Complexity and algorithms for reasoning about time: a graph-theoretic approach

Martin Charles Golumbic
IBM Israel Scientific Center
Technion City, Haifa, Israel
and Bar-Ilan University
Ramat Gan, Israel.
email: golumbic@israearn.bitnet

Ron Shamir
Department of Computer Science
Sackler Faculty of Exact Sciences
Tel Aviv University
Tel-Aviv 69978, Israel.
email: shamir@math.tau.ac.il

May 1991, revised May 1992

Abstract

Temporal events are regarded here as intervals on a time line. This paper deals with problems in reasoning about such intervals when the precise topological relationship between them is unknown or only partially specified. This work unifies notions of interval algebras in artificial intelligence with those of interval orders and interval graphs in combinatorics.

The *satisfiability, minimal labeling, all solutions* and *all realizations* problems are considered for temporal (interval) data. Several versions are investigated by restricting the possible interval relationships yielding different complexity results. We show that even when the temporal data comprises of subsets of relations based on intersection and precedence only, the satisfiability question is NP-complete. On the positive side, we give efficient algorithms for several restrictions of the problem. In the process, the *interval graph sandwich problem* is introduced, and is shown to be NP-complete. This problem is also important in molecular biology, where it arises in physical mapping of DNA material.

Keywords: temporal reasoning, interval graphs, interval orders, satisfiability, complexity, algorithmic analysis, NP-completeness, sandwich problems, DNA mapping

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 [25, 26], behavioral psychology [12], temporal reasoning [2], operations research [33], medical diagnosis [31], circuit design [43] and combinatorics [34]. Indeed, it was the intersection data of time intervals that lead Hajös [23] to define and ask for a characterization of interval graphs, and which provides the clues for solving the "Berge mystery story" [20, p. 20]. Other applications arise in non-temporal contexts: For example, in molecular biology, arrangement of DNA segments along a linear chain involves similar problems [7].

Motivated by the proposed use of Allen's interval algebra [2] in order to maintain knowledge about time dependent events, several researchers have investigated variants of this algebra and the trade-off between expressivity, completeness and computational complexity in reasoning with them [39, 42]. This approach leaves unspecified the exact temporal relationship between events. In [4] combinatorial structures have been introduced for representing temporal data associated with interval (precedence) orders and interval (intersection) graphs. In that approach the temporal relations are specified as "precedes", "follows", or "intersects" but the exact ordering of the endpoints of the intervals is left unspecified.

In this paper, we relate the two notions of interval algebra from the temporal reasoning community and interval graphs from the combinatorics community, obtaining new algorithmic complexity results of interest to both disciplines. In Section 2, we present the background on constraint satisfiability problems as they relate to the study of interval algebras, and introduce the notions of macro relations and restricted domains. Macro relations allow coarse partitions of the primitive interval relations which are suitable to certain applications and provide the most general theorems. Restricted domains forbid specified combinations of primitive relations, thereby giving polynomial time solutions, in many useful cases, to otherwise intractable problems. General results on the relative complexity of the problems of interval satisfiability, minimal labeling and finding all consistent solutions are presented in Section 3. Several new NP-completeness results are proved in Section 4, specifically the interval satisfiability problem for the 3-valued interval algebra and the interval graph sandwich problem. Section 5 deals with polynomial time solutions to restricted domain problems. Our conclusions are given in Section 6.

2 Background

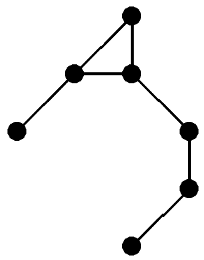
2.1 Constraint satisfiability problems

A *constraint satisfiability problem* (CSP) consists of a set of *variables* $V = \{v_1, \ldots, v_n\}$, their associated *domains* $D_1, \ldots, D_n$ and a set of *constraints* on these variables. The domain of a variable is a group of values to which the variable may be instantiated. A value of a variable is also called its *label*. Constraint satisfiability problems are also known in the artificial intelligence literature as *consistent labeling problems* [32]. A *solution* to the CSP consists of an instantiation (or labeling) of all the variables which does not violate any of the constraints, i.e., a consistent labeling of each variable with a value from its domain.

The *constraint satisfiability problem* itself is to determine if there *exists* a solution; this is essentially equivalent to the so called *consistent singleton labeling problem* which is to *find* one solution. The *minimal labeling problem* (MLP) is to determine the "strongest possible assertions about the domains, which do not change the set of solutions" i.e., to determine the minimal sets of values $D'_i \subseteq D_i$ such that the set of solutions for the resulting CSP will be identical to that of the original problem. (In particular, this implies that the remaining labels are exactly those which participate in some solution that is consistent with the input data.) In computer vision, temporal reasoning and other application areas, it is sometimes regarded as an approximation strategy for first cutting down the solution search space before other (possibly exhaustive) solution finding techniques are employed. The *all solutions problem* (ASP), is to construct a *polynomial representation structure* Σ 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 structure Σ thus represents *all* possible solutions.

$$\begin{array}{cccccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array} \qquad \begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}$$

$\mathbf{M} = D_1 \times D_2$ $\qquad$ $\mathbf{M}'$



Σ

Figure 1: An example of a constraint satisfaction problem.

Example 2.1 The matrix $\mathbf{M}$ in Figure 1 illustrates the cross product of the domains for a problem on two variables where 1 indicates a solution and 0 a non-solution, as determined by the constraints which are only implicit here. The satisfiability problem is to determine if $\mathbf{M}$ has a 1 in it, and consistent singleton labeling must find the coordinates of one. The minimal labeling problem is to determine the submatrix $\mathbf{M}'$ of $\mathbf{M}$ obtained by eliminating all rows and columns having only 0 entries. The all consistent solutions problem must construct a representation structure, such as the graph Σ of closest neighbors in Figure 1, which allows generating all solutions.

In the context of this paper, the variables will correspond to pairs of temporal events (i.e., intervals) and will take on values which represent the qualitative relationship between them (i.e., intersect, overlap, contain, less than, etc.) The constraints will be those implicitly imposed upon the events by their being intervals on a line.

2.2 Interval algebras and macro relations

Allen [2] has defined a model for temporal logic whose elements are time intervals (corresponding to events that happen during a certain period of time) and uses the following 13 primitive relations

$$\{\prec, \succ, m, m^{-1}, o, o^{-1}, s, s^{-1}, f, f^{-1}, d, d^{-1}, \equiv\}$$

to express the relative position of two intervals (Table 1). We will call this *the 13-valued interval algebra* $\mathcal{A}_{13}$. Nökel [31] has observed that these relations can be partially ordered to form a lattice (see Figure 2(I)) which may be used for studying convexity problems.

There are two lines of specialization which we study in this paper, macro relations and restricted domains. Macro relations refers to partitioning the 13 primitive relations

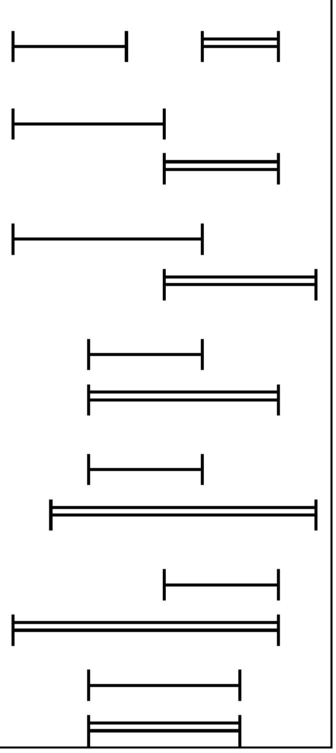
RELATION	NOTATION	INTERPRETATION
x before y	$\prec$	
y after x	$\succ$	
x meets y	m	
y met-by x	m^{-1}	
x overlaps y	o	
y overlapped-by x	o^{-1}	
x starts y	s	
y started-by x	s^{-1}	
x during y	d	
y includes x	d^{-1}	
x finishes y	f	
y finished-by x	f^{-1}	
x equals y	$\equiv$	

Table 1: The 13-valued interval algebra A_{13}.
Single line: x interval. Double line: y interval.

into more coarse relations by regarding a subset of primitive relations as a macro relation. Restricted domains refers to designating explicitly the allowable subsets of relations which may serve as domains, which we will discuss in Section 5.

2.2.1 Macro relations

We let

$$\cap \; = \; \{m, m^{-1}, o, o^{-1}, s, s^{-1}, f, f^{-1}, d, d^{-1}, \equiv\}$$

$$\alpha \; = \; \{m, o\}, \quad \alpha^{-1} \; = \; \{m^{-1}, o^{-1}\}$$

$$\subset \; = \; \{s, f, d\}, \quad \subset^{-1} \; = \; \{s^{-1}, f^{-1}, d^{-1}\}$$

From these we define the 3-valued and 7-valued interval algebras $\mathcal{A}_3$ and $\mathcal{A}_7$ whose

