A Spatial/Temporal Relation Computing Technology for Multimedia Presentation Designs

Timothy K. Shih, Y. H, Wang, C. H. Kuo
Lawrence Y. Deng, and D. R. Jiang
Dept. of Computer Science and
Information Engineering
Tamkang University
Tamsui, Taiwan 251, R.O.C.
email: TSHIH@CS.TKU.EDU.TW

Wen C. Pai and C. C. Wang
Kuang Wu Institute of
Technology and Commerce
Paitou, Taipei
Taiwan 112
R.O.C.

Abstract

Relations among temporal intervals can be used to assist the automatic generation of multimedia presentations. In this paper, we analyze the domains of interval temporal relations. A set of algorithms is proposed to derive reasonable relations between intervals. Possible conflicts in the user specification are firstly detected and eliminated. Our mechanism then constructs partial order relations among temporal intervals before the presentation time chart is built. The algorithm is extended for objects in an arbitrary n-dimensional space. Thus, presentation layouts in a 2-D space, or Virtual Reality object representations in a 3-D space can be constructed. We use our algorithms to design a reasoning system that generates the schedule and layout of multimedia presentations. The main contributions of this paper are in its theoretical analysis of interval relation composition and a systematic approach for automation. We hope that, with our analysis and algorithms, the knowledge underlying temporal interval relations can be used in many computer applications, especially those in multimedia computing.

Key words: Temporal Interval Relations, Multimedia Presentations, Spatial/Temporal Model, Partial Order Relations, Graph Applications

1 Introduction

Multimedia applications usually contains a number of multimedia resources to be presented sequentially or concurrently. Temporal interval relations among resources are provided by the users. These resources need to be analyzed to ensure that there is no conflict among resources. Moreover, some of these resources, such as video clips or animations, occupy screen space. The spatial relations among resources need to be computed and represented in the multimedia program. The work discussed in [1] only states temporal interval relations. We found that these relations can be generalized for spatial modeling.

The contributions of our paper are to give a complete discussion of different possible domains of interval relations based on graph representations. A set of algorithms are developed to derive multimedia presentation schedules and layouts from user specifications. Possible conflicts of user specifications are eliminated. Most importantly, we extend the algorithms to compute relations among objects in an arbitrary n-dimensional space. Section 2 discusses the relation domains and their properties. In section 3, algorithms are presented. The extension of these algorithms are given in section 4. Issues of realizing an automatic system is given in section 5. The design of graphical user interface of our system is given in section 6. And finally, our conclusions are given in section 7.

2 The Relation Domains

According to the interval temporal relations introduced in [1], there exists thirteen binary relations between two temporal intervals. These relations were used in many spatial/temporal computation researches [2] of multimedia applications, including the one we proposed [3]. Allen's work discussed in [1] includes a table showing the composition of interval temporal relations. Based on the table, we propose two algorithms in this paper, using the directed graph, for interval relation compositions. These algorithms can be used to compute the binary relation between an arbitrary pair of intervals. In sections 4, we extend our algorithms for objects in an arbitrary n-dimensional space.

1060-3425/98 $10.00 © 1998 IEEE
Proc. 31st Annual Hawaii International
Conference on System Sciences

The composition of interval temporal relations may result in an *unknown derivation*. For instance, if "X before Y" and "Y after Z", there is no information that can be derived between X and Z. On the other hand, the composition may result in a *multiple derivation*. For example, if "X before Y" and "Y during Z", the composed relation for X and Z could be "before", "overlaps", "meets", "during", or "starts". These derived relations are called *reasonable relations* in our discussion. A *reasonable set* is a set of reasonable relations according to our definition. A reasonable set can not be empty, since there must exist at least one relation between any two intervals, assuming that they are in the same one dimensional space (i.e., the time line). However, a reasonable set may contain all 13 relations, which also denote that there is no information can be derived.

