 = \phi$, the problem is polynomial by the algorithm of Booth and Lueker [1976]. We show that in the general case, the problem is NP-complete.

An application in molecular biology: In physical mapping of DNA, information on intersection or non-intersection of pairs of segments originating from a certain DNA chain is known from experiments, without knowledge of the nucleotide sequences of the segments or the chain. The goal is to find out how the segments can be arranged as intervals along a line (the DNA chain), so that their pairwise intersections in that arrangement match the experimental data. (This question, raised by the famous biologist Seymour Benzer, was one of the original motivations for the study of interval graphs. In fact, Benzer raised the *decision version* of this question, since at the time the linearity of the DNA was only a hypothesis.) In the graph presentation, vertices correspond to segments and two vertices are connected by an E^1-edge (resp., E^3-edge) if their segments are known to intersect (resp., not to intersect). E^2-edges correspond to the case where the experimental information on the intersections is inconclusive, or simply unavailable. The decision problem is thus equivalent to the IGSP.

Theorem 11. *The interval graph sandwich problem is NP-complete.*

Proof. One proof (omitted in this paper) relies on a reduction from the Not-All-Equal 3-Satisfiability problem and the fact that an interval graph cannot contain an asteriodal triplet of vertices. The detailed proof is given in Golumbic and Shamir [1993]. An alternate (and shorter) proof based on a reduction from the Betweenness problem is given in Golumbic, Kaplan and Shamir [1995a].

Theorem 12. *ISAT is NP-complete for* A_3.

Proof. For a given instance of the IGSP problem, construct an instance of ISAT on A_3 as follows: For each edge $(x, y) \in E^1, E^2$ or E^3, let $D(x, y) = \cap, \prec \cap \succ$ or $\prec\succ$, respectively. It is clear that this ISAT problem has a solution if and only if the IGSP has one.

Corollary 13. *ISAT, MLP, ACSP and ESP are NP-hard for* A_3, A_6, A_7 *and* A_{13}.

Proof. This follows from the observation that the algebra A_3 is contained in each A_i and that for any $i = 3, 6, 7, 13$, ISAT has a solution if and only if MLP has a non-empty solution if and only if ACSP has a non-empty solution if and only if ESP has a non-empty solution. ■

3.2. Restricted Domains and Tractable Fragments. Due to the intractability of the general versions of ISAT, MLP and ACSP, attention has been focused on (1) polynomial time heuristic algorithms for approximate MLP solutions on A_{13} followed by efficient (albeit exponential) backtrack search, Allen [1983] and vanBeek [1989, 1991], and (2) exact polynomial time solutions to several restricted cases of the interval satisfiability problem, Golumbic and Shamir [1993]. In the latter, by suitably restricting the input domain of an NP-complete problem, one can often obtain a special class which admits a polynomial time algorithm, which is the topic of this section.

Solutions to several restricted cases of ISAT have been known for a long time, and many of these cases correspond to other applications. These will be extended by the new results presented here, and lead us to our second line of specialization, namely, restricted domains. In the general case for an interval algebra A_i, each relation set $D(x, y)$ may take any of $2^i - 1$ possible values. We may restrict this by designating a *fragment* or *restricted domain* Δ to denote a particular family of relation sets in A_i and requiring that each relation set $D(x, y)$ be a member of Δ. We represent each relation set in A_3 by a concatenation of its atomic relations, omitting braces. Hence, $\prec \cap$ represents $\{\prec, \cap\}$, $\prec$ represents $\{\prec\}$, etc. The seven possible relation sets in A_3

in this notation are: $\prec$, $\succ$, $\cap$, $\prec\cap$, $\cap\succ$, $\prec\succ$, $\prec\cap\succ$. Thus, the NP-completeness of the interval graph sandwich problem shows that even when all relation sets are restricted to be from the fragment $\Delta_0 = \{\cap, \prec\succ, \prec\cap\succ\}$ (meaning *intersect, disjoint* or *don't care*), ISAT remains NP-complete. The tractible and intractible fragments for A_{13} are fully characterized below.

