Reasoning with Multi-Point Events

R. Wetprasit and A. Sattar
School of Computing and Information Technology
Griffith University, NATHAN
Brisbane, 4111
AUSTRALIA
{rattana,sattar}@cit.gu.edu.au

L. Al-Khatib
Computer Science Program
Florida Institute of Technology
150 W. University Blvd.
Melbourne, Fl. 32901, USA
lina@cs.fit.edu

Abstract

Recent research on qualitative reasoning has focussed on representing and reasoning about events that occur repeatedly. Allen's *interval algebra* has been modified to model events that are collections of *convex* intervals—*a non-convex interval*. Using the modified version of Allen's algebra, constraint-based algorithms have been investigated for finding feasible relations in a network of non-convex intervals.

In this paper, we propose to model reoccuring events as multi-point events by extending Vilain and Kautz's *point algebra*. We then propose an exact algorithm (based on van Beek's exact algorithm) for finding feasible relations for multi-point event networks. The complexity of our method is compared with previously known results both for reoccuring and non-reoccuring events. We identify the special cases for which our multi-point based algorithm can find *exact solution*. Finally, we summarise our paper with brief discussion on ongoing and future research.

Keywords: Temporal Reasoning, Knowledge Representation, Constraint Satisfaction.

1 Introduction

Representing and reasoning with temporal knowledge have been considered as an essential component of intelligent systems, especially in qualitative reasoning. It has also been an active area of research in philosophy, psychology, linguistics and computer science. Time plays a crucial role in many areas of reasoning such as program verification, scheduling, planning and natural language understanding. Since the provocative paper on *The Naive Physics manifesto* by Pat Hayes [8], a great deal of research has been done in the area of qualitative/temporal reasoning [2, 3, 4, 5, 9, 15, 1, 7, 11, 14].

Two well understood approaches for representing temporal information (as qualitative relations between interval and time points) are *interval algebra* by Allen [2] and *point algebra* by Vilain and Kautz [15]. Motivated by computational advantages, Peter van Beek proposed a restricted algebra (*pointizable interval algebra*) [14] that can be translated into equivalent set of relations in point algebra. However, reasoning with temporal information involving events that occur repeatedly, such as **Two times of bus arrivals are earlier than specified in the time table**, has not been extensively investigated in the literature. Khatib [1] and Ladkin [10] have recently proposed interval-based approaches to model reoccuring events by extending Allen's interval algebra.

In this paper, we propose to model reoccuring events as multi-point events by extending Vilain and Kautz's *point algebra*. We then propose an exact algorithm (based on van Beek's exact algorithm) for finding feasible relations for multi-point event networks. The complexity of our method is compared with previously known results both for reoccuring and non-reoccuring events. We identify the special cases for which our multi-point based algorithm performs better than the non-convex interval based algorithms.

The rest of the paper is organised in a number of sections. The next section gives an overview of the previous research on temporal reasoning including interval algebra (IA), point algebra (PA), and pointizable interval algebra (SIA). Section 3 introduces the non-convex interval model. Section 4 describes our proposed framework for *multi-point events* (MPEs). Section 5 presents an algorithm to find possible feasible relations between multi-point events. Section 6 describes an example to demonstrate the functioning of our algorithm. Finally, we conclude our paper with comments on ongoing and future work.

2 Basic Approaches to Temporal Reasoning

Two basic approaches to representing temporal information are: *interval algebra* (IA) [2] and *point algebra* (PA) [15]. The interval algebra (IA) consists of a set, say I, of 13 atomic convex relations (ACRs) between two intervals: $I = \{<, >, =, m, \tilde{m}, o, \tilde{o}, d, \tilde{d}, s, \tilde{s}, f, \tilde{f}\}$. Precisely, this algebra is defined over elements of the powerset of I which is of size $2^{|I|}$, an unary operator *inverse* ($\tilde{}$), and binary operators: *intersection* ($\times$) and *composition* ($\circ$). Operands of the operators are sets of basic relations. The intersection of two relations is simply a set theoretic operation. The inverse and composition operations distribute over set union. Therefore, the inverse of a relation is the union of the inverse of each basic relation. The composition is an operation over two relations between two pairs of intervals with one common interval, i.e., $xR \circ Sz \leftrightarrow \exists y \; xRy \wedge \; ySz$.

The second approach considers events as instances of time, called *point algebra* (PA) as proposed by Vilain and Kautz [15]. It involves a mapping from time points to some real numbers on an imaginary time line. The basic elements of PA are three atomic point relations (APRs) that hold between two points: $P = \{<, >, =\}$. The PA consists of the powerset of P which is of size $2^{|P|}$, and the same operators as in IA.

These algebras are expressively rich enough to represent ambiguous and/or qualitative information about relationship between two intervals (points) in terms of disjunction of possible atomic relations. For example, **Sam attended a lecture and left after the lecture was finished** tells us that the interval of the lecture ended either before or at the same time as Sam left. Though we don't know explicitly the relation between two time intervals: lecture and Sam's presence, we could represent possible relations between these two intervals as **the lecture** $\{=, d, o, s, f, \tilde{f}\}$ **Sam is in the theatre**, whereas a comma denotes disjunction 'or'.