In the multimedia presentation scheduling system we proposed [3], the temporal relations of multimedia objects are provided by the user. In some cases, relation compositions may result in a *conflict derivation* due to the user specification. However, we did not consider this case in [3]. For example, if specifications "X before Y", "Y before Z", and "X after Z" are declared by the user, there exists a conflict between X and Z. We can not tell whether it is "X after Z" or "X before Z". The new algorithms we propose in this paper overcome our previous strategy. The new presentation scheduling system is able to issue an error message showing the conflict and suggest the user to choose a correct relation.

We analyze the domain of interval temporal relations and use an directed graph to compute the relations of multimedia objects. In the computation, we consider all possibilities: the unknown derivations, the multiple derivations, and the conflict derivations. Some terms used in the graph are defined as the following:

Definition: An *user edge* denotes a relation between a pair of objects defined by the user. The relation may be reasonable or non-reasonable. ■

Definition: A *derived edge* holds a non-empty set of reasonable relations derived by our algorithm. The relation of the two objects connected by the derived edge can be any reasonable relation in the set. ■

For each pair of objects in the time line, there exists a set of possible binary relations held between the pair of objects. For an arbitrary number of objects (denoted by nodes), some of the relations (denoted by edges) are specified by the user while others are derived. If there exists a cycle in the directed relation graph, a conflict derivation may occur. However, if there exists no cycle, there is no conflict. There may exist an unknown derivation which represents that there is no enough information to derive a relation between a pair of objects. Based on the above considerations, we suggest that the computation domain reveals four types, as discussed below.

The *complete relation domain* is a complete graph which contains possible conflicts. We want to find a *reasonable relation domain* containing no conflict derivation. Note that, in these two domains (i.e., the complete and the reasonable), both user edges and derived edges exist. If there is a conflict among a set of user edges, one of the user edge must be removed from the cycle, or the relation of that user edge must be re-assigned. If there is no conflict, the two domains are equal.

The *reduced relation domain* contains relations specified by the user only. It is possible that the user issues a conflict situation. To avoid the occurrence of conflicts, we place a restriction on the user's interaction. Instead of allowing the user to add an arbitrary relation to the relation graph, we only allow the user to add objects to a *restricted relation domain*, which is a tree and a subdomain of the reduced relation domain. That is, when the user is about to add a new edge, the user either adds a new node connected to an existing node via an user edge, or joins two sub-trees via the user edge. No cycle is created in the restricted relation domain. Thus, the conflict situation does not exist. When deleting an user edge, the user has to maintain the connectivity of the tree. If all nodes are connected, the user specification is complete. Otherwise, the presentation system should alert the user to complete the specification. The above domains can be summarized as the following with figure1 illustrating their set diagrams:

- The **complete relation domain** (a complete graph): contains user edges and derived edges, with possible cycles and possible conflicts.
- The **reasonable relation domain** (a graph): contains user edges and derived edges, with possible cycles but no conflict.
- The **reduced relation domain** (a graph): contains only user edges, with possible cycles and possible conflicts.
- The **restricted relation domain** (a tree): contains only user edges, without cycle.

The four domains are used in the analysis and computation of object relations. In the next section, we propose two algorithms computing the reasonable relation domain.

3 Generating Presentation

This section discusses a serial of algorithms to generate multimedia presentations from user specifications. Firstly, we discuss two algorithms for computing the reasonable relation domain, which contains no conflict and thus can be used for generating schedule or layout of a presentation. Next, a representation of partial

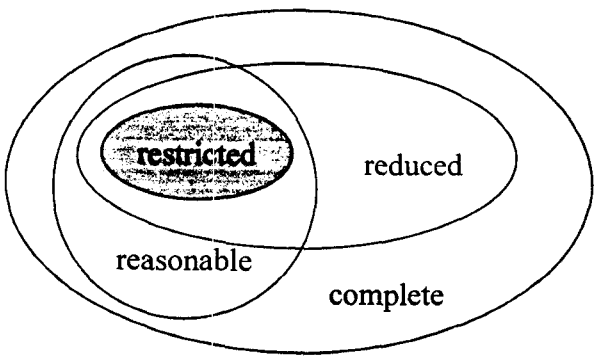

Figure 1: Relation Domains

order relation is used to denote the topological order of objects. An algorithm takes as input a reasonable relation domain and generates these partial order sets (POSets) is presented. Another algorithm generates schedule and layout from the POSets is also given.