In the temporal reasoning literature, polynomial algorithms have been developed for ISAT and MLP on special restrictions of A_{13}; e.g., by assuming that all events are points instead of intervals, Ladkin and Maddux [1994] and vanBeek [1989, 1991], or by assuming that the restricted domain satisfies certain convexity conditions, Nökel [1989, 1991]. Nebel and Burckert [1995] introduce a fragment of A_{13} called the 'ORD-Horn algebra' which they prove to be the maximal tractable fragment of A_{13} containing all the 13 atomic relations.

A number of well-known recognition problems in graph theory and partially ordered sets may be viewed as restricted interval satisfiability problems. Five of these, all of which have polynomial time solutions, are given in Figure 3.3 along with their appropriate Δ, (see Golumbic [1980]).

Class	Restricted Domain Δ
Interval orders	$\{\prec, \succ, \cap\}$
Interval graphs	$\{\prec\succ, \cap\}$
Circle (or overlap) graphs	$\{\alpha, \alpha^1, \equiv\}, \{\prec, \succ, \subset, \subset^{-1}\}\}$
Interval containment graphs	$\{\{\subset, \subset^{-1}\}, \{\prec, \succ, \alpha, \alpha^{-1}, \equiv\}\}$
Posets of dimension 2	$\{\{\subset\}, \{\subset^{-1}\}, \{\prec, \succ, \alpha, \alpha^{-1}, \equiv\}\}$

Figure 3.3. Polynomial interval satisfiability problems in graph theory.

We describe the results of a systematic study on the complexity of the restricted domains in A_3. Since we can assume that two converse relations sets ($\prec$ and $\succ$, or $\prec\cap$ and $\succ\cap$) always appear together in a restricted domain, there are 31 possible fragments. All of these cases have now been classified as either polynomial or NP-complete. For certain restricted domains special polynomial algorithms have been devised. An interesting example is the restricted domain $A_3 - \prec\succ$: By excluding only one out of the seven relation sets, ISAT becomes linearly solvable. For lack of space, we only state here the results for restricted domains without proofs or algorithms, and refer the interested reader to Golumbic and Shamir [1993] for fragments Δ_1–Δ_3 and Webber [1995] for Δ_4 in the theorems below. Another positive result is that the polynomiality of ISAT implies the polynomiality of MLP and the existence of a polynomial representation structure for ACSP. These results apply to each of the algebras A_i ($i = 3, 6, 7, 13$) and also to any restricted domain in these algebras.

Theorem 14. *Let Δ_1 be the restricted domain $A_3 - \prec\succ$, that is*

$$\Delta_1 = \{\prec, \succ, \cap, \prec\cap, \cap\succ, \prec\cap\succ\}.$$

(1) ISAT(Δ_1) is solvable in linear time.

(2) MLP(Δ_1) is solvable in $O(mn)$ steps for a problem with n events and m explicit relation sets.

(3) There exists a polynomial representation structure which facilitates the solution of ESP on Δ_1 in $O(n)$ steps per realization.

Theorem 15. *ISAT is solvable in $O(n^3)$ time for $\Delta_2 = \{\prec, \succ, \cap, \prec\succ\}$.*

Theorem 16. *ISAT is solvable in linear time for $\Delta_3 = \{\prec, \succ, \prec\cap\succ, \prec\succ\}$.*

Theorem 17. *ISAT is NP-complete for $\Delta_4 = \{\prec\cap, \cap\succ, \prec\succ\}$.*

Since every subset of a polynomially solvable restricted domain is polynomial, and every superset of an NP-complete fragment is NP-complete, the above results resolve the complexity of all 31 possible restricted domains in A_3.

An application in archaeology: The *seriation* problem in archaeology attempts to place a set of artifact types in their proper chronological order. This problem was formulated by Flinders Petrie, a well-known archaeologist at the turn of the century, while studying 800 types of pottery found in 900 Egyptian graves. To each artifact type there corresponds a time interval (unknown to us) during which it was in use. To each grave there is a point in time (also unknown) when its contents were interred. Each grave provides the intersection data for the intervals corresponding to its contents.