Interestingly, the problem of qualitative/temporal reasoning could be formulated as the *Constraint Satisfaction Problem* (CSP) by transforming the interval or point algebra specification of a problem domain into a *constraint network* [2, 15, 14, 12, 11, 1]. For IA, each node in the CSP network represents a variable which is an interval. The domain of a variable is an ordered pair of real numbers over time line, or $D_i = \{< a, b >: a < b\}$. The label on an arc connecting two nodes is an element from the set of possible relations between two intervals. Similarly, in case of PA, each node represents a variable (time point) and domain of variable is *one* real value on time line. We can then apply well known CSP techniques such

as *path-consistency* algorithms [2, 15, 12, 1] to construct feasible relations both for IA and PA networks. The problem of finding feasible relations is NP-complete for IA [15, 14, 7]. In [7], Golumbic and Shamir have proved more stronger results for the full IA and subsets of IA which include at least the relations **before or after**. They have shown that the problem of finding feasible relations (also called as *minimal labelling problem* (MLP)), *interval satisfiability problem* (ISAT), *all solutions problem* (ASP), and *all realisations problem* (ARP) are NP-complete. However, feasible relations can be computed in polynomial time for PA [15, 14]. Though computationally attractive PA is intuitively not appealing as much as IA in terms of representing real world events. It is evident from the fact that IA cannot be completely translated into PA [15, 14]. The subset of IA which can be translated into PA is called *pointizable* IA (SIA) [14]. For example if L denotes the interval of **the lecture**, and S denotes the interval **Sam is in the class**, the relation between **the lecture** and the interval that **Sam is in the class** can be encoded into PA relation as follow

$$((L^- < S^+) \wedge (L^+ > S^-) \wedge (L^+ \leq S^+))$$

where L^- and L^+ represent the starting and ending points of interval L, respectively. There is no constraint between the starting points of both intervals. If we don't have any information about relations between two events (either intervals or points), then the relationship between such two events could be any of all possible relations, called *Universal relation* (U), i.e., the set I for IA, or the set P for PA. The subset of IA in which cannot be translated into PA is called *disjointedness relations* and thoroughly investigated in [6].

In [14], van Beek investigated a number of constraint network based algorithms for finding feasible relations and a consistent scenario including the exact algorithm for minimal labelling problem of PA. The exact algorithm is based on the fact that there is an infeasible relation which occurs in the 4-vertices subgraph of path-consistent PA network, called a *forbidden subgraph* (see Figure 1), when we allow the relation '$\neq$' in our language.

The complexity of algorithm [14] is the maximum of the complexities of path consistency $O(n^3)$, and the algorithm to find and remove the infeasible relations, $O(mn^2)$, i.e., $O(max(n^3, mn^2))$. However, in the worst case m, the number of '$\neq$' edges, is $O(n^2)$. Thus, the worst case complexity of the exact algorithm is $O(n^4)$.

3 Non-Convex Interval Model

In Allen's interval model, an interval represents a consecutive event. However, there are some classes of events that are collections of many continuous intervals, called *non-convex intervals* (NCIs) [1]. This allows us to reason about single events that are interrupted and later resumed (gaped events), events that occur repeatedly (recurring events), and events that consist of many related subevents (unions of events).

For computational purpose, this model captures the internal relations between subintervals of two NCIs by using a matrix relation of size $n \times m$, when n and m are the numbers of subintervals in those two NCIs. Each entry of matrix relations is an element of CR, since the model also allows the qualitative information between pairs of subintervals.

A node in NCI network represents a NCI, when the domain of each variable is the set of sequences of ordered pairs of real numbers each representing a convex subinterval, or $D_i =$

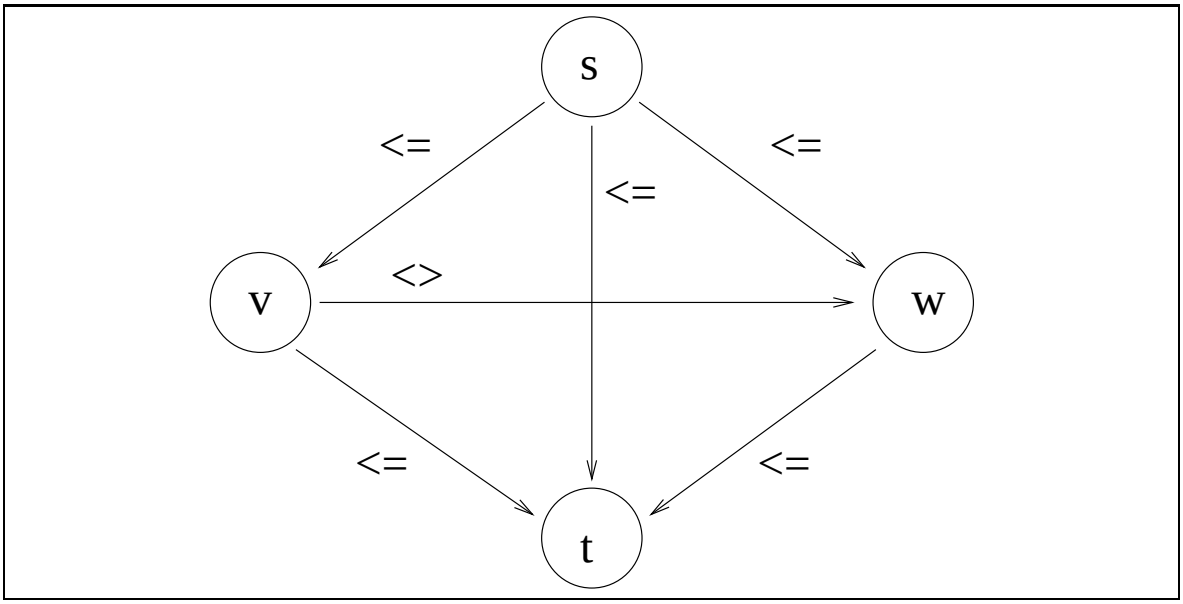


Figure 1: A forbidden subgraph

$\{< a_1, b_1 >, < a_2, b_2 >, ..., < a_m, b_m >: a_j < b_j < a_{j+1}\}$, where there are m subintervals in the NCI. A matrix relation infers a constraint or label between the two consecutive nodes.