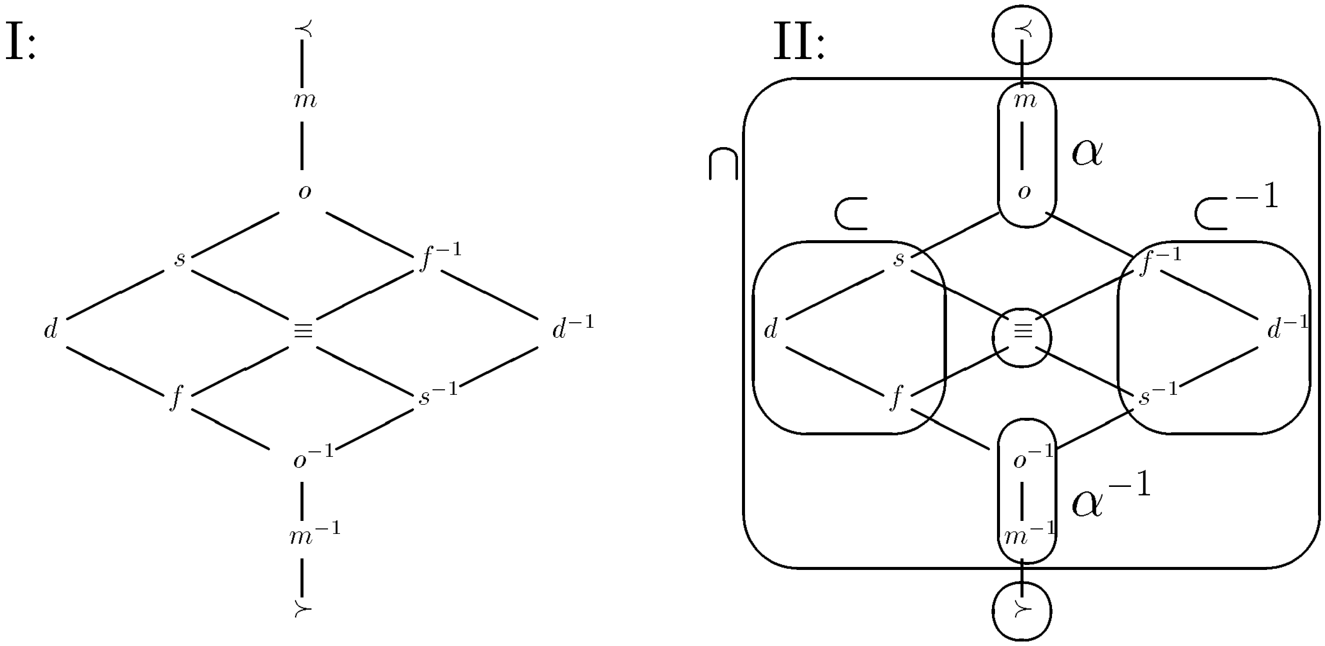


Figure 2: I: The Nökel lattice for $\mathcal{A}_{13}$. II: Macro relations for $\mathcal{A}_3$ and $\mathcal{A}_7$. In the partial order, $R_1 < R_2$ if two intervals x, y satisfying xR_1y can be transformed to satisfy xR_2y by keeping interval x fixed and shifting one or both endpoints of interval y to the right.

elements are called its *atomic relations*, respectively,

$$\mathcal{A}_3 \;:\; \{\prec, \succ, \cap\}$$

$$\mathcal{A}_7 \;:\; \{\prec, \succ, \alpha, \alpha^{-1}, \subset, \subset^{-1}, \equiv\}$$

These macro-algebras are illustrated in Figure 2(II) as overlays on the Nökel lattice. The choice of which of these algebras $\mathcal{A}_i$ to use depends on the nature of the application, data, constraints and on the type of complexity result being proved.

Remark 2.2 Frequently an assumption is made in combinatorics that all interval endpoints are distinct. (This is sometimes accomplished by perturbing the endpoints "by an epsilon" while preserving intersection and disjointness.) This simplification eliminates the seven relations $m, m^{-1}, s, s^{-1}, f, f^{-1}, \equiv$ leaving one with the 6-valued algebra

$$\mathcal{A}_6 \;:\; \{\prec, \succ, o, o^{-1}, d, d^{-1}\}$$

Although this assumption may be inappropriate or too strong in some applications of temporal reasoning, it has proved useful in others. We choose here to allow common endpoints, since it permits a more general analysis, even in the case of $\mathcal{A}_3$.

Remark 2.3 Abusing the terminology slightly, we call each $\mathcal{A}_i$ an algebra, for the following reason: The set of possible relations in each $\mathcal{A}_i$ forms a class closed under sum (set-theoretic union), product (set-theoretic intersection), and converse. Each class contains the empty relation (empty set of relations) and the universal relation (union of all possible relations). Hence, it forms a Boolean algebra. Moreover, an additional operation of composition of relations can be naturally defined (see [2, 29]). If the class also contains the identity relation with respect to composition then it qualifies as a *relation algebra*, in the sense defined by Tarski [37]. For example, Ladkin and Maddux have shown that $\mathcal{A}_{13}$ is a relation algebra [29].

2.3 Temporal reasoning as a constraint satisfiability problem

We now define our temporal reasoning problems in the context of constraint satisfiability. For each pair of events x and y, let $D(x,y)$ be a set of atomic relations in the algebra $\mathcal{A}_i$. 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 $D(x,y)$. (In the language of constraint satisfiability, there is a variable $v(x,y)$ representing the relation between x and y, and its value must be taken from those in the set $D(x,y)$ corresponding to its domain.) For example, we read $D(x,y) = \{\prec, \subset\}$ as x is either before or contained in y. We also call a set of atomic relations a *relation set.* Occasionally we shall use the notation $xD(x,y)y$ for short. For example, $D(x,y) = \{\prec, d^{-1}, \equiv\}$ 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 omit braces when there is no ambiguity, e.g., $x{\prec}y$ or $x{\cap}y$.

An *instance* (or input) for all the problems studied in this paper consists of n events, and a relation set $D(x,y)$ for every pair of distinct events x, y. Without loss of generality, we assume that for each pair of events 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, $R \in D(x,y) \Leftrightarrow R^{-1} \in D(y,x)$. Otherwise, it is a simple matter to restrict the relation sets further so that they satisfy these properties or are shown to be unsatisfiable. Hence, we can assume that the input contains for each pair of events x, y only one relation set, say $D(x,y)$, and that the other relation set $D(y,x)$ is given implicitly by the above rule.

Remark 2.4 The *size* of an instance is the number s of non-trivial relation sets in it, and is linearly equivalent to the binary input length for each $\mathcal{A}_i$. We will assume that relation sets which are trivial (i.e., contain all the atomic relations) are not listed explicitly as part of a problem input, and that each event is involved in at least one non-trivial relation set. Hence, the input size s of a problem with n events satisfies $s = O(n^2)$ and $s = \Omega(n)$.

An *interval realization* for the instance $\{D(x,y)\}$ is an assignment of intervals on the real line to events, so that for each pair of events, one of the atomic relations in their relation set holds. 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. Two realizations are *distinct* if the orders of the endpoints in them differ. The input data $\{D(x,y)\}$ is *consistent* (or *satisfiable*) if it admits at least one realization.

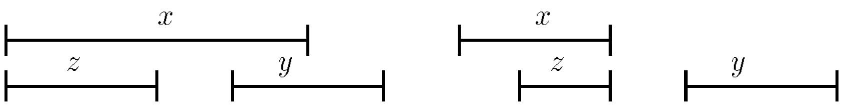
An *instantiation* is an assignment of one atomic relation from $D(x,y)$ to each pair of events x, y, and it is *consistent* if the resulting instance is consistent. A consistent instantiation is also called *a solution*. Clearly, to every interval realization there corresponds a unique instantiation. However, for algebras in which macro relations are defined, more than one interval realization may correspond to the same solution.

The *interval satisfiability problem* (ISAT), called consistent singleton labeling in [39], is determining the existence of (and finding) one instantiation that is consistent with the input data $\{D(x,y)\}$. Occasionally, we shall make an additional distinction between the *decision version* of ISAT, whose output is only a yes/no answer, and the *constructive version*, whose output is the answer 'no' if the input is inconsistent and a solution if it is consistent.

The *minimal labeling problem* (MLP) (as mentioned above) is to determine the minimal sets $D'(x,y) \subseteq D(x,y)$ such that the set of solutions is unchanged, and every remaining atomic relation participates in some solution.

Example 2.5 $x\{\prec, m, o\}y,\ y\{\prec, \equiv, \succ\}z,\ z\{f, s\}x$.
Here $xoy,\ y{\succ}z,\ zsx$, and $x{\prec}y,\ y{\succ}z,\ zfx$ are both consistent with the input, as shown in Figure 3. On the other hand, $y{\prec}z$ and $z \equiv y$ are impossible. The minimal labeling for this problem is $x\{\prec, m, o\}y,\ y{\succ}z,\ z\{f, s\}x$.

The *all solutions problem* (ASP) is that of determining a polynomial representation structure Σ for *all* consistent instantiations. Our contention is that the all solutions problem is a more faithful closure problem for interval algebras than the minimal labeling problem, since not all tuples of the cross product of a minimal labeling are solutions. Consider the following simple example.



I: $xoy, \; y{\succ}z, \; zsx$ II: $x{\prec}y, \; y{\succ}z, \; zfx$

Figure 3: Two interval realizations for Example 2

Example 2.6 $a\{\prec, \succ\}b, \quad b\{\prec, \succ\}c, \quad a\{\prec, \succ\}c.$
This is a minimal labeling, and yet only 6 of the possible 8 instantiations are consistent.

The closely related *all realizations problem* (ARP), is that of enumerating all the distinct interval realizations for a given instance. For $\mathcal{A}_{13}$, ASP and ARP are equivalent, but in the smaller algebras there may be several (or many) distinct endpoint sequences which realize the same instantiation.

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 The relative complexity of the four consistency problems

In this section we show that the polynomiality of the interval satisfiability problem (even in the weaker decision version) implies the polynomiality of MLP and the existence of a polynomial representation structure for ASP. The scope of these results is in fact broader than our representation suggests. They are valid for general constraint satisfiability problems, provided that the maximum number of labels in a domain is bounded by a constant. Another result which is more specific to the interval context is that the polynomiality of ISAT implies also the existence of a polynomial representation structure for ARP. We present here the results for the algebras $\mathcal{A}_i, i = 3, 6, 7, 13,$ and they apply also to any restricted domain in these algebras.

Proposition 3.1 *The minimal labeling problem and the interval satisfiability problem (in the decision and the constructive versions) are polynomially equivalent for each of the algebras* $\mathcal{A}_i, i = 3, 6, 7, 13$.

Proof. Clearly an answer to the MLP gives an answer to the ISAT decision problem. For the converse, one can use an oracle for ISAT (in either version) to solve MLP as follows: Replace one relation set by one of the atomic relations it contains, and keep the rest of the problem input unchanged. ISAT is satisfiable for the resulting problem if and only if that atomic relation is part of a minimal labeling of the original problem. Hence, MLP can be solved by $O(n^2)$ calls to an ISAT oracle.