3.3. Point Algebras. Villain and Kautz [1986] have suggested an alternate model, which they have called the *point algebra* (PA), to simplify and cope with the computational limitations of IA, the Interval Algebra A_{13}. The point algebra, which we denote by P_3, deals only with the sets of *point relations*, namely, the 7 non-empty subsets of $\{<, =, >\}$:

$$<, =, >, \leq, \geq, \neq, ?$$

where $\leq, \geq, \neq, ?$ stand for $\{<, =\}$, $\{=, >\}$, $\{<, >\}$, $\{<, =, >\}$, respectively. All the computational problems which we have studied for the interval algebras are polynomial for point algebras. For further reading on this and their applications, see Nökel [1991] and vanBeek [1992].

4. Temporal Logic

The approach to temporal reasoning perhaps most familiar to the mathematics community is the study of temporal logics. Temporal logic is not simply a variation of predicate or propositional logic, but rather, a very special family of logics centered upon a distinguished variable t which ranges over a *flow of time.* What makes the logic special is that the truth of axioms and propositions involving the time parameter t are dictated by its particular properties. How we choose to view time, as distinct time points 0,1,2,3,..., as a continuous total order, as a multiple time flow with linear past and branching futures, etc., imposes its unique attributes on the logical model.

Some temporal examples are quite easy to express in logical form. The sentence, *"Fred is driving the carpool today and will continue to drive the carpool every three days,"* could be represented as

$$\begin{aligned}&Drive(\underline{\text{now}}, \text{Fred, Carpool})\\ &\quad \&(\forall t, s \geq \underline{\text{now}})\\ &\qquad [Drive(t,\text{Fred,Carpool}) \ \& \ (t+2=s) \longrightarrow Drive(s,\text{Fred, Carpool})]\end{aligned}$$

But very quickly we can get to sentences whose predicate logic forms are increasingly unreadable. Consider the following example, which describes a situation which will continue until another situation finishes:

> *"Since B became University President (Pres) the value of the endowment fund (Value) has steadily been going down, and it will continue to go down for as long as he or she remains President."*

$$\begin{aligned}&\exists t < \underline{\text{now}}[Pres(t,B)\\ &\quad \& \ \exists s < t \forall u[s < u < t \longrightarrow \neg Pres(u,B)]\\ &\quad \& \ \forall u[t < u < \underline{\text{now}} \longrightarrow Pres(u,B)]\\ &\quad \& \forall u,v,x,y[t < u < v < \underline{\text{now}} \ \& \ Value(u,x) \ \& \ Value(v,y) \longrightarrow x > y]]\\ &\& \ \forall s > \underline{\text{now}}[\forall u[\underline{\text{now}} < u < s \longrightarrow Pres(u,B)]\\ &\quad \longrightarrow \forall u,v,x,y[\underline{\text{now}} < u < v < s \ \& \ Value(u,x) \ \& \ Value(v,y) \longrightarrow x > y]]\end{aligned}$$

We could model the temporal dependence between our incumbent President and the financial decline of the Endowment using the qualitative approach of interval algebra in Section 3. A concise *temporal logic* solution to such an example is also possible. Temporal logics add special operators, constructs and rules to a standard propositional or predicate logic in order to enhance its expressibility and capacity for reasoning about time. The examples above are patterned after those in the recent book by Gabbay, Hodkinson and Reynolds [1994] which may be consulted for a comprehensive treatment of the subject.

Constraints of space and time do not permit us here to give an extensive survey of the logical approach to temporal reasoning, nor the use of logic programming as a computational tool for automated reasoning about time. The chapters by Kirchner and Lassez in this book deal with some of these aspects. A further source of material can be found in the *Journal of Logic and Computation*, the survey by Emerson [1990] and other similar publications. For the remainder of this section we will highlight some of the main issues and obstacles in this field.