The 3-consistency checking is done in two levels: between two NCIs and three NCIs. The *canonical form* is defined to ensure 3-consistency between three subintervals being in two different NCIs. A matrix relation is in the *canonical form* if its entries satisfy the path consistency conditions among their neighbours (more detail of the algorithm for converting a matrix into the canonical form is in [1]).

The 3-consistency between three NCIs obtains from the composition operations over two matrix relations.

The constraint satisfaction algorithm propagates the path consistency checking, while the complexity relies on the number of NCIs and number of subintervals in each NCI, in which we assume that there are at most n subintervals in each NCI. The complexity of the *basic* algorithm for constraint propagation for NCI networks is $O(n^5k^3)$.

Eliminating the infeasible relations of NCI network to satisfy path-consistency can be achieved in two ways. Firstly, propagating the consistency checking on NCI network as proposed in [1], secondly, constructing the corresponding IA network where the number of nodes is the sum of subintervals in all NCIs, then applying the algorithm for IA networks. Performing on NCI networks is empirically shown more powerful than on IA networks in terms of time (number of composition operations), and space (memory units) requirement.

Moreover, in [1] Khatib has pointed out that there is only a small region of a matrix relation, called *mixed region*, which is affected when the relations between subintervals are updated. Concentrating only on the mixed region would eliminate redundant computation on matrix operations. Consequently, it reduces the complexity of the algorithm to $O(n^3k^3)$.

4 Multi-Point Events and Their Relations

4.1 Basic Terminology

A *multi-point event* is a collection of points, when each point represents the related subevents. A multi-point event is in *normal form* if all pairs of points in a multi-point event have the relation either 'before' or 'after' (referenced as '<' and '>' in [15]). The *size* of a multi-point event is the number of points in that multi-point event.

For a normal multi-point event P with n points, we define the following:

- *SIZE(P)* is the function returns the size of P which is n.
- A multi-point event (MPE) of size n is called n-point event.
- $P_1 < P_2 < ... < P_i < P_{i+1} < ... < P_n$, when P_i refers to the ith point of P, $1 \leq i \leq n$.
- $R(I, J)$ is the *multi-point relation* (MPR) between I and J. An element $R(I_i, J_j)$ is a point relation between the ith point of the multi-point event I and the jth point of the multi-point event J.

4.2 Binary Relations

The *binary relations* between MPEs can be classified into two categories: *external* and *internal* relations.

Definition 1 (An external relation) *An external relation between two MPEs, I and J, are the Allen's interval relation between the smallest intervals containing MPE I and J. Suppose $I = \{I_1 < ... < I_n\}$ and $J = \{J_1 < ... < J_m\}$ are two MPEs. An external relation between I and J is defined by the relations between I_1, J_1, I_n and J_m. The external relations are crucial when we limit our consideration only on the boundary of those two MPEs.*

Definition 2 (An internal relation) *An internal relation between two MPEs I and J is a binary point relation between a point in MPE I and a point in MPE J or vise versa. The internal relations between I and J are the inverse of the internal relations between J and I, therefore, knowing one set of them is sufficient to determine the other.*

4.3 Matrix Relation Representation

Internal relations between two MPEs, I of size n and J of size m, can be represented as a $n \times m$ matrix of point relations (PR): $(\emptyset, <, \leq, =, >, \geq, \neq, ?)$. ? means there is no constraint between two points and can be either '<', '=', or '>', while '$\neq$' is '<', or '>'. The rows of matrix represent the points of I and the columns represent the points of J. The matrix element in row (i) and column (j) is an element of PR.

Example: Consider the explicit information about three times of Tom's arrival at home. **The first time he arrived before raining, the second time the rain started at the same time he arrived and then the phone rang, the third time he arrived after the rain had started but the same time as the call.**

There are three MPEs here: the events that Tom arrived, the rain started, and the phone rang. And if we symbolise the three MPEs as T, R, and P, respectively, this information can be depicted in Figure 2:

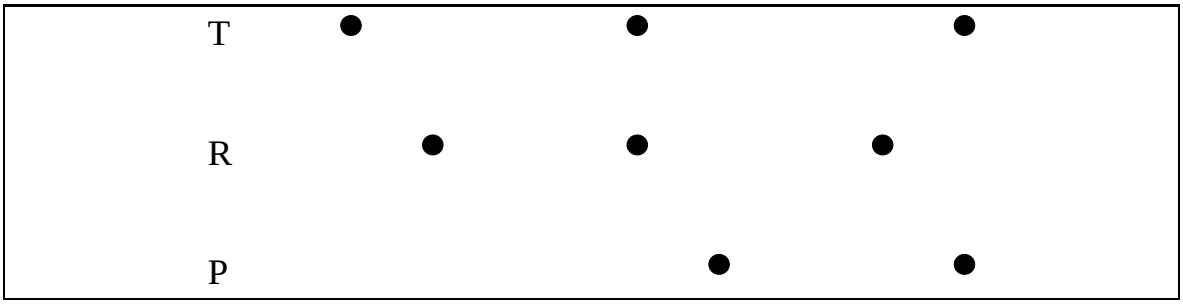


Figure 2: Pictorial Representation

The matrix relations representing the internal relations between T and R, and R and P are following:

$$R(T,R) = \begin{vmatrix} < & < & < \\ > & = & < \\ > & > & > \end{vmatrix}$$

$$R(R,P) = \begin{vmatrix} < & < \\ < & < \\ > & < \end{vmatrix}$$

4.4 MPE Networks

A binary constraint network of k MPEs is defined and has characteristics as follows:

- $N = \{N_1, N_2, ..., N_k\}$ is the set of nodes where each node represents an individual MPE.
- $D = \{D_1, D_2, ..., D_k\}$ is the set of domains for the nodes. Since nodes are MPEs, their domain can be defined to be the set of real numbers each representing a point subevent, or
$$D_i = \{a_1, a_2, .., a_n : a_j < a_{j+1}\}$$
where SIZE(N_i) is n.
- $A = \{C_{1,2}, C_{1,3}, ..., C_{1,k}, C_{2,1}, C_{2,3}, ..., C_{2,k}, C_{k,1}, C_{k,2}, ..., C_{k,k-1}\}$ is the set of labels on the arcs between the k nodes. $C_{i,j}$ is the label on the arc from N_i to N_j. The value of $C_{i,j}$ is a matrix relation of size SIZE(N_i) $\times$ SIZE(N_j).
- If the arc from N_i to N_j appears in the network with label $C_{i,j}$, then the arc from N_j to N_i doesn't have to be explicitly represented. It could be defined as $\breve{C}_{i,j}$.
- If no arc appears between N_i and N_j, then $C_{i,j}$ is assumed to be the matrix of *universal point relations.*
- An **instantiation** of a node is an k-tuple $(x_1, x_2, ..., x_k)$ when $x_i \in D_i$.
- A **consistent instantiation** of a network is an instantiation such that all the relations between nodes are satisfied.
- A network is said to be **overconstrained** if no consistent instantiation exists.

- A **scenario** is a set of atomic relations between pairs of MPEs. Each atomic relation corresponds to a matrix label for each arc.
- A **consistent scenario** is a scenario that has a consistent instantiation.
- The matrix of feasible relations between two MPEs N_i and N_j is the matrix in which each element is consisting of all and only the feasible relations. Also, the matrix of feasible relations is called the **minimal label**.
- A MPE network can be transformed into an equivalent PA network that is called a **corresponding PA network**

4.5 Canonical Form

Similarly to [1], we also define the canonical form to ensure that a MPR between any two MPEs always satisfies the path consistency conditions [13].

A matrix relation $A_{n\times m}$ is said to be in the canonical form if the following conditions are satisfied [1]:

1. $A_{i,j-1} \subseteq A_{i,j} \circ >$ $(\forall 1 \leq i \leq n, 1 < j \leq m)$
2. $A_{i+1,j} \subseteq > \circ A_{i,j}$ $(\forall 1 \leq i < n, 1 \leq j \leq m)$
3. $A_{i,j+1} \subseteq A_{i,j} \circ <$ $(\forall 1 \leq i \leq n, 1 \leq j < m)$
4. $A_{i-1,j} \subseteq < \circ A_{i,j}$ $(\forall 1 < i \leq n, 1 \leq j \leq m)$

These conditions could be used to approximate the path-consistency of the MPE network. The following theorem considers a trivial case of MPE network which has only two MPEs. Before introducing that theorem, we give a related lemma as follows:

Lemma 3 *Given a MPE network with k MPEs ($k \geq 1$), each contains n points ($n \geq 2$). The MPE network has a corresponding PA network.*

Proof: The corresponding PA network of the MPE network is a PA network with $k \times n$ nodes, each node represents each time point subevent in a MPE. The labels on arcs of the PA network capture all internal MPRs (when $k > 1$) and the relations '$<$' between node i and $i+1$ of a common MPE ($1 \leq i \leq n$). ■

Theorem 4 *Suppose a MPE network has two nodes I and J. Let A be the matrix relation between I and J. If A is in canonical form, then the corresponding PA network is path-consistent.*

Proof: The proof is simple, and could be constructed in the similar manner as the proof of Theorem 4.6 in [1]. ■

4.6 Matrix Relations Operations

The operations on two matrix relations, which are necessary in solving reasoning tasks, are defined by adopting the operations on constraint matrices from [11]. Table 1 defines the result of matrix operations ($\oplus$, $\otimes$, $\hat{}$, $\odot$, and $\breve{}$) on two MPEs in terms of basic operations in point algebra. The symbols $+$, $\times$, $^{-}$, $\circ$, and $\breve{}$ correspond to union, intersection, complement, composition and inverse operators defined in PA, respectively. Of course, matrix operations are subject to certain conditions, e.g., *union* operation could only be performed on two matrices of the same size.