To complete the equivalence we shall show that given an oracle for MLP and a consistent input I, one can construct in polynomial time a instantiation consistent with the input and thereby solve the constructive version of ISAT: Apply the MLP oracle to input I. Since each atomic relation in the resulting output J is realized in some solution, replacing one non-singleton relation set by a single atom in it creates a new instance J' which is consistent, and whose solution set is contained in that of I. Now rename J' as I and repeat the process until all relation sets are atoms. The final set is the desired instantiation. Hence, the constructive version of ISAT can be solved by $O(n^2)$ calls to an MLP oracle. ■

Proposition 3.2 *In any of the algebras* $\mathcal{A}_i, i = 3, 6, 7, 13$, *if the interval satisfiability problem is polynomial, then there exists a polynomial representation structure for the corresponding all solutions problem.*

Proof. By Proposition 3.1, the polynomiality of ISAT implies the polynomiality of MLP. Denote the answer to the MLP by $D = (D^1, D^2, \ldots, D^k)$, where each $D^i = \{R^i(1), \ldots R^i(n_i)\}$ is a relation set in the minimal labeling, and the atomic relations it consists of are $R^i(j)$. Clearly the size of D is polynomial in the size of the instance problem. The orders of the relation sets and of the atoms in each set may be arbitrary.

The algorithm for solving ASP is an enumeration based on the structure D: It repeatedly replaces relation sets by atomic relations until a feasible instance in which all relations are atomic is produced. The algorithm proceeds in levels, according to the number of relation sets which have already been replaced by atomic relations. In each step, it replaces one D^i by an atomic relation $R^i(j) \in D^i$ and solves ISAT on the resulting problem. If the new problem is satisfiable, then the algorithm moves one level forward in the enumeration: D^{i+1} is also replaced by an atomic relation. If the new problem is not satisfiable, then the next atomic relation from D^i replaces R^i_j. (Call this step a *sideward move.*) If all atomic relations in D^i have already been exhausted, then the algorithm moves one level backward in the enumeration, the next atomic relation in D^{i-1} is fixed, and the original D^i is restored. A realization is determined whenever the algorithm reaches level $k+1$. The algorithm terminates when level 1 has been exhausted. (Viewed differently, the algorithm performs an exhaustive depth first search of the solution space, with a consistency check on the partially restricted solution before each move down the tree. By pruning branches off the tree if ISAT at their root is false, most futile searches are avoided. This pruning is the reason for the polynomiality of the algorithm.)

A formal presentation of the algorithm follows. An auxiliary vector, $a = (a^1, \ldots a^k)$ records the atomic relation to be explored next in each relation set. A constrained instance $D(a) = (\overline{D}^1, \ldots, \overline{D}^k)$ is defined by:

$$\overline{D}^i = \begin{cases} D^i & if\ a^i = 0 \\ R^i(a^i) & if\ a^i > 0 \end{cases}$$

where i records the current level in the algorithm.

algorithm ASP
begin
0. /* *Initialize* */ $a \leftarrow (1, 0, \ldots, 0)$; $i \leftarrow 1$
1. Solve ISAT($D(a)$). **if** FALSE **then** go to 2
else 1.1 **if** $i < k$ **then** /* *move forward* */ $a^{i+1} \leftarrow 1$; $i \leftarrow i+1$; Go to 1
1.2 **else** (ISAT(D(a))=TRUE, $i = k$) output $D(a)$ and continue
2. /* *modify D(a) for the next instance to be checked* */
2.1 **if** $a^i < n_i$ **then** /* *move sidewards* */ $a^i \leftarrow a^i + 1$; Go to 1
2.2 **else** $(a^i = n_i)$ **if** $i = 1$ **then** STOP
else $(a^i = n_i, i > 1)$ /* *move backwards* */ $a^i \leftarrow 0$; $i \leftarrow i - 1$; go to 2.1
end

The correctness of the algorithm is clear. To prove its complexity, note that the algorithm does not move forward to level $j+1$ unless at least one of the atomic relations in D^j participates in some solution. Hence, the algorithm performs at most $k \cdot \beta$ calls to an ISAT oracle for every consistent instantiation it produces, where β is the maximum number of possible atomic relations in any level ($\beta = i$ in $\mathcal{A}_i$). ■

We conclude this section by stating an analogous result showing that ARP is polynomial whenever ISAT is:

Proposition 3.3 *In any of the algebras $\mathcal{A}_i, i = 3, 6, 7, 13$, if ISAT is polynomial, then there exists a polynomial representation structure for the all realizations problem.*

The proof is deferred to Section 5.5, since it will use some of the tools developed in Section 5.

Remark 3.4 The propositions above are included here to point out a theoretical result, and the specific relative complexity bounds are less important. For example, one can replace in the enumeration scheme above the ISAT subroutine by an MLP subroutine. In certain cases, as we shall see, MLP is not harder than ISAT, and the MLP solution restricts the enumeration more efficiently. We shall also see in the following sections some cases where the approach above for MLP and ASP translates into concrete, more efficient algorithms, and others which can be solved more efficiently by different techniques.

4 NP-completeness of the complete algebras

Vilian and Kautz [42] have shown that the interval satisfiability problem is NP-complete for $\mathcal{A}_{13}$. Their proof relies on relations in which endpoints are equal, such as in the *meets* relation. We prove here a stronger theorem that ISAT is NP-complete for $\mathcal{A}_3$, and obtain as a corollary that all four problems ISAT, MLP, ASP and ARP are intractable for all four interval algebras $\mathcal{A}_i,\ i = 3, 6, 7, 13$.

To prove that ISAT is NP-complete for $\mathcal{A}_3$, we introduce a new combinatorial problem, called the *interval graph sandwich problem*, which we prove NP-complete and show to be a special case of ISAT. We first give some necessary background on interval graphs.

Let $\{I(v)\}_{v \in V}$ be a set of intervals on the real line. We can define a strict partial order $\prec$ on V where for any v, w in V we have $v \prec w$ if and only if interval $I(v)$ is strictly to the left of interval $I(w)$. The partial order implied from this definition is the *interval order* and $\{I(v)\}_{v \in V}$ is called an *interval representation.* An undirected graph $G = (V, E)$ is called an *interval graph* if its vertices v can be represented by intervals $I(v)$ on the real line such that two vertices are adjacent if and only if the corresponding intervals intersect. Thus, an interval graph is the *incomparability* graph of an interval order. (For further references on interval orders and interval graphs the reader should see [15, 16, 18, 20, 21].)

For a given interval representation $\{I(v)\}_{v \in V}$ we define its *endpoint-sequence* to be sequence of *endpoints* as we scan left to right along the real line. We denote by l_v the left-endpoint (*startpoint*) of interval $I(v)$ and by r_v the right-endpoint (*finishpoint*) of the same interval, i.e., $I(v) = [l_v, r_v]$. Generally, there may be several topologically different interval representations for a given interval order or interval graph and in the worst case exponentially many. This lack of specificity is inherent in much of temporal reasoning and indicates partial knowledge about the actual occurrences of the temporal events.

Let E^1 and E^2 be two disjoint sets of edges defined on the same vertex-set V. A graph $G(V, E)$ with $E^1 \subseteq E \subseteq E^1 \cup E^2$ is called a *sandwich graph* for (E^1, E^2). The interval graph sandwich (IGS) problem is the following:

> **Interval Graph Sandwich problem:**
> INPUT: Two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$ with the same set of vertices, and disjoint edge-sets E^1 and E^2.
> QUESTION: Is there a sandwich graph for (E^1, E^2) which is an interval graph?

Define $F = \overline{\{E^1 \cup E^2\}}$ (i. e., F is the set of non-edges in the graph $(V, E^1 \cup E^2)$.) When $E^1 = \emptyset$ or $F = \emptyset$, the answer is trivially yes. When $E^2 = \emptyset$, the problem is polynomial by the algorithm of Booth and Lueker [9]. We shall show that in the general case, the problem is NP-complete.

An application in molecular biology: Interval sandwich problems arise in various practical contexts. One example is in physical mapping of DNA in molecular biology [11]. In this problem, 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 linear 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 [7], 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 if their segments are known to intersect. If complete information is available, then for each pair of segments it is known whether they intersect or not. In that case, $E^2 = \emptyset$, so the decision question is efficiently solvable. However, information on the intersections of segments is often known only in part, because of lack of experimental data or inconclusive experimental results. That ambiguity introduces E^2-edges into the graph. In that case, the decision problem is equivalent to the IGS problem.

We need two more concepts before we can prove the main result of this section. First, three vertices x, y, z in the graph $G = (V, E)$ are called an *asteriodal triplet* if no two of them are connected by an edge, and for each two of the three, there is a path connecting them which does not pass through any vertex adjacent to the third. Lekkerkerker and Boland observed [30] that an interval graph cannot contain an asteriodal triplet. Second, the Not-All-Equal 3-Satisfiability (NAE-3SAT) problem is a restriction of the 3-Satisfiability problem in which one asks for a truth assignment such that each clause contains at least one true literal and at least one false literal. Schaefer [35] has shown that NAE-3SAT is NP-complete. Note that in this problem we can assume without loss of generality that no clause contains a variable and its negation.

Theorem 4.1 *The interval graph sandwich problem is NP-Complete.*

Proof. The problem is clearly in NP since a given interval representation can be verified to fit the input in polynomial time. We describe a reduction from NAE-3SAT: Let Φ be a CNF-formula with variables $X_1, \ldots, X_n$ and clauses $C_1, \ldots, C_m$. We construct an instance (V, E^1, E^2) of the IGS problem as follows:

1. Define a vertex p. For each variable $X_i, i = 1, \ldots, n$, define four vertices $x_i, \overline{x}_i, x'_i$, $\overline{x}'_i$. The vertices $x'_i, \overline{x}'_i$ are called *the private vertices* of variable X_i. The vertices $x_i, \overline{x}_i$ are called *the literal vertices* of X_i. Connect the four vertices to each other and to p to form a *variable subgraph* as shown in figure 4(I).