One complication that arises is when we need to express a difference between the *planned* future and the *actual* future. For example, *"Glenn took a flight to Orlando, but the plane landed in Tampa instead."* Although our common notion of time is linear, here we see the need for branching of time between various possible outcomes. Another matter of difficulty is incorporating *functional* dependence, as with the loaded gun in the Yale shooting problem which may jam or backfire. Even when the order of events is determined and unambiguous, there remain past-future relationships which must be constrained by *causality* or *closeness*. An example illustrating this is given in Gabbay et al. [1994], *"Last year, every time I went to London I came back with a headache."*

There is no one single temporal logic, as we have alluded to before, which can capture even a majority of the concepts of time one can imagine. Specialized temporal logics are needed to neatly express each such set of relationships. We have mentioned before the choice of how to view the flow of time, as integers, reals, poset, etc., and the units of time, as points, intervals, or infinite subsequences. Other factors include restrictions of finiteness, events being true 'once only' or multiple times, temporal connectives like 'X true until Y' or 'true tomorrow' and descriptive durations such as 'sometimes true', 'usually true', 'forever more true', 'occasionally true'.

There are many philosophical aspects to temporal reasoning which further complicate logical representation. Consider the following example.

Example 18. *A construction site which at time t_i is 'building a factory' but which at a later time t_j has changed to 'building a supermarket' without any changes whatsoever in the plans. The truth of 'what' is being built is only in the mind of the current owner. Or to make things even more interesting, suppose a sales representative at time t_k $(t_i < t_k < t_j)$ has two other potential customers, a skating rink operator and a bowling alley chain. What is the construction site for them?*

These and other considerations make temporal reasoning a major challenge for artificial intelligence.

5. Temporal Issues in Other Areas

5.1. Distributed Artificial Intelligence: A Handful of Autonomous Agents.

Five Golumbic women, each with a mind of her own, each pursuing her individual goals, usually self-motivated, share a common shortage of resources. There is no centralized decision maker (i.e., dictator). An environment for conflict! An opportunity for cooperation!

Arguments, persuasive counter-arguments, negotiation

This is what *Distributed Artificial Intelligence* (DAI) is all about: establishing (relative) cooperation and coordination between autonomous agents. Through communication and techniques for executing negotiations all disputed subjects are to be resolved. Moreover, since cooperation between the agents is usually needed in advance to accomplish their goals, it should be planned ahead. The DAI literature is young and growing rapidly and raises many temporal issues.

Example 19. *It's 6:45 pm and three women need reasonable shares of the remaining hot water and individual six minute time slots in the shower. Two of them need seven minute time slots with the hair dryer (presumably after their showers). All five have to eat something, but none of them has any time to throw together a quiche and salad. Four have to access the ironing board, and one has to spend a few quiet moments with the author. There is ample dressing and mirror space, but only two sinks. Two must have use of the one computer with a printer, and three will require vehicular support. All have a deadline of 7:30 pm.*

In distributed AI, one agent tries to persuade another agent to change its intentions and to increase its will to cooperate. *'You don't really need the station wagon; it's only two kilometers to the Teen Ween Campfire Rockathon and there is no parking.'* The negotiation process influences the other agent's mental state. *'I can skip that shower; why break a long standing precedent?'* During each phase of negotiation, the agent has the ability to generate any type of argument or justification that will convince its counterpart to cooperate. *'When I was your age, I could whip up a salmon mousse and Minute Rice in less than a minute.'*

After repeated encounters, the agents will analyze their opponent's behavior, learn from past events, and gather experience. *'Let's let A dry her hair first; last time, she (i) kept pulling the plug on us, (ii) kept making those teary faces at us, (iii) almost missed her 7:30 deadline, (iv) all the above.'* The agents 'learn' to execute more effective negotiation in future conflicts. *'Somebody order a pizza.'*

The goal of distributed AI is to successfully accomplish all of this by setting down a number of inference rules based on a formal logic. This model simulates the negotiations. The more realistic the rules and the better the logical methodology, the greater the chance of actually providing an automated solution. Further reading on DAI can be found in Bond and Gasser [1988], O'Hare and Jennings [1996].