Union	$C = A \oplus B$ iff $\forall i,j\ (1 \le i \le n, 1 \le j \le m)\ C_{i,j} = A_{i,j} + B_{i,j}$
Intersection	$C = A \otimes B$ iff $\forall i,j\ (1 \le i \le n, 1 \le j \le m)\ C_{i,j} = A_{i,j} \times B_{i,j}$
Complement	$C = \hat{A}$ iff $\forall i,j\ (1 \le i \le n, 1 \le j \le m)\ C_{i,j} = {}^{-}A_{i,j}$
Composition	$C = A \odot B$ iff $\forall i,j\ (1 \le i \le n, 1 \le j \le m)\ C_{i,j} = \prod_{k=1}^{r}(A_{i,k} \circ B_{k,j})$ $= A_{i,1} \circ B_{1,j} \times A_{i,2} \circ B_{2,j} \times ... \times A_{i,r} \circ B_{r,j}$
Inverse	$C = \breve{A}$ iff $\forall i,j\ (1 \le i \le n, 1 \le j \le m)\ C_{i,j} = {}^{\breve{}}A_{j,i}$

Table 1: The Multi-Point Event Operations

5 Reasoning with MPE Networks

In this section, we propose an algorithm to find all feasible relations for MPE network based on van Beek's exact algorithm for PA network [14]. He has shown that the path-consistency algorithm alone is not sufficient for PA network by raising a counter-example consisting of four vertices called a *forbidden subgraph*, and then proposed a polynomial time algorithm to correctly find the minimal labels for a PA network.

We define the forbidden subgraph for MPE network in terms of points in MPEs. Then we present an algorithm using the canonical form matrices and the matrix operations to solve the minimal label problem in polynomial time. We prove that we do not need to apply the path-consistency algorithm again after performing the algorithm since the network is still path-consistent. We analyse the complexity of the algorithm and finally we give an example to compare the performance between the two approaches: manipulating on MPE network and the corresponding PA network.

Definition 5 *Given a MPE subgraph of any four nodes: V, W, S, and T. Let V be ($V_1 < ... < V_m$), W be ($W_1 < ... < W_n$), S be ($S_1 < ... < S_o$), and T be ($T_1 < ... < T_p$), where m, n, o and p are the numbers of points in V, W, S and T respectively. The subgraph is called a* forbidden *MPE subgraph, if the following conditions are satisfied:*

$$\begin{aligned} R(V_v, W_w) &= \text{`}\neq\text{'} \\ R(V_v, S_s) &= \text{`}\geq\text{'} \\ R(W_w, S_s) &= \text{`}\geq\text{'} \\ R(V_v, T_t) &= \text{`}\leq\text{'} \\ R(W_w, T_t) &= \text{`}\leq\text{'} \end{aligned}$$

The infeasible relation in the forbidden MPE subgraph is $R(S_s, T_t)$, which is '$\leq$'. This relation causes inconsistency among those four nodes. However, if we don't allow '=' between S_s and T_t, the subgraph becomes 4-consistent.

If S is identical to T, the relation between S_s and T_t must be '$<$' because all MPEs are in normal form (either '$<$' or '$>$' allowed between points in MPE). Thus, we don't need to take this case into consideration. Similarly, we can show that other pairs of MPE nodes in the forbidden subgraph cannot be identical. Therefore, the set of four MPE nodes used in a forbidden subgraph are unique.

Algorithm 6 (FEASIBLE_MPE)
Input: *A* MPE *network represented by matrix relations R[I, J], where an entry of R[I, J] is an internal relation between points in MPEs I and J*
Output: *The set of feasible relations for R[I, J],* $I, J = 1, 2, .., k$

begin
    PATH_CONSISTENCY_MPE
    FIND_SUBGRAPHS_MPE
end

Procedure 7 (PATH_CONSISTENCY_MPE)
begin
    $Q := \{(I, K, J) \mid 1 \leq I < J \leq k,\ 1 \leq K \leq k,\ K \neq J\}$
    While Q is not empty do
        begin
            select and delete a path (I, K, J) *from Q*
            $Temp := R[I, J] \otimes (R[I, K] \odot R[K, J])$
            If $(Temp \neq R[I, J])$ *then*
                begin
                    Canonical_Conv(Temp)
                    $R[I, J] := Temp$
                    $R[J, I] := \breve{T}emp$ *(inverse of Temp)*
                    $Q := Q \cup RELATED\_PATHS(I, J)$
                end
        end
end

procedure RELATED_PATHS(I, J)
    Return $\{(I, J, K),\ (K, I, J) \mid 1 \leq K \leq k,\ K \neq I,\ K \neq J\}$

Procedure 8 (FIND_SUBGRAPHS_MPE)
begin
    For all matrix relations such that $R[V, W]_{v,w}$ = '$\neq$'
        $(1 \leq V < W \leq k)$ *and* $(1 \leq v < w \leq n)$ *do*
    begin
        For all MPE K $(1 \leq K \leq k,\ K \neq V, W)$ *do*
        begin

$P = P \cup adj\_MPEs(\geq, V_v, K) \times adj\_MPEs(\geq, W_w, K)$
$Q = Q \cup adj\_MPEs(\leq, V_v, K) \times adj\_MPEs(\leq, W_w, K)$
end
For each $S_s \in P$ *do*
For each $T_t \in Q$ *do*
begin
$R[S,T]_{s,t} := \text{`}<\text{'}$
$R[T,S]_{t,s} := \text{`}>\text{'}$
end
end
end

Our algorithm for computing feasible relations consists of main tasks: checking path-consistency, and finding forbidden MPE subgraphs and eliminating the infeasible relations from them. The path consistency algorithm determines the consistency between three MPEs throughout the network. Every time when there is an update, we maintain the consistency between two MPEs by transforming the changed matrix relations into canonical form before inserting to the database, by calling procedure Canonical_Conv (detailed in [1]). Function $adj\_MPEs(\geq, V_v, K)$ returns the set of elements K_k, in which $R(V_v, K_k) = \text{`}\geq\text{'}$ by checking all points in MPE K.