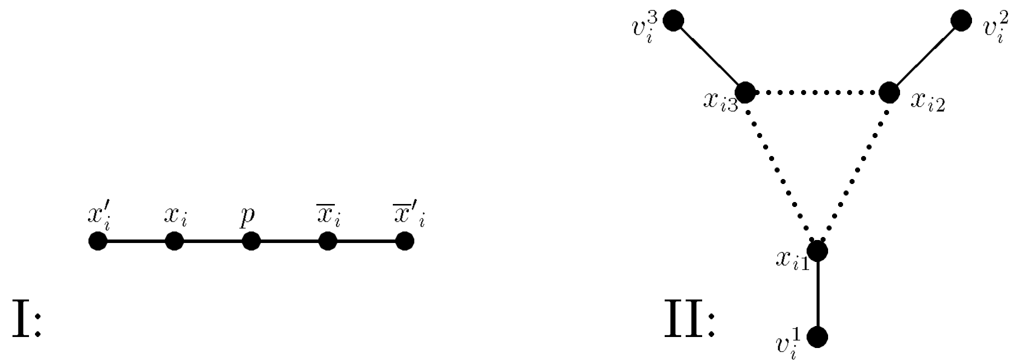


Figure 4: I: variable subgraph. II: clause subgraph (Solid edges are E^1-edges, dotted edges are E^2-edges. Non-edges are in F.)

2. For each clause $C_i = [X_{i1} \bigvee X_{i2} \bigvee X_{i3}]$, $i = 1, \ldots, m$, define three vertices v_i^1, v_i^2, v_i^3, which will be called the *private vertices* of clause C_i. Denote the literal vertices corresponding to the three literals of that clause by x_{i1}, x_{i2} and x_{i3}. Connect these six vertices to form a *clause subgraph*, as shown in figure 4(II).

3. The set F consists exactly of those edges between vertices from the same clause or variable subgraph, which are neither in E^1 nor in E^2 in that subgraph. All the edges between two distinct subgraphs which were not determined by (1) or (2) are E^2-edges. Specifically, all the following are E^2-edges:

3.1 For $i = 1, \ldots, n$, connect each of the vertices $x_i, \overline{x}_i, x'_i, \overline{x}'_i$ to each of $x_j, \overline{x}_j, x'_j, \overline{x}'_j$, $j \neq i$.

3.2. For $i = 1, \ldots, m$, connect each private vertex of clause C_i to each private vertex of clause C_j, $i \neq j$.

3.3. For $i = 1, \ldots, m$, connect each private vertex of clause C_i to p and to each literal vertex except those of the two other literals in clause C_i.

3.4. For $i = 1, \ldots n$, connect each private vertex of variable X_i to each private vertex of clause $C_j, j = 1, \ldots m$.

Clearly this construction requires polynomial time. Let us now prove its validity: Suppose first that there exists an interval graph sandwich G' for the problem. Fix a realization of G', and denote the interval corresponding to vertex v in the realization by $I(v)$. Denote the endpoints of interval $I(p)$ by p_1 and p_2. By the construction of the variable subgraph (part 1), for each variable X_i, $i = 1, \ldots, n$ the intervals $I(x_i)$ and $I(\overline{x}_i)$ are disjoint, and either $p_1 \in I(x_i)$ and $p_2 \in I(\overline{x}_i)$, or $p_2 \in I(x_i)$ and $p_1 \in I(\overline{x}_i)$. Assign truth values to the variables as follows: $v(X_i) = TRUE$ if and only if $p_1 \in I(x_i)$. By the above argument, exactly one literal for each variable is true.

Consider now the clause $C_i = [X_{i1} \bigvee X_{i2} \bigvee X_{i3}]$. The intervals $I(x_{i1}), I(x_{i2}), I(x_{i3})$ cannot be pairwise disjoint, since each of them contains either p_1 or p_2. By the construction of the clause subgraph (part 2), the three intervals cannot have a non-empty intersection, since then in the subgraph corresponding to that clause, v_i^1, v_i^2, v_i^3 will form an asteriodal triplet, which is impossible since the graph is interval. Hence, either one or two of the intervals contains p_1, and so one or two of the literals in each clause are true, as required in NAE-3SAT.

Next, suppose v is a NAE truth assignment which satisfies the formula Φ. We shall prove that there exists an interval graph sandwich for (E^1, E^2) by creating a realization for it:

(a) Choose an interval $I(p) = [p_1, p_2]$ arbitrarily, and fix some point p_3 inside it, i.e., $p_1 < p_3 < p_2$. Define intervals $A_1 = [t_1^1, t_1^2]$ and $B_1 = [f_1^1, f_1^2]$ by choosing points $t_1^1, t_1^2, f_1^1, f_1^2$ satisfying

$$t_1^1 < p_1 < t_1^2 < p_3 < f_1^1 < p_2 < f_1^2$$

For $i = 2, \ldots, n$ define inductively intervals A_i, B_i by choosing points $t_i^1, t_i^2, f_i^1, f_i^2$ such that

$$A_i = [t_i^1, t_i^2] \text{ where } t_i^1 < t_{i-1}^1 < p_1 < t_i^2 < t_{i-1}^2$$

$$B_i = [f_i^1, f_i^2] \text{ where } f_{i-1}^1 < f_i^1 < p_2 < f_{i-1}^2 < f_i^2$$

An example of the construction of these intervals is given in figure 5.

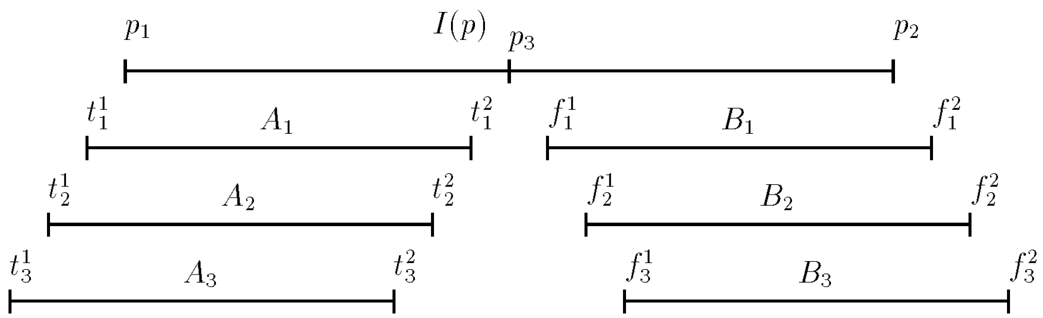


Figure 5: Construction of the A and B intervals.

(b) If $v(X_i) = TRUE$, then set $I(x_i) = A_i$ and $I(\overline{x}_i) = B_i$. Otherwise, set $I(x_i) = B_i$ and $I(\overline{x}_i) = A_i$. This guarantees that $I(x_i)$ and $I(\overline{x}_i)$ are disjoint, and that one of them contains p_1 and the other contains p_2, as prescribed by (1). The intervals of the private vertices of each variable can now easily be placed so as to satisfy (1).

(c) The arrangement of the literal intervals in part (a) above guarantees that for every two such intervals which have non-empty intersection, no interval is contained in the other. Since v is a not-all-equal truth assignment, for every clause at most two literals are true and at most two are false. Hence, the intervals of the private vertices of each clause can be placed so as to satisfy (2) (see the example in figure 6).

The conditions in (3) are automatically satisfied by the above realization, since they pose no additional restrictions. ∎

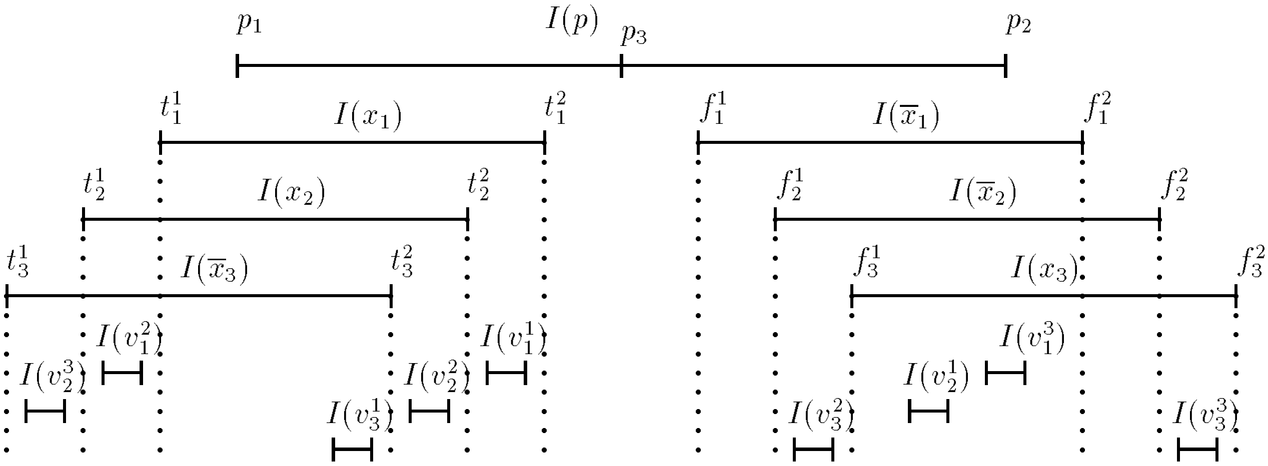


Figure 6: Example of an interval realization for the formula $F = [X_1 \bigvee X_2 \bigvee X_3][\overline{X}_1 \bigvee X_2 \bigvee \overline{X}_3][X_1 \bigvee \overline{X}_2 \bigvee X_3]$ with not-all-equal truth assignment $v(X_1) = v(X_2) = v(\overline{X}_3) = TRUE$. For simplicity, private intervals of variables are omitted.

Theorem 4.2 *ISAT is NP-complete for $\mathcal{A}_3$.*

Proof. For a given instance of the interval graph sandwich problem, we construct an equivalent formulation as an ISAT problem for $\mathcal{A}_3$ as follows:

For each edge $(x, y) \in E^1$, let $D(x, y) = \{\cap\}$.
For each edge $(x, y) \in E^2$, let $D(x, y) = \{\prec, \cap, \succ\}$.
For each pair $(x, y) \notin E^1 \cup E_2$, let $D(x, y) = \{\prec, \succ\}$.