5.2. Temporal Databases, Query Languages and Discovery. Databases often contain information which have temporal attributes, e.g., when the information was stored, when it became valid, when it ceased to be valid, etc. It is of increasing importance to be able to represent this temporal data in a manner that facilitates answering temporal queries.

One approach has been to extend large programming methodologies for this purpose, and this has led researchers to design such languages as Templog, temporal extensions of Datalog, Temporal Deductive Databases, Tempur, TL1, THLP, TQuel, HRDM, TSQL2 and others, for the purpose of expressive and efficient temporal query processing. In the design of such a language or methodology, many of the same issues of representation which we have discussed earlier arise, namely, time model options (discrete, continuous, point-based, interval-based), explicit representation of a set of values (lists of dates, times, etc. such as a printed train schedule) or implicit representation (deduction rules, repetition formulas, constraints, etc., such as, $true\ at\ t_0$ and $true\ \forall t = t_0 + 24n$ for $1 \leq n \leq 10$, n integer).

Another approach extends database systems by adding temporal operations and queries, and others by building totally new data models and languages. Surveys of the issues being studied for temporal databases can be found in Chomicki [1994], Roddick and Patrick [1992] and the articles appearing in Tansel, et al. [1993]. For further reading in this area, see references [1, 11, 14–19, 23–26, 47, 51, 62, 64, 78, 81, 84, 86].

6. Conclusion

We have briefly explored some of the main themes of temporal reasoning through qualitative, quantitative and logical methods. It is a vast topic on which many volumes have been written. We wish the reader success in following a few of the ideas presented here, pursuing a path through those volumes, and building upon them new results, theories, structures and algorithms for reasoning about time.

Acknowledgements

I am pleased to acknowledge my colleague Ron Shamir for our collaborative work relating combinatorial and temporal results on interval relations, and Itay Meiri for his work on constraint-based temporal reasoning which appeared in the working notes. My thanks to Fred Hoffman and Glenn Shafer for their careful reading and helpful comments on this paper.

A personal remark: My research in the area of applied combinatorial algorithms during the past twenty years has taken me into many disciplines within computer science and discrete applied mathematics. While maintaining my longstanding interest in perfect graphs and algorithmic graph theory, a collage of related topics has emerged: constraint based scheduling from artificial intelligence, compiler and software optimization, VLSI circuit problems, physical mapping of DNA, and many mathematical results on classes of intersection graphs and their applications.

This eclectic collection has an algorithmic commonality whose properties have been the focus of this researcher's work. "Reasoning about time" has entered almost all of these areas. Planning the efficient use of resources, scheduling meetings, analyzing a sequence of events which have happened, ordering a list of instructions to minimize wasted processor time or to meet a deadline, designing timing driven algorithms for early delay estimation, applying the same temporal interval results to genetic patterns in molecular biology, are all examples of *time* playing a central role in every aspect of our lives.

It is my hope that this article will provide an introduction to the field of temporal reasoning and opportunities for further research.

References

[1] M. Abadi and Z. Manna, Temporal logic programming, *Journal of Symbolic Computation* **8** (1989).

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[3] J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* **26** (1983), 832-843.

[4] J. F. Allen, Towards a general theory of action and time, *Artificial Intelligence* **23** (1984), 123-154.

[5] J. F. Allen and P. J. Hayes, A common-sense theory of time, *Proc. IJCAI-85*, Los Angeles, CA, pp. 528-531, 1985.

[6] J. F. Allen and P. J. Hayes, Moments and points in an interval-based temporal logic, *Computational Intelligence* **5** (1989), 223-238.

[7] M. J. Almeida, *Reasoning About the Temporal Structure of Narrative*, Ph.D. thesis, State University of New York, Buffalo, NY, 1987.

[8] S. Arnborg, D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM Journal of Algebraic Discrete Methods* **8** (1987), 177-184.

[9] B. Aspvall and Y. Shiloach, A polynomial time algorithm for solving systems of linear inequalities with two variables per inequality, *SIAM Journal of Computing* **9** (1980), 827-845.