One thing we have to ensure here is that after re-labelling the forbidden subgraph to achieve 4-consistency, we do not loose the path-consistency of the MPE network.

Lemma 9 *Changing the label $R(S_s, T_t)$ of the forbidden subgraph defined in Definition 5 will not lead to path inconsistency.*

Proof: This can be proved by two path consistency levels: between two MPEs and more than two MPEs.

Between two MPEs:

For this case, it implies the question that we need to check whether the matrix $R(S,T)$ is in canonical form or not. What follows will show that we need not to do so.

Recall that MPEs we are considering are restricted to be in normal form. Given a new label on $R(S_s, T_t)$, if we were to apply the procedure to convert the matrix into canonical form, we need to consider its four neighbours which are $R(S_s, T_{t+1})$, $R(S_{s-1}, T_t)$, $R(S_s, T_{t-1})$, and $R(S_{s+1}, T_t)$. We will demonstrate only the right-hand neighbour: $R(S_s, T_{t+1})$. The rest cases can be proved by the same argument.

$R(S_s, T_{t+1})$: If we denote the relation $R(S,T)$ as the matrix A, the two conditions that would be examined are:

$$A_{s,t+1} \subseteq A_{s,t} \circ A_{t,t+1}$$

$$A_{t+1,t} \subseteq A_{t+1,s} \circ A_{s,t}$$

Consider the first condition. Before updating $A_{s,t}$, we have $A_{s,t} = \leq$, and $A_{t,t+1} = <$, therefore, $A_{s,t+1} = <$ or

$$\begin{array}{rcl} A_{s,t+1} & \subseteq & A_{s,t} \circ A_{t,t+1} \\ < & \subseteq & \leq \circ < . \end{array}$$

This shows that changing $A_{s,t}$ to '$<$' does not affect $A_{s,t+1}$.

Consider the second condition. Since we know that $A_{t+1,t} = >$ and $A_{s,t} = \leq$, thus $A_{t+1,s} = >$ or

$$\begin{array}{rcl} A_{t+1,t} & \subseteq & A_{t+1,s} \circ A_{s,t} \\ > & \subseteq & > \circ \leq \end{array}$$

Thus, updating $A_{s,t}$ to '$<$' also does not constraint $A_{t+1,t}$

Therefore, after performing procedure FIND_SUBGRAPHS_MPE, the matrix is still in canonical form.

More than two MPEs:

The proof of this case has been justified from [14], and is based on the argument that the four nodes which form a forbidden subgraph are in four different MPEs (V, W, S, T).

If we were to check the path consistency after changing the relation $R(S_s, T_t)$ to '$<$', the set of all possible triples that would be examined is given by procedure RELATED_PATHS: $\{(S, T, K), (K, S, T) \mid 1 \leq K \leq k, K \neq S, K \neq T\}$. Thus there are two cases:

Case 1 (S, T, K): The condition needed to be satisfy is:

$$R(S_s, K_k) \subseteq R(S_s, T_t) \circ R(T_t, K_k)$$

For any K, changing the relation of $R(S_s, T_t)$ from '$\leq$' to '$<$' would force $R(S_s, K_k)$ to be '$<$' only when $R(T_t, K_k)$ is either '$\leq$' or '$=$'. There are two possibilities here. If $S = K$, $R(S_s, K_k)$ is already '$<$' since MPEs are in normal form. If $S \neq K$, because we know that $V_v \leq T_t$, $W_w \leq T_t$, and $T_t \leq K_k$ (or $T_t = K_k$), this implies that $V_v \leq K_k$ and $W_w \leq K_k$. Thus, the four nodes: V_v, W_w, S_s, K_k form the forbidden subgraph and the relation $R(S_s, K_k)$ will be found and changed to '$<$' in the current call of procedure FIND_SUBGRAPHS_MPE.

Therefore, repeating procedure PATH_CONSISTENCY is not necessary.

Case 2 (K, S, T): This can be proved as case 1. ■

Theorem 10 *The algorithm FEASIBLE_MPE, for k MPE nodes in the constraint network and each node contains at most n points, has a time complexity of $O(max(n^5k^3, mn^2k^2))$, where m is the number of '$\neq$' internal relations in the network.*

Proof: For the procedure PATH_CONSISTENCY_MPE, initially, Q starts with $\frac{1}{2}k(k-1)(k-2)$ triples. Each arc of the network is labelled by a matrix relation of size $n \times n$, and it is this matrix relation that can be changed. As a result, there are at most $O(n^2k^3)$ triples could go into the Q. In each While loop, $O(n^2 + nn^2 + n^2)$ operations are needed for intersection, composition and converting matrix *Temp* to canonical form, respectively[1]. Therefore, the complexity of procedure PATH_CONSISTENCY_MPE is $O(n^5k^3)$.

Procedure FIND_SUBGRAPHS_MPE initially searches for the '$\neq$' edges. The number of all possible internal relations of MPE network needed to search for '$\neq$' relations are $O(n^2k^2)$.

For each '$\neq$' edge, we require to find the other two nodes that create a forbidden MPE subgraph, in which there are $2(k-2)$ matrix relations labelling on the edges connecting to that '$\neq$' edge. And for each matrix, there are n relations needed to examine (each adj_MPEs function performs over n elements of one row). Thus, the first inner loop requires $O(nk)$ run time. Then there are at most $n(k-2)$ nodes for S_s and $n(k-2)$ nodes for T_t resulted from the previous loop that their relations must be updated. So the run time of the second inner loop

[1]The complexity of procedure Canonical_Conv has been proved to be $O(n^2)$ in [1].

is $O(n^2k^2)$. Therefore the complexity for procedure FIND_SUBGRAPHS_MPE is $O(n^4, k^4)$, or $O(mn^2k^2)$ when m is the number of '$\neq$' edges.