It is clear that this ISAT problem has a solution if and only if there is an interval graph sandwich. ■

Corollary 4.3 *ISAT, MLP, ASP and ARP are NP-hard for $\mathcal{A}_3$, $\mathcal{A}_6$, $\mathcal{A}_7$ and $\mathcal{A}_{13}$.*

Proof. This follows from the observation that the algebra $\mathcal{A}_3$ is contained in each $\mathcal{A}_i$ and that for any $i = 3, 6, 7, 13$, each of the four problems has a non-empty solution if and only if any of them has one. ■

5 Restricted domain problems

Because of the intractability of the general versions of the consistency problems, attention has been focused on the work of several authors who have studied polynomial time approximation algorithms for MLP on $\mathcal{A}_{13}$ [2, 39, 40]. Previous to the work of Belfer and Golumbic [4, 5, 6], we do not know of any study which has investigated the ARP. Solutions to several restricted cases of the interval satisfiability problem have been known for a long time. These will be extended by the new results presented in this section.

By suitably restricting the input domain of an NP-complete problem, one can often obtain a special class which admits a polynomial time algorithm. In the general case for an interval algebra $\mathcal{A}_i$, each relation set $D(x,y)$ may take any of $2^i - 1$ possible values (the power set formed by the atoms of $\mathcal{A}_i$, excluding the empty subset of relations). In this section, we restrict this by designating Δ to denote a particular family of relation sets in $\mathcal{A}_i$ and requiring that each set $D(x,y)$ be a member of Δ. To simplify notation we shall represent each relation set in $\mathcal{A}_3$ 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 $\mathcal{A}_3$ in this notation are:

$$\prec, \succ, \cap, \prec\cap, \cap\succ, \prec\succ, \prec\cap\succ.$$

ISAT(Δ) will denote the ISAT problem where all the relation sets are restricted to the set Δ. The proof of Theorem 4.2 shows that even when all relation sets are restricted to be from $\Delta_0 = \{\cap, \prec\succ, \prec\cap\succ\}$ (meaning *intersect, disjoint* or *don't care*), ISAT remains NP-complete. This is restated in the following corollary:

Corollary 5.1 *ISAT(Δ_0) is NP-complete.* ■

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 Table 2 along with their appropriate Δ, see [15, 20, 34].

Class	Restricted Domain	Reference
Interval orders	$\Delta = \{\prec, \succ, \cap\}$	[14]
Interval graphs	$\Delta = \{\prec\!\succ, \cap\}$	[18, 17, 9, 28]
Circle (or overlap) graphs	$\Delta = \{\{\alpha, \alpha^{-1}, \equiv\}, \{\prec, \succ, \subset, \subset^{-1}\}\}$	[19, 8]
Interval containment graphs	$\Delta = \{\{\subset, \subset^{-1}\}, \{\prec, \succ, \alpha, \alpha^{-1}, \equiv\}\}$	[22]
Posets of dimension 2	$\Delta = \{\{\subset\}, \{\subset^{-1}\}, \{\prec, \succ, \alpha, \alpha^{-1}, \equiv\}\}$	[13, 3, 22]

Table 2: Five cases of interval satisfiability problems in graph theory which admit polynomial time solutions.

The results of [4, 5, 6] demonstrate polynomial time solutions for the ARP in $\mathcal{A}_3$ restricted to (i) $\Delta = \{\prec, \succ, \cap\}$ (interval orders) using the so called Π structure and its associated construction algorithms, and (ii) $\Delta = \{\prec\!\succ, \cap\}$ (interval graphs) using the endpoint-tree structure and its construction algorithms.

The interest in restricted domains is both practical and theoretical. Some practical applications in which restricted domains naturally arise are medical databases, medical diagnosis and natural language processing (see [41, p. 328] and the references thereof), circuit design [43, p. 184], and physical mapping in molecular biology (see the discussion in the preceding section). Also, the numerous applications of interval graphs and interval orders [15, 20, 34] can all be viewed as applications of restricted domains. As we shall discuss in Section 6, knowing which restricted domains are polynomial can also speed up the enumeration in NP-hard domains.

In this section we shall study the complexity of ISAT on four additional restricted domains in $\mathcal{A}_3$. Using these results, we shall obtain a complexity classification of all but four of the restricted domains in $\mathcal{A}_3$.

5.1 Algorithms for the domain $\mathcal{A}_3 - \prec\!\succ$.

In this section we deal with problems in the domain:

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

That is, $\mathcal{A}_3 - \prec\!\succ$. We shall give efficient algorithms for ISAT, MLP and ARP on this domain, and they will apply immediately to any subdomain of Δ_1. Hence, by excluding just a *single* relation set from $\mathcal{A}_3$ the problems become tractable.

5.1.1 A linear time algorithm for ISAT(Δ_1)

The polynomial algorithm for ISAT(Δ_1) utilizes a directed graph, which will be described later, with vertices corresponding to the endpoints of event intervals and arcs representing the relative order of endpoints. The key observation here is that every relation in Δ_1 is equivalent to a certain *order requirement* between a pair (or two pairs) of such endpoints. This observation is proved below:

Lemma 5.2 *Let i and j be the event intervals $[l_i, r_i]$ and $[l_j, r_j]$, respectively. In each of the following cases, the intervals satisfy the set of relations if and only if their endpoints satisfy the corresponding inequalities:*

$$i{\prec}j \;\Leftrightarrow\; r_i < l_j \qquad\qquad i{\succ}j \;\Leftrightarrow\; r_j < l_i \tag{1}$$

$$i{\prec}{\cap}j \;\Leftrightarrow\; l_i \le r_j \qquad\qquad i{\cap}{\succ}j \;\Leftrightarrow\; l_j \le r_i \tag{2}$$

$$i{\cap}j \;\Leftrightarrow\; l_i \le r_j \text{ and } l_j \le r_i \tag{3}$$

If $i{\prec}{\cap}{\succ}j$, no constraint is imposed.

Proof. (1) is immediate. The conditions in (2) are the negation of those in (1). (3) is equivalent to the intersection of the two conditions in (2). ∎

The graph is now constructed as follows: For an instance J of ISAT(Δ_1) with n events, form a directed graph $G(J) = G(V, E)$ with vertex set $V = \{r_1, \ldots r_n, l_1, \ldots l_n\}$. The arc set E consists of two disjoint subsets, E_0 and E_1. The former will represent weak inequality and the latter will represent strict inequality between pairs of endpoints. The arcs are defined as follows:

$$(l_i, r_i) \in E_0 \qquad\qquad i = 1, \ldots, n \tag{4}$$

$$(r_i, l_j) \in E_1 \qquad\qquad \forall i, j \text{ s.t. } i{\prec}j \tag{5}$$

$$(l_i, r_j) \in E_0 \qquad\qquad \forall i, j \text{ s.t. } i{\prec}{\cap}j \tag{6}$$

$$(l_i, r_j) \in E_0 \text{ and } (l_j, r_i) \in E_0 \qquad\qquad \forall i, j \text{ s.t. } i{\cap}j \tag{7}$$

For pairs i, j with the relation $i{\prec}{\cap}{\succ}j$, no arc is introduced. Define now $E = E_0 \cup E_1$. We call the arcs in E_0 (resp., E_1) the *weak arcs* (resp., *strict arcs*). Note that the graph G is bipartite. Denote the two parts of the vertex set by $R = \{r_1, \ldots, r_n\}$ and $L = \{l_1, \ldots, l_n\}$, and call an arc an *RL-arcs* (resp., *LR-arc*) if it is directed from R to L (resp., from L to R).

Remark 5.3 In the graph G all the RL-arcs are strict and all the LR-arcs are weak. Hence, we need not record explicitly the type of each arc since it is implied by its direction.

Lemma 5.4 *Every cycle in G contains a strict arc.*

Proof. Since G is bipartite, a cycle must contain vertices both from R and from L. In particular, it must contain an RL-arc, so the claim follows by Remark 5.3. ■

An algorithm for solving ISAT(Δ_1) is described below:

algorithm Δ_1-CONSISTENCY;
begin
1. construct $G(J)$ according to rules (4)-(7).
2. if $G(J)$ contains a cycle - J is not satisfiable. Otherwise, it is.

end

Lemma 5.5 *Suppose $G(J) = (V, E)$ is acyclic. Then a linear order on V is consistent with the partial order $G(J)$ if and only if it is a realization of J.*

Proof. Take any linear order P which extends the partial order G. P is an ordering of all the endpoints, which by Lemma 5.2 satisfies all the relations in the input, so P gives a realization for J. On the other hand, every realization of J gives a linear order of the endpoints, in which each of the input relations must be satisfied, by Lemma 5.2. Hence, the linear order must be consistent with the partial order $G(J)$. ■

Theorem 5.6 *Algorithm Δ_1-CONSISTENCY correctly recognizes if an instance of ISAT(Δ_1) is satisfiable in linear time.*

Proof. *Validity:* By Lemma 5.2, each arc reflects the order relation of a pair of interval endpoints as prescribed by the input relations. If G contains a cycle, then by Lemma 5.4 that cycle contains a strict arc. Hence, that cycle must satisfy $r_i < l_j \leq \ldots \leq r_i$, which implies that the input relations cannot be satisfied. In case G is acyclic, Lemma 5.5 implies that J is satisfiable.

Complexity: Constructing $G(J)$ requires $O(s)$ steps, where s is the number of input relation sets, since the effort is constant per relation. Checking if G is acyclic can be done, for example, by depth first search, in time linear in $|E|$, the number of arcs [38]. Since $|E| \leq 2s + n$, using remark 2.4 we conclude that the algorithm is linear. ■

Remark 5.7 Skrien [36] studied the question of chronologically ordering an interval graph, and defined a similar graph in his analysis. He assumes that the underlying graph is known to be interval, and is concerned with the question of whether a partial orientation of the endpoints can be extended to a transitive one. However, since he works on the complete graph and not only on the bipartite subgraph, and does not assume (4), his approach leads to a significantly more complicated characterization and proof, and his algorithm has higher complexity.