[10] R. C. Backhouse and B. A. Carre, Regular algebra applied to path-finding problems, *J. Inst. Math. Applications* **15** (1975), 161-186.

[11] M. Baudinet, On the expressiveness of temporal logic programming, *Information and Computation* **117** (1995), 157-180.

[12] U. Bertelé and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, New York, 1972.

[13] A.H.Bond and L.Gasser, eds., *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[14] B. C. Bruce, A model for temporal references and its application in a question answering program, *Artificial Intelligence* **3** (1972), 1-25.

[15] J. Chomicki, Temporal query languages: a survey, *Kansas State University Report*, Jan. 1995. An earlier version appears in *Lecture Notes in Artificial Intelligence* **827** (1994).

[16] J. Chomicki, Efficient checking of temporal integrity constraints using bounded history encoding, *ACM Trans. on Database Systems* **20** (1995), 149-186.

[17] J. Chomicki and D. Niwinski, On the feasibility of checking temporal integrity constraints, *J. Computer System Sciences* **51** (1995), 523-535.

[18] J. Chomicki and D. Toman, Implementing temporal integrity constraints using an active DBMS, *IEEE Trans. Knowledge and Data Engr.* **7** (1995), 566-582.

[19] J. Clifford and D. S. Warren, Formal semantics for time in databases, *ACM Transactions on Database Systems* **8** (1983), 214-254.

[20] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[21] G. B. Dantzig, *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1962.

[22] E. Davis, Constraint propagation with interval labels, *Artificial Intelligence* **32** (1987), 281-331.

[23] T. Dean, Large-scale temporal data bases for planning in complex domains, *Proc. IJCAI-87*, Milan, Italy, pp. 860-866, 1987.

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

[25] T. Dean, J. Firby and D. Miller, Hierarchical planning involving deadline, travel time and resources, *Computational Intelligence* **4** (1990), 381-398.

[26] T. L. Dean and D. V. McDermott, Temporal data base management, *Artificial Intelligence* **32** (1987), 1-55.

[27] R. Dechter, Enhancement schemes for constraint processing: backjumping, learning, and cutset decompositions, *Artificial Intelligence* **41** (1990), 273-312.

[28] R. Dechter, Constraint networks, in: *Encyclopedia of Artificial Intelligence, 2nd Edition*, S. Shapiro, editor, John Wiley, New York, 1992.

[29] R. Dechter, I. Meiri and J. Pearl, Temporal constraint networks, *Artificial Intelligence* **49** (1991), 61-95.

[30] R. Dechter and J. Pearl, Network-based heuristics for constraint satisfaction problems, *Artificial Intelligence* **34** (1987), 1-38.

[31] R. Dechter and J. Pearl, Tree clustering for constraint networks, *Artificial Intelligence* **38** (1988), 353-366.

[32] Y. Deville and P. Van Hentenryck, An efficient arc consistency algorithm for a class of CSP problems. *Proc. IJCAI-91*, Sydney, Australia, 1991, pp. 325-330.

[33] E. A. Emerson, Temporal and modal logic, in: *Handbook of Theoretical Computer Science,* J. van Leeuwen, editor, Elsevier/MIT Press, 1990, pp. 995-1072.

[34] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[35] C. Freksa, Temporal reasoning based on semi-intervals, *Artificial Intelligence* **54** (1992), 199-227.

[36] E. C. Freuder, Synthesizing constraint expressions, *Communications of the ACM* **21** (1978), 958-965.

[37] E. C. Freuder, A sufficient condition of backtrack-free search, *Journal of the ACM* **29** (1982), 24-32.

[38] D. M. Gabbay, I. Hodkinson and M. Reynolds, *Temporal logic: Mathematical Foundations and Computational Aspects (vol. 1)*, Oxford University Press, 1994.

[39] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, CA, 1979.

[40] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[41] M. C. Golumbic, H. Kaplan and R. Shamir, On the complexity of DNA physical mapping, *Advances in Applied Mathematics* **15** (1994), 251-261.