Over all, the complexity of our algorithm is $O(max(n^5k^3, mn^2k^2))$ or $O(max(n^5k^3, n^4k^4))$. ■

Theorem 11 *The closure of a MPE network, calculated by the algorithm FEASIBLE_MPE, corresponds to a path consistent PA network,*

Proof: We can formulate the proof in the same way as the proof of Theorem 4.9 in [1]. ■

Theorem 12 *The algorithm FEASIBLE_MPE correctly finds the minimal labels for all internal relations in the MPE network.*

Proof: The proof could be constructed using insights from the proof of Theorem 4.5 in [14]. We will provide a complete proof in the full paper. ■

6 An Example: Constraint Propagation in a MPE network

Let us consider a MPE network consists of four MPEs: P of size 2, Q of size 2, R of size 3 and S of size 4. The initial internal relations are represented as follows:

$$R(P,Q) = \left| \begin{array}{cc} \neq & \leq \\ > & < \end{array} \right|$$

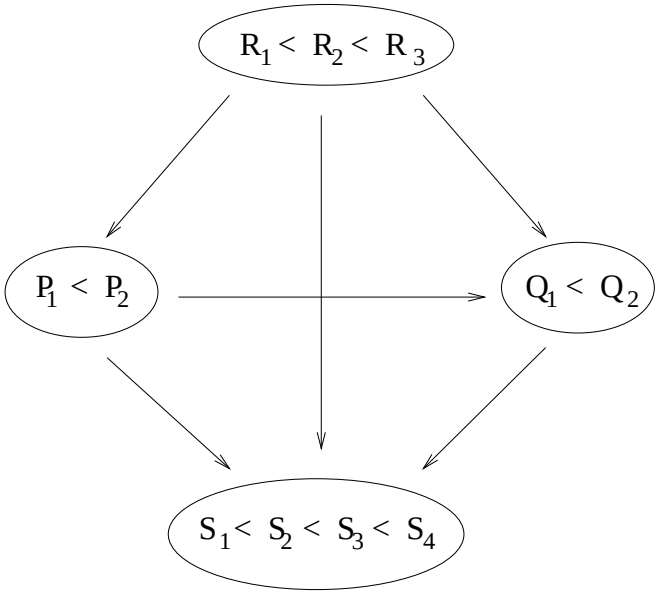
$$R(Q,S) = \left| \begin{array}{cccc} > & > & \leq & < \\ > & > & > & < \end{array} \right|$$

$$R(P,S) = \left| \begin{array}{cccc} > & > & \leq & < \\ > & > & > & \leq \end{array} \right|$$

$$R(R,Q) = \left| \begin{array}{cc} < & < \\ \leq & \neq \\ > & \neq \end{array} \right|$$

$$R(R,P) = \left| \begin{array}{cc} \leq & \leq \\ \leq & \leq \\ > & < \end{array} \right|$$

$$R(R,S) = \left| \begin{array}{cccc} > & \leq & < & < \\ > & > & \leq & < \\ > & > & > & < \end{array} \right|$$



This simple example is used to demonstrate the functioning of our algorithm FEASIBLE_MPE. We analyse the efficiency of the algorithm in terms of composition operations

performed on point relations and memory units to store all elements of the six matrix relations. We then compare the figures with the performance of the algorithm proposed in [14] for the corresponding PA network.

6.1 MPE Network

Step 1: Adding Step

Firstly, initialise all elements of the proper size matrices, representing MPR, to *universal relations*, and initialise Q to be empty. The next step is to call the procedure Canonical_Conv to convert each matrix relations into canonical form. Then we can directly add the matrix relations to the database.

After 6 calls to Canonical_Conv, we get the result as follow: (Only some relations in matrix $R(P,Q)$, $R(R,Q)$, $R(R,P)$ are changed.)

$$R(P,Q) = \begin{vmatrix} \neq & < \\ > & < \end{vmatrix}$$

$$R(R,Q) = \begin{vmatrix} < & < \\ \leq & < \\ > & \neq \end{vmatrix}$$

$$R(R,P) = \begin{vmatrix} < & < \\ \leq & < \\ > & < \end{vmatrix}$$

Step 2: Constraint Propagation Step

The next step is considering the constraint satisfaction by using the algorithm in Figure 6. We split this stage into 2 steps (Step 2.1 and Step 2.2).

Step 2.1 Calling to PATH_CONSISTENCY_MPE algorithm.

The queue starts with 12 triples: {(P,R,Q),(P,S,Q),(P,Q,R),(P,S,R), (P,Q,S),(P,R,S), (Q,P,R),(Q,S,R),(Q,P,S),(Q,R,S),(R,P,S),(R,Q,S)}. Follow the algorithm, only two matrix relations are updated: $R(P,S)$ and $R(R,Q)$.

$$R(P,S) = \begin{vmatrix} > & > & \leq & < \\ > & > & > & < \end{vmatrix}$$

$$R(R,Q) = \begin{vmatrix} < & < \\ \leq & < \\ > & < \end{vmatrix}$$

Step 2.2 Calling to FIND_SUBGRAPHS_MPE algorithm.