Remark 5.8 Ladkin and Maddux [29] and van Beek [41] independently studied a restricted domain of $\mathcal{A}_{13}$ called "pointisable relations" in [29], or "SIA" in [41]. Viewed as a subset of $\mathcal{A}_{13}$, Δ_1 is contained in this domain. On that domain, ISAT is solvable in $O(n^2)$ steps using the algorithm for the "point algebra", in which all intervals are assumed to be single points [40]. Nökel [31] has also studied a subset of that domain.

Remark 5.9 Since all the constraints defining an instance of ISAT(Δ_1) are linear inequalities, the satisfiability problem can be reformulated as a feasibility question for a system of linear inequalities. This can be solved by linear programming algorithms, and in fact, by specialized algorithms using the fact that only two variables appear in each inequality [24]. While this is less efficient than the method described above, it allows the natural introduction of additional linear constraints, outside the scope of $\mathcal{A}_3$ or even $\mathcal{A}_{13}$, like lengths of intervals, fixing endpoints to specific time values, etc.

5.1.2 The Minimal Labeling Problem

Once $G(J)$ has been constructed and shown to be acyclic, the MLP can be solved by forming the transitive closure of $G(J)$, deducing from it additional (weak and strict) orders of endpoints, and then using the equivalence established in Lemma 5.2 between these orders and interval relations to create the minimal labeling. The algorithm is formally given below:

algorithm MIN-LABEL(J);
begin
1. Compute $G^* = (V, E^*)$, the transitive closure of $G(J)$.
2. for each pair i, j **do** :
   - 2.1. **if** $(r_i, l_j) \in E^*$ **then** set $i \prec j$; **if** $(r_j, l_i) \in E^*$ **then** set $i \succ j$
     else ($(r_i, l_j) \notin E^*$, $(r_j, l_i) \notin E^*$)
   - 2.2. **if** $(l_i, r_j) \in E^*$ and $(l_j, r_i) \notin E^*$ **then** set $i {\prec}{\cap} j$;

if $(l_i, r_j) \notin E^*$ and $(l_j, r_i) \in E^*$ **then** set $i{\cap}{\succ}j$
2.3. **if** $(l_i, r_j) \in E^*$ and $(l_j, r_i) \in E^*$ **then** set $i{\cap}j$
2.4 **else** ($(l_j, r_i) \notin E^*$ and $(l_i, r_j) \notin E^*$) set $i{\prec}{\cap}{\succ}j$

end

Let us call the new instance resulting from the procedure above J^*. Note that in general the transitive closure is not bipartite, but the algorithm uses only the RL-arcs and the LR-arcs of the transitive closure. These arcs are sufficient to solve the MLP.

Lemma 5.10 *For every RL-arc (r, l) in E^* there exists a path from r to l in G which contains a strict arc.*

Proof. If $(r, l) \in E^*$ then there exists a path from r to l in the original graph G. Since that path must contain at least one RL-arc in E, the claim follows by Remark 5.3. ■

For the statement of the next theorem, let $T(n, m)$ be the complexity of computing the transitive closure of an acyclic directed graph with n-vertices and m arcs. Recall that the transitive closure can be computed in time $O(nm)$, by using, for example, Warshall's algorithm. For dense graphs, the algorithms using fast Boolean matrix multiplication are asymptotically faster (see [1]). As before, s is the number of non-trivial relation sets in the input.

Theorem 5.11 *Algorithm* MIN-LABEL *computes a minimal labeling of an instance in Δ_1 in $T(n, s)$ steps.*

Proof. *Validity:* We have to show that J^* is a minimal labeling of J, i.e., (A) J and J^* have the same set of solutions, and (B) every atomic relation in J^* holds in some solution:

By Lemma 5.10 all the RL-arcs of G^* correspond to strict order of the corresponding endpoints. We shall assume that all its LR-arcs correspond to weak inequalities. Hence, the bipartite subgraph $G^*_B = (V, E^*_B)$ where $E^*_B = E^* \cap \{\{R \times L\} \cup \{L \times R\}\}$ has exactly the same structure of the original graph G, i.e., it satisfies Remark 5.3. In particular the results of the previous section can be used to infer the interval relations from the endpoint inequalities. Moreover, G^*_B and the corresponding instance J^* satisfy Lemma 5.5.

(A) Since all the constraints of J are satisfied in J^*, every solution of J^* is also a solution of J. By Lemma 5.5, every solution of J corresponds to a linear order P which is consistent with the partial order G. By transitivity the linear order P is also consistent with G^*. Using Lemma 5.5 and Lemma 5.2 we conclude that P is a corresponding solution of J^*.

(B) If the relation of i and j in J^* has been determined in steps 2.1 or 2.3 of the algorithm, then i, j are related by exactly one atomic relation, and that relation must be satisfied in every solution of J. Suppose the relation of i, j was determined in step 2.2, and $(l_i, r_j) \in E^*$ and $(l_j, r_i) \notin E^*$. To show that linear orders with $l_j \leq r_i$ and $l_j > r_i$ are both possible, it suffices to show that l_j and r_i are incomparable in G^*. We already know that $(l_j, r_i) \notin E^*$. Since the relation was not determined in step 2.1, $(r_i, l_j) \notin E^*$, the result follows. The proof for the other cases is similar.

Complexity: Step 1 requires $T(n, s)$ operations. Step 2 requires constant time per pair, for a total of $O(n^2)$ operations. Since $T(n, s) = \Omega(n^2)$, the result follows. ■

Remark 5.12 The transitive closure operation in algorithm MIN-LABEL contains additional information which we do not need to solve the MLP. For example, we may obtain relations of the type $l_i < l_j$, $l_i < r_j$ or $r_i < r_j$. While this kind of information is outside the scope of $\mathcal{A}_3$, it may be useful for other purposes.

5.1.3 The All Realizations Problem

We finish this section by sketching a simple procedure to solve ARP(Δ_1): The graph G^* generated in the previous section corresponds to a partially ordered set. The question thus reduces to constructing all the linear orders consistent with a partial order. This can be done by placing a minimal element in all possible positions with respect to previously ordered elements and repeating recursively. This requires at most $O(n)$ steps per order produced. The distinction between strict and weak inequalities (following from the original distinction between strict and weak arcs) can also be maintained in such a procedure.

5.2 The restricted domain $\{\prec, \succ, \cap, \prec\!\succ\}$.

Graph theoretic techniques provide a proof of the next new result. The following classical theorem characterizing interval graphs will be needed.

Theorem 5.13 *(Gilmore and Hoffman [18]) G is an interval graph if and only if G is a chordal graph and its complement $\overline{G}$ is transitively orientable.*

An undirected graph is *chordal* if for every cycle of length greater than or equal to 4 there is an edge (chord) connecting two vertices which are not consecutive in the cycle. Chordal graphs are also called triangulated graphs. The complement $\overline{G}$ of G is the undirected graph whose edges are the non-edges of G; a graph is *transitively orientable* (TRO) if each undirected edge can be assigned a direction so that the resulting orientation satisfies the usual transitivity property. Transitively orientable graphs are also called comparability graphs, (see [20]).

Theorem 5.14 *ISAT is solvable in $O(n^3)$ time for $\Delta_2 = \{\prec, \succ, \cap, \prec\!\!\succ\}$*

Proof. By our assumption that in the input satisfies $R \in D(x,y) \Leftrightarrow R^{-1} \in D(y,x)$, we may assume that for each pair of elements x and y exactly one of the following holds:

(i) $x \cap y$ (ii) $x \prec\!\!\succ y$ (iii) $x \prec y$ (iv) $x \succ y$.

We construct two complementary graphs G and H whose vertices correspond to the events as follows. The graph $G = (V, E)$ is undirected where

$$\{x,y\} \in E \;\Leftrightarrow\; x \cap y.$$

The graph $H = (V, E')$ has both directed and undirected edges where

$$\{x,y\} \in E' \;\Leftrightarrow\; x \prec\!\!\succ y$$

$$(x,y) \in E' \;\Leftrightarrow\; x \prec y$$

(An undirected edge between x and y is denoted here by $\{x,y\}$, and a directed edge from x to y by (x,y).) It is an easy consequence of Theorem 5.13, that ISAT has a solution if and only if G is chordal and H has a transitive orientation.

Testing whether G is chordal can be done in $O(|V| + |E|)$ time [9], and obtaining a transitive orientation for H can be achieved in $O(|V||E|)$ time by the following variant of the TRO algorithm [20, p. 124] for undirected graphs.

Beginning with the undirected graph $\overline{G}$, we must find a transitive orientation F which is consistent with the directed edges in H. The following definitions are needed:

Let $\overline{G} = (V, \overline{E})$. An oriented edge (x, y) *directly forces* every edge $\{w, y\} \in \overline{E}$ such that $\{x, w\} \notin \overline{E}$ to the orientation (w, y), and every edge $\{x, z\} \in \overline{E}$ such that $\{y, z\} \notin \overline{E}$ to the orientation (x, z). By applying this operation further, (x, y) *indirectly forces* edges which are directly forced by those already forced. The *implication class* B of a directed edge (x, y) consists of all orientations of edges which are forced - directly or indirectly - by the orientation (x, y). For a discussion and more details, see [20, p. 105-109].

begin

Initialize: $F \leftarrow \emptyset$

1. Arbitrarily choose a directed edge (x, y) from $H - F$. If none exists, go to step 4.

2. Enumerate the implication class B of $\overline{G}$ containing (x, y).
If B contains any oriented edge which is opposite to its direction in H,
then H has no transitive orientation; STOP, with failure. Otherwise, continue.

3. Let $F \leftarrow F \cup B$. Delete all undirected edges from $\overline{G}$ which were oriented in B.
Go to step 1.

4. Continue with the usual TRO algorithm for undirected graphs,
which is repeated here as steps 5-7 for completeness.

5. Arbitrarily choose an orientation (x, y) for an arbitrary edge remaining in $\overline{G}$.
If none exists, then all edges are oriented and F is the desired
transitive orientation; STOP, with success. Otherwise, continue.

6. Enumerate the implication class B of $\overline{G}$ containing (x, y).
If B contains any edge which is oriented in both directions,
then there is no transitive orientation; STOP, with failure. Otherwise, continue.