[42] M. C. Golumbic and R. Shamir, Complexity and algorithms for reasoning about time: a graph-theoretic approach, *J. Assoc. Comput. Mach.* **40** (1993), 1108-1133.

[43] J. Y. Halpern and Y. Shoham, A propositional modal logic of time intervals, *Journal of the ACM* **38** (1991), 935-962.

[44] I. Hamlet and J. Hunter, A representation of time for medical expert systems, *Lecture Notes in Medical Informatics* **33**, Springer-Verlag, 1987, 112-119.

[45] S. Hanks and D. V. McDermott, Default reasoning, nonmonotonic logics, and the frame problem. *Proc. AAAI-86*, Philadelphia, PA, pp. 328-333, 1986.

[46] R. M. Haralick and G. L. Elliott, Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence* **14** (1980), 263-313.

[47] C. S. Jensen, and R. T. Snodgrass, Temporal specialization and generalization, *IEEE Trans. on Knowledge and Data Engr.* **6** (1994), 954-974.

[48] K. Kahn and G. A. Gorry, Mechanizing temporal knowledge, *Artificial Intelligence* **9** (1977), 87-108.

[49] H. Kautz and P. B. Ladkin, Integrating metric and qualitative temporal reasoning, *Proc. AAAI-91*, Anaheim, CA, pp. 241-246, 1991.

[50] D. G. Kendall, Incidence matrices, interval graphs and seriation in archaeology, *Pacific J. Math* **28** (1969), 565-570.

[51] M. Koubarakis, Database models for infinite and indefinite temporal information, *Information Systems* **19** (1994), 141-174.

[52] P. B. Ladkin, Time representation: a taxonomy of interval relations, *Proc. AAAI-86*, Philadelphia, PA, pp. 360-366, 1986.

[53] P. B. Ladkin, Metric constraint satisfaction with intervals, Technical Report TR-89-038, International Computer Science Institute, Berkeley, CA, 1989.

[54] P. B. Ladkin and R. D. Maddux, The algebra of constraint satisfaction problems and temporal reasoning, Technical Report, Kestrel Institute, Palo Alto, CA, 1988.

[55] P. B. Ladkin and R. D. Maddux, On binary constraint problems, *Journal of the ACM* **41** (1994), 435-469.

[56] P. B. Ladkin and A. Reinefeld, Effective solutions of qualitative interval constraint problems, *Artificial Intelligence* **57** (1992), 105-124.

[57] C. E. Leiserson and J. B. Saxe, A mixed-integer linear programming problem which is efficiently solvable, *Proc. 21st Annual Allerton Conference on Communication, Control, and Computing*, pp. 204-213, 1983.

[58] Y. Z. Liao and C. K. Wong, An algorithm to compact a VLSI symbolic layout with mixed constraints, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **CAD-2** (1983), 62-69.

[59] A. K. Mackworth, Consistency in networks of relations, *Artificial Intelligence* **8** (1977), 99-118.

[60] A. K. Mackworth, Constraint satisfaction, in: *Encyclopedia of Artificial Intelligence, 2nd Edition*, S. Shapiro, editor, John Wiley, New York, 1992.

[61] A. K. Mackworth and E. C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, *Artificial Intelligence* **25** (1985), 65-74.

[62] R. Maiocchi and B. Pernici, Temporal data management systems: A comparative view, *IEEE Trans. on Knowledge and Data Engr.* **3** (1991), 504-524.

[63] J. Malik and T. O. Binford, Reasoning in time and space, *Proc. IJCAI-83*, Karlsruhe, Germany, pp. 343-345, 1983.

[64] L. E. McKenzie, Jr. and R. T. Snodgrass, Evaluation of relational algebras incorporating the time dimension in databases, *ACM Computing Surveys* **23** (1991), 501-543.

[65] D. V. McDermott, A temporal logic for reasoning about processes and plans, *Cognitive Science* **6** (1982), 101-155.

[66] I. Meiri, Combining qualitative and quantitative constraints in temporal reasoning, *Artificial Intelligence* (to appear 1996).