There is only one '$\neq$' relation in the matrix relation: $R(P_1,Q_1)$.

$$adj\_MPEs(\geq, P_1, K) = \{R_2\}$$

$$adj\_MPEs(\geq, Q_1, K) = \{R_2\}$$

$$adj\_MPEs(\leq, P_1, K) = \{S_3\}$$

$$adj\_MPEs(\leq, Q_1, K) = \{S_3\}$$

Then we change the relation $R(R_2, S_3)$ to '$<$'. Finally, all feasible relations between pairs of MPEs are identified:

$$R(P,Q) = \begin{vmatrix} \neq & < \\ > & < \end{vmatrix}$$

$$R(Q,S) = \begin{vmatrix} > & > & \leq & < \\ > & > & > & < \end{vmatrix}$$

$$R(P,S) = \begin{vmatrix} > & > & \leq & < \\ > & > & > & < \end{vmatrix}$$

$$R(R,Q) = \begin{vmatrix} < & < \\ \leq & < \\ > & < \end{vmatrix}$$

$$R(R,P) = \begin{vmatrix} < & < \\ \leq & < \\ > & < \end{vmatrix}$$

$$R(R,S) = \begin{vmatrix} > & \leq & < & < \\ > & > & < & < \\ > & > & > & < \end{vmatrix}$$

According to the Definition 4.6, performing a matrix composition of two matrix relations requires $n \times r \times m$ point compositions, when the dimensions of the first and second matrices are $n \times r$ and $r \times m$, respectively. Therefore, the number of composition operations performed on point relations for this example is 464 (114 composition for converting matrix relations into canonical form in step 1, 316 composition when calculating matrix composition in step 2.1, and another 34 composition to check whether the changed matrices are in canonical form step 2.1). The process required 44 memory units to store all elements of the six matrix relations.

6.2 PA Network

We construct the PA network corresponding to the MPE network for the example shown above. The corresponding PA network is composed of 11 nodes. The nodes in corresponding PA network represent points that are subevents of P, Q, R, and S. Each subevent is considered to be an independent point event. We then apply the algorithm FEASIBLE [14] to the PA network. We found that the resulting point relations in the corresponding PA network are the same as performing on MPE network. However, the number of point composition operations required is *greater than* 495. We specify *greater than* because we omit the operations demanded when there are some updates. And the memory units used to store point relations between all 11 nodes are 55 units.

The comparison between these two approaches indicates that MPE-based reasoner performs better than the traditional point-based approach. This might be the advantage of the omitted computation between pairs of points in the MPE.

7 Conclusion

The main contribution of this paper is an extension of point-based representation to reason with the recurring events that are considered as collections of point subevents. The algorithm we proposed correctly finds all feasible relations in the MPE network. The complexity of this algorithm is $O(max(n^5k^3, mn^2k^2))$, where k is the number of MPEs with maximum n points and m is the number of '$\neq$' internal relations in the network. The complexity for finding the same solutions for non-reoccuring PA network with the same data (nk point events) is $O(max(n^3k^3, mn^2k^2))$, where m is also the number of '$\neq$' relations in the network [14].

For non-convex interval network, the minimal labels of the network can be approximately achieved in $O(n^5k^3)$ complexity algorithm, or $O(n^3k^3)$ when considering only on the affected relations [1]. However, if we restrict the internal relations between subintervals to be pointizable interval relations (SIA) [14] and transform into MPE network, our algorithm yields the exact solutions. The complexity of our algorithm is proportional to the number of '$\neq$' edges, the worst case would seldom occur in the real world domain.

Currently, we are working on improving complexity of the algorithm we presented in this paper by avoiding the redundant computation of the unconstrained relations, and on constructing the algorithm for finding a consistent scenario for the MPE network.

Further work is required to develop a system that can exactly handle the relations with the disjointedness relations such as **before or after** among non-convex intervals. Those relations play a crucial role in planning and scheduling [6]. We hope to enhance the proposed point-based framework to handle such information.

References

[1] L. Al-Khatib. *Reasoning with Non-Convex Time Intervals.* PhD dissertation, Florida Institute of Technology, Melbourne, Florida, 1994.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Readings in Knowledge Representation*, pages 510–521, 1983.

[3] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[4] J. Allen and H.A. Kautz. A model of naive temporal reasoning. *Formal Theories of the Commonsense World*, pages 251–268, 1985.

[5] J. Allen and J.A. Koomen. Planning using a temporal world model. *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 741–747, 1983.

[6] A. Gerevini and L. Schubert. On point-based temporal disjointness. *Artificial Intelligence*, pages 347–361, 1994.

[7] A. Gerevini and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of ACM*, pages 1108–1133, 1992.

[8] P.J. Hayes. The naive physics manifesto. *Expert Systems*, 1979.

[9] I.S. Kohane. *Temporal Reasoning in Medical Expert Systems.* MIT Laboratory for Computer Science, Technical Report TR-389, Cambridge, 1987.

[10] P. Ladkin. Time representation: A taxonomy of interval relations. *Proceedings of AAAI-86*, pages 360–366, 1986.

[11] P. Ladkin and R.D. Maddux. On binary constraint problems. *Journal of ACM*, 41(3):435–409, 1994.

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

[13] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[14] P. Van Beek. *Exact and Approximate Reasoning about Qualitative Temporal Relations.* Technical Report TR-90-29, University of Alberta, Edmonton, Alberta, Canada, 1990.

[15] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. *Proceedings of AAAI-86*, pages 377–382, 1986.