7. Let $F \leftarrow F \cup B$. Delete all undirected edges from $\overline{G}$ which were oriented in B.
Go to step 5.

end

The proof of correctness and polynomial complexity of this variation is almost identical with that of the original TRO algorithm, and is therefore omitted. ■

For a different approach to the same problem, see [27].

5.3 The domain $\{\prec, \succ, \prec\!\cap\!\succ, \prec\!\!\diamond\!\!\succ\}$.

Theorem 5.15 *ISAT is solvable in linear time for* $\Delta_3 = \{\prec, \succ, \prec\!\cap\!\succ, \prec\!\!\diamond\!\!\succ\}$.

Proof. Form a directed graph $G(V, E)$ with vertices corresponding to events, and $(u, v) \in E$ if $u \prec v$. If G contains a cycle, then the instance is clearly not satisfiable. If G is acyclic, then one can create an interval realization of G in which (1) all intervals are disjoint, and (2) $(u, v) \in E$ if $u \prec v$. This can be done by taking any linear extension of G and ordering the intervals so that they are disjoint and ordered according to that order. In the resulting realization, (1) and (2) are satisfied. (2) guarantees that all $\prec$ and $\succ$ relations in the input are satisfied, and (1) guarantees that all other relations ($\prec\!\succ$ and $\prec\!\cap\!\succ$) are satisfied. The linear complexity follows by the same argument as for Theorem 5.6. ∎

5.4 The domain $\{\prec\!\cap, \cap\!\succ, \prec\!\succ, \prec\!\cap\!\succ\}$.

Lemma 5.16 *Let i, x, y, j be events satisfying $i \prec\!\cap j$, $j \prec\!\cap x \prec\!\cap i$, $j \prec\!\cap y \prec\!\cap i$, and $x \prec\!\succ y$. Then in every realization of these relations $i \cap j$.*

Proof. Suppose $i \prec j$ in some realization. Denote by g an interval in that realization satisfying $i \prec g \prec j$. Since $i \prec j$, in order to satisfy $j \prec\!\cap x \prec\!\cap i$, one must have $x \cap i$ and $x \cap j$. This in turn implies that $g \subset x$. Since $j \prec\!\cap y \prec\!\cap i$, the same argument implies that $g \subset y$. Hence, $x \cap y$, a contradiction. ∎

Theorem 5.17 *ISAT is NP-complete for $\Delta_4 = \{\prec\!\cap, \cap\!\succ, \prec\!\succ, \prec\!\cap\!\succ\}$.*

Proof. Clearly ISAT(Δ_4) is in NP. Let $\Delta_5 = \{\prec\!\cap, \cap\!\succ, \prec\!\succ, \prec\!\cap\!\succ, \cap\}$. Since ISAT($\Delta_0$) is NP-complete by Theorem 4.1, and since $\Delta_0 \subset \Delta_5$, ISAT($\Delta_5$) is also NP-complete. We shall give a polynomial reduction from an instance I of ISAT(Δ_5) to an instance I' of ISAT(Δ_4):

If in instance I events i and j are related by any one of the relations $i \prec\!\cap j$, $i \cap\!\succ j$, $i \prec\!\succ j$, or $i \prec\!\cap\!\succ j$, the same relation is transferred to I'. For every relation $i \cap j$, define auxiliary events x_{ij} and y_{ij}, and introduce the following relations in I':

1. $i \prec\!\cap j$, $j \prec\!\cap x_{ij} \prec\!\cap i$, $j \prec\!\cap y_{ij} \prec\!\cap i$, and $x_{ij} \prec\!\succ y_{ij}$.

2. $k \prec\!\cap\!\succ x_{ij}$ and $k \prec\!\cap\!\succ y_{ij}$ for every event $k \neq i, j, x_{ij}, y_{ij}$ (including other auxiliary events).

Clearly this reduction is polynomial.

Suppose I' is satisfiable. Then all those relations which were transferred from I without change are satisfied in an interval realization of I'. In addition, for every pair i, j in I with relation $i \cap j$, all relations of the types (1)-(2) above are satisfied. From (1) and Lemma 5.16, $i \cap j$ is satisfied as required. Hence, the intervals in I' which correspond to the original events form a realization of I.

Suppose now I is satisfiable. Then there is an interval realization in which all endpoints are distinct. (Every realization can be perturbed into one in which all endpoint are distinct without changing any relation in $\mathcal{A}_3$: Each set of identical endpoints can be strictly ordered arbitrarily as long as all the startpoints in the set precede all the finishpoints in it.) Take such a distinct-endpoints interval realization of I and extend it to form an interval realization of I' by forming the auxiliary intervals as follows: For each relation $i \cap j$ in the realization, by the distinct-endpoints property, the intersection of the intervals of i and j is an interval (and not only a single point). Hence, the intervals of events x_{ij} and y_{ij} can be placed so that they are disjoint and contained in the intersection of the intervals of i and j (see figure 7). This guarantees that all relations of type (1) are satisfied. Type (2) relations are trivially satisfied by any interval realization. ■

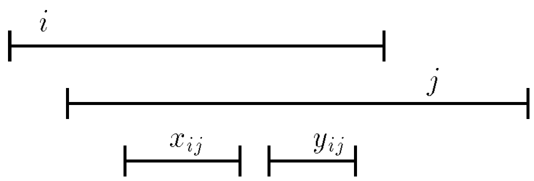


Figure 7: an example.

5.5 Proof of Proposition 3.3

We shall show that given a solution, (in which all relation sets are *singletons* of the algebra,) one can generate all the interval realizations corresponding to it using a polynomial representation structure Σ'. By Proposition 3.2, polynomiality of ISAT implies the existence of a polynomial representation structure Σ for ASP, for each of the algebras. Hence, one can use Σ to generate the next solution J_i in polynomial time, and then, given J_i, use Σ'_i to generate all the interval realizations corresponding to J_i in polynomial time per realization. The composition of Σ and the Σ'_i-s will satisfy the requirements of the theorem. For convenience, we shall discuss here generating only those realizations in which the intervals have positive length.

For $\mathcal{A}_{13}$, to each solution corresponds a unique endpoint sequence, since every atomic relation uniquely determines the equalities and inequalities among the endpoints of the two intervals involved. Hence, ASP and ARP are equivalent for $\mathcal{A}_{13}$. The same is true for $\mathcal{A}_6$, since for problems in $\mathcal{A}_6$ all endpoints are required to be distinct. For $\mathcal{A}_3$, a solution is just an instance in the restricted domain $\{\prec, \cap, \succ\}$, and therefore the Π structure of [4] provides a polynomial representation structure. (In [4] all endpoints are assumed to be distinct, but the same structure can be modified to handle equalities as well).

It remains to prove the assertion for $\mathcal{A}_7$. Analogously to the proofs in section 5.1, we describe each atomic relation in $\mathcal{A}_7$ by one or more inequalities on the endpoints:

$$i \prec j \;\Leftrightarrow\; r_i < l_j \tag{1}$$

$$i \alpha j \;\Leftrightarrow (\; l_i < l_j \text{ and } l_j \leq r_i \;\text{ and } r_i < r_j\;) \tag{2}$$

$$i \equiv j \;\Leftrightarrow\; (\; l_j = l_i \text{ and } r_j = r_i\;) \tag{3}$$

$$i \subset j \;\Leftrightarrow\; (\; l_j \leq l_i \text{ and } r_i \leq r_j \text{ but not both } l_i = l_j \text{ and } r_i = r_j\;) \tag{4}$$

The converse relations are defined analogously. Also, for every event i,

$$l_i < r_i. \tag{5}$$

By identifying every pair of intervals i, j satisfying $i \equiv j$, we can get an instance in the restricted domain $\{\prec, \alpha, \subset, \succ, \alpha^{-1}, \subset^{-1}\}$. Similarly to the construction in Theorem 5.6, form a directed graph with vertices corresponding to interval endpoints, and an arc labelled $<$ (resp., $\leq$) from x to y if the relation between the two endpoints implied by the input using (1)-(5) is $x < y$ (resp., $x \leq y$). Shrink every cycle of arcs labelled $\leq$, identifying the corresponding endpoints. An argument analogous to that in Lemma 5.5 shows that the resulting graph G' is acyclic, and that to every interval realization of the input corresponds a linear extension of G'. Also, to every linear extension corresponds an interval realization consistent with the input, provided that the linear extension contains no new equivalent intervals, violating condition (4). Hence, one can generate all linear extensions in the manner indicated in Section 5.1.3. To avoid realizations containing new equivalent intervals, one can keep pointers in both directions between arcs corresponding to $l_j \leq l_i$ and to $r_i \leq r_j$ (generated by condition (4)), and use these pointers to make sure during the enumeration that whenever one weak inequality is set to equality, the other equality is disallowed. ■

Remark 5.18 A problem related to ASP and ARP is that of determining the *number* of realizations (or solutions) for a given input. Belfer and Golumbic have given a closed form formula to compute the number of distinct-endpoint realizations for inputs in the restricted domain $\{\prec, \succ, \cap\}$, (interval orders) [4]. For the restricted domain $\{\prec\!\!\succ, \cap\}$, (interval graphs,) they describe a polynomial algorithm to compute the number of distinct-endpoint realizations. On the other hand, the consistent instances for the restricted domain problem on $\Delta = \{\prec, \succ, \prec\!\!\succ\}$ are precisely the partial orders, and Brightwell and Winkler have recently shown that computing the number of linear extensions of a partial order is #P-complete [10]. This implies that computing the number of solutions (and the number of realizations) is #P-complete for $\Delta = \{\prec, \succ, \prec\!\!\succ\}$, and thus also for $\mathcal{A}_3$ and the larger algebras.

6 Conclusion

In this paper we have dealt with the consistency of assertions about the relations of intervals. We have investigated four basic problems in temporal reasoning: determining satisfiability of assertion on time intervals (ISAT), maximum strengthening of a satisfiable assertion (MLP), and producing all the consistent instantiations (ASP) or all interval realizations (ARP) via a polynomial representation structure. The three latter problems were shown to be tractable whenever ISAT is.