[67] R. Mohr and T. C. Henderson, Arc and path consistency revisited, *Artificial Intelligence* **28** (1986), 225-233.

[68] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, *Information Sciences* **7** (1974), 95-132.

[69] B. A. Nadel, Constraint satisfaction algorithms, *Computational Intelligence* **5** (1989), 188-224.

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

[71] K. Nökel, Temporal matching: Recognizing dynamic situations from discrete measurements, *Proc. IJCAI-89*, Detroit, MI, pp. 1255-1260, 1989.

[72] K. Nökel, Temporally distributed symptoms in technical diagnosis, *Lecture Notes in Artificial Intelligence* **517**, Springer-Verlag, 1991.

[73] B. Nudel, Consistent-labeling problems and their algorithms: expected-complexities and theory-based heuristics, *Artificial Intelligence* **21** (1983), 135-178.

[74] G. O'Hare and N. Jennings, *Foundations of Distributed Artificial Intelligence*, John Wiley, 1996.

[75] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[76] M. Poesio and R. J. Brachman, Metric constraints for maintaining appointments: dates and repeated activities, *Proc. AAAI-91*, Anaheim, CA, pp. 253-259, 1991.

[77] P. W. Purdom, Search rearrangement backtracking and polynomial average time, *Artificial Intelligence* **21** (1983), 117-133.

[78] J. F. Roddick and J. D. Patrick, Temporal semantics in information systems - a survey, *Information Systems* **17** (1992), 249-267.

[79] Y. Shoham, *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press, Cambridge, MA, 1988.

[80] R. Shostak, Deciding linear inequalities by computing loop residues, *Journal of the ACM* **28** (1981), 769-779.

[81] R. T. Snodgrass, Temporal databases, in: *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Springer-Verlag, *LNCS* **639**, 1992.

[82] F. Song and R. Cohen, The interpretation of temporal relations in narrative, *Proc. AAAI-88*, St. Paul, MN, pp. 745-750, 1988.

[83] F. Song and R. Cohen, Temporal reasoning during plan recognition, *Proc. AAAI-91*, Anaheim, CA, pp. 247-252, 1991.

[84] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev and R. Snodgrass, eds., *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings, 1993.

[85] A. Tarski, On the calculus of relations, *J. Symbolic Logic* **6** (1941), 73-89.

[86] C. I. Theodoulidis and P. Loucopoulos, The time dimension in conceptual modelling, *Information Systems* **16** (1991), 273-300.

[87] R. E. Valdés-Pérez, The satisfiability of temporal constraint networks, *Proc. AAAI-87*, Seattle, WA, pp. 256-260, 1987.

[88] P. VanBeek, Approximation algorithms for temporal reasoning, *Proc. IJCAI-89*, Detroit, MI, pp. 1291-1296, 1989.

[89] P. VanBeek, Temporal query processing with indefinite information, *Artificial Intelligence Med.* **3** (1991), 745-750.

[90] P. VanBeek, Reasoning about qualitative information, *Artificial Intelligence* **58** (1992), 297-326.

[91] P. VanBeek and R. Cohen, Exact and approximate reasoning about temporal relations, *Computational Intelligence* **6** (1990), 132-144.

[92] J. F. A. K. VanBenthem, *The Logic of Time, 2nd Edition*, D. Reidel, 1991.

[93] S. A. Vere, Planning in time: Windows and durations for activities and goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **5** (1983), 246-267.

[94] M. Vilain, A system for reasoning about time, *Proc. AAAI-82*, Pittsburgh, PA, pp. 197-201, 1982.

[95] M. Vilain and H. Kautz, Constraint propagation algorithms for temporal reasoning, *Proc. AAAI-86*, Philadelphia, PA, pp. 377-382, 1986.

[96] A. B. Webber, Proof of the interval satisfiability conjecture, *Annals of Mathematics and Artificial Intelligence* **15** (1995), 231-238.

[97] B. C. Williams, Doing time: Putting qualitative reasoning on firmer ground, *Proc. AAAI-86*, Philadelphia, PA, pp. 105-113, 1986.