We have shown that even a major simplification of Allen's interval algebra - from thirteen atomic relations to three only - leaves the satisfiability problem intractable. On the positive side, we have shown that in this simplified algebra, $\mathcal{A}_3$, many restricted domain problems, (i. e., problems for which the input is restricted to a prespecified subset of the relation sets) are efficiently solvable. Polynomiality was shown in some cases by pointing out the equivalence to well known polynomial graph theoretic problems, and in others by techniques developed here. Table 3 summarizes the current complexity status on all restricted domains in $\mathcal{A}_3$. In the table, $\prec$ represents both $\prec$ and $\succ$, and $\prec\cap$ represents both $\prec\cap$ and $\cap\succ$, (Note that although there are seven non-empty relation sets in $\mathcal{A}_3$, we need to consider only the power set of the five relations $\prec, \cap, \prec\cap, \prec\!\!\succ, \prec\cap\succ$, since if a relation appears in the input, its converse also appears implicitly. Hence, $\succ$ appears in the input whenever $\prec$ does, and $\cap\succ$ appears whenever $\prec\cap$ does.) Each restricted domain is the union of its coordinates in the table. The top line for each entry in the table indicates the complexity of that domain, and the bottom indicates the source of the result. 'Poly' denotes a polynomially solvable problem, and 'NPC' an NP-complete one. 'T' denotes a case which is trivially satisfied by setting all intervals disjoint or identical. Parentheses indicate that this result is a corollary of a stronger result.

		$\cap$	$\prec\!\succ$	$\prec\!\succ, \cap$
	–	T	T	Poly interval graph
$\prec$	Poly linear order	Poly interval order	Poly acyclic graph	Poly Δ_2
$\prec\cap$	T	T	open	open
$\prec\cap\succ$	T	T	T	NPC Δ_0, interval sandwich
$\prec, \prec\cap$	Poly (Δ_1)	Poly (Δ_1)	open	open
$\prec, \prec\cap\succ$	Poly (Δ_1)	Poly (Δ_1)	Poly Δ_3	NPC (Δ_0)
$\prec\cap, \prec\cap\succ$	T	T	NPC Δ_4	NPC (Δ_0)
$\prec, \prec\cap, \prec\cap\succ$	Poly (Δ_1)	Poly Δ_1	NPC (Δ_4)	NPC (Δ_0)

Table 3: Complexity of ISAT on all restricted subdomains of A_3.

We conjecture that ISAT is NP-complete on $\Delta_5 = \{\prec\cap, \cap\succ, \prec\!\succ\}$. A proof to this conjecture will resolve the remaining four cases. Note that the large number of polynomial restrictions can be used to speed up the solution of a non-polynomial one, by reducing the required enumeration size: For a given instance, one can easily find which polynomial restriction is "closest" to it, in the sense that a minimum number of relation sets in the instance fall outside the restriction. By fixing the possible relations in those (hopefully few) relation sets, the resulting problem can be solved polynomially.

We have used the framework and the terminology of Allen's interval calculus for temporal reasoning, but the results apply in any context where the consistency of assertions about relations of intervals must be verified. The tools we have used were mainly from graph theory and combinatorics. We have hoped to demonstrate that the interconnection between these two disciplines is quite rich, and its investigation may benefit both fields. Many open questions arise as a result of this joint viewpoint: Further classification of the restricted domains in each of the the algebras $\mathcal{A}_i, i = 3, 6, 7, 13$ in terms of complexity, obtaining more efficient algorithms for MLP, ASP and ARP on specific domains, and many more.

Acknowledgment

Helpful conversations with Fred Roberts and Peter Ladkin are gratefully acknowledged. The initial stage of this work was done while the first author was a visitor at the IBM T.J. Watson Research Center, Yorktown Heights, NY, and the second author was a visitor at DIMACS and RUTCOR, Rutgers University, NJ. The work of the second author was supported in part by AFOSR grants 89-0512 and 90-0008, and by NSF grant STC88-09648.

References

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

[2] J. F. Allen. "Maintaining knowledge about temporal intervals." *Comm. ACM* 26 (1983) 832–843.

[3] K. R. Baker, P. C. Fishburn and F. S. Roberts. "Partial orders of dimension 2." *Networks* 2 (1972) 11–28.

[4] A. Belfer and M. C. Golumbic. "Counting endpoint sequences for interval orders and interval graphs." *Discrete Math.* (to appear).

[5] A. Belfer and M. C. Golumbic. " A combinatorial approach to temporal reasoning." *Proc. Fifth Jerusalem Conf. on Information Technology*, IEEE Computer Society Press, October 1990, 774–780.

[6] A. Belfer and M. C. Golumbic. " The role of combinatorial structures in temporal reasoning." IBM Research Report, (in preparation).

[7] S. Benzer. "On the topology of the genetic fine structure." *Proc. Nat. Acad. Sci. USA* 45 (1959) 1607–1620.

[8] A. Bouchet. "Reducing prime graphs and recognizing circle graphs." *Combinatorica* 7 (1987) 243–254.

[9] K. S. Booth and G. S. Lueker. "Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms." *J. Comput. Sys. Sci.* 13 (1976) 335–379.

[10] G. Brightwell and P. Winkler. "Counting linear extensions is #P-complete". *Proc. 23rd Annual ACM Symp. on Theory of Computing* (1991) 175–181.

[11] A. V. Carrano. "Establishing the order of human chromosome-specific DNA fragments", in *Biotechnology and the Human Genome*, A. D. Woodhead and B. J. Barnhart, eds. Plenum Press, NY (1988) 37–50.

[12] C. H. Coombs and J. E. K. Smith. "On the detection of structures in attitudes and developmental processes." *Psych. Rev.* 80 (1973) 337–351.

[13] B. Dushnik and E. W. Miller. "Partially ordered sets." *Amer. J. Math.* 63 (1941) 600–610.

[14] P. Fishburn. "Intransitive indifference with unequal indifference intervals." *J. Math. Psych.* 7 (1970) 144-149.

[15] P. Fishburn. *Interval Orders and Interval Graphs.* Wiley, New York, 1985.

[16] P. Fishburn. "Interval graphs and interval orders." *Discrete Math.* 55 (1985) 135–149.

[17] D. R. Fulkerson and O. A. Gross, "Incidence matrices and interval graphs." *Pacific J. Math.* 15 (1965) 835–855.

[18] P. C. Gilmore and A. J. Hoffman. "A characterization of comparability graphs and of interval graphs." *Canad. J. Math.* 16 (1964) 539–548.

[19] C. P. Gabor, K. J. Supowit and W-L.Hsu. "Recognizing circle graphs in polynomial time." *J. ACM* 36 (1989) 435–473.

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

[21] M. C. Golumbic. "Interval graphs and related topics." *Discrete Math.* 55 (1985) 113–121.

[22] M. C. Golumbic and E. R. Scheinerman. "Containment graphs, posets and related classes of graphs." *Ann. N.Y. Acad. Sci.* 555 (1989) 192–204.

[23] G. Hajös. "Über eine Art von Graphen." *Intern. Math. Nachr.* 11 (1957) problem 65.

[24] D. S. Hochbaum and J. Naor. "Simple and fast algorithms for linear and integer programs with two variables per inequality." *Proc. IPCO 92, Integer Programming and Combinatorial Optimization*, Carnegie Mellon University Press, (1992) (to appear).

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

[26] D. G. Kendall. "Some problems and methods in statistical archaeology." *World Archaeol.* 1 (1969) 68–76.

[27] N. Korte and R. H. Möhring. "Transitive orientation of graphs with side constraints." *Proc. WG' 85, Int. Workshop on Graph Theoretic Concepts in Computer Science*, H. Noltemeier, ed. Universitätsverlag Rudolf Trauner, Linz (1985) 143–160.

[28] N. Korte and R. H. Möhring. "An incremental linear time algorithm for recognizing interval graphs." *SIAM J. Comput.* 18 (1989) 68–81.

[29] P. B. Ladkin and R. Maddux. "The algebra of constraint satisfaction problems and temporal reasoning." Technical report, Kestrel Institute, Palo Alto, 1988.

[30] C. G. Lekkerkerker and J. Ch. Boland. "Representation of a finite graph by a set of interval on the real line." *Fundam. Math.* 51 (1962) 45–64.

[31] K. Nökel. *Temporally Distributed Symptoms in Technical Diagnosis.* Lecture Notes in Artificial Intelligence 517, Springer Verlag, 1991.

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

[33] C. Papadimitriou and M. Yannakakis. "Scheduling interval ordered tasks." *SIAM J. Comput.* 8 (1979) 405–409.

[34] F. S. Roberts. *Discrete Mathematical Models, with Applications to Social Biological and Environmental Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[35] T. J. Schaefer. "The complexity of satisfiability problems." *Proc. 10th Annual ACM Symp. on Theory of Computing* (1978) 216–226.

[36] D. Skrien. "Chronological ordering of interval graphs." *Discrete Appl. Math.* 8 (1984) 69–83.

[37] A. Tarski. "On the calculus of relations." *Journal of Symbolic Logic* 6 (1941) 73–89.

[38] R. E. Tarjan. "Depth-first search and linear graph algorithms." *SIAM Journal on Computing* 1 (1972) 146–160.

[39] P. van Beek. "Approximation algorithms for temporal reasoning." *Proc. Eleventh Int'l. Joint Conf. on Artificial Intelligence (IJCAI-89)*, August 1989, 1291–1296.

[40] P. van Beek. "Reasoning about qualitative temporal information." *Proc. Eighth Nat'l. Conf. on Artificial Intelligence (AAAI-90)*, August 1990, 728–734.

[41] P. van Beek. "Temporal query processing with indefinite information." *Artificial Intelligence in Medicine* 3 (1991) 329–339.

[42] M. Vilian and H. Kautz. "Constraint propagation algorithms for temporal reasoning." *Proc. Fifth Nat'l. Conf. on Artificial Intelligence (AAAI-86)*, August 1986, 337–382.

[43] S. A. Ward and R. H. Halstead. *Computation Structures.* MIT Press, 1990.