3.1 Computing Reasonable Relations

The first algorithm computes the reasonable relation domain from the reduced relation domain. User edge conflicts are eliminated and derived edges and cycles without conflict are added. The second increases user edges one by one in a restricted domain, and the corresponding derived edges are added to the reasonable relation domain. The resulting domain may have cycles due to the insertion of derived edges. But no conflict occurs.

Algorithm 1

The purpose of the first algorithm is to add derived edges to the reduced relation domain. If there is a conflict cycle in the original reduced relation domain, the algorithm eliminates that conflict first by alerting the user to select a reasonable relation. Thus, the resulting reasonable relation domain is a complete graph, which is equal to the complete relation domain. This conflict elimination is achieved by invoking the *EliminateConflicts* algorithm. Suppose G is a graph of the reduced relation domain, and GV and GE are the vertex set and edge set of G, respectively. Initially the reasonable relation domain is set to the reduced relation domain. The algorithm computes derived edges based on user edges. The reason of using the user edges is that these edges contain the minimal and sufficient information of what the user wants. If the algorithm computes derived edges from other derived edges, eventually, the algorithm has to compute the set intersection of all possible derivations for the reasonable set of the new derived edge. The algorithm starts from taking each path of user edges of length 2, and computes a derived edge from that path. The insertion of edge $e = (a, b)$ results a cycle, but no conflict. The reasonable set of edge e (i.e., $e.rs$) is computed from two edges, (a, n_{k-1}) and (n_{k-1}, b), which are user edges or derived edges. Since we increase the path length, pl, of the path of user edges one by one. the derived edge (a, n_{k-1}) (or user edge, if $pl = 2$) must have been computed in a previous iteration. The algorithm repeats until all edges are added to the complete graph K_n, which contains $n*(n-1)/2$ edges. The first algorithm, *ComputeRD*1, is given below:

Algorithm : *ComputeRD*1
Input : $G = (GV, GE)$
Output : $K_n = (K_n V, K_n E)$
Preconditions : *true*
Postconditions : $GV = K_n V \wedge GE \subseteq K_n E$
Steps :
1 : $G =$ *EliminateConflicts*(G)
2 : $K_n = G \wedge pl = 2$
3 : repeat until $\mid K_n E \mid = \mid K_n V \mid * (\mid K_n V \mid -1) \; / \; 2$
  3.1 : for each $e = (a, b) \wedge e \notin K_n E \wedge a \in K_n V \wedge b \in K_n V$ such that there is a path of user edges from a to b, with path length $= pl$
  3.2 : suppose $((n_1, n_2), (n_2, n_3), ..., (n_{k-1}, n_k))$ is a path with $a = n_1 \wedge b = n_k \wedge k = pl + 1$
  3.3 : set $e.rs = RelComp((a, n_{k-1}).rs, (n_{k-1}, b).rs)$
  3.4 : $K_n E = K_n E \cup \{ e \}$
  3.5 : $pl = pl + 1$

In *ComputeRD*1, we use the conflict elimination algorithm. A conflict occurs only if there is a cycle. For each cycle in the reduced relation domain, the *EliminateConflicts* algorithm finds a derived edge between any two consecutive edges, namely, (n_i, n_{i+1}) and (n_{i+1}, n_{i+2}). The algorithm then checks if the last user edge making the cycle represents a relation (i.e., $(n_k, n_{k-1}).r$) belongs to the reasonable set computed for the user edge (i.e., rs). If not so, the algorithm asks the user to choose an arbitrary relation r' belongs to the reasonable set and use the relation to replace the original one.

Algorithm : *EliminateConflicts*
Input : $G = (GV, GE)$
Output : $G' = (G'V, G'E)$
Preconditions : G contains only user edges $\wedge$ $G' = G$
Postconditions : $G' = G$, but the reasonable sets of edges in G' may be changed
Steps :
1 : for each $P = ((n_1, n_2), (n_2, n_3), ..., (n_{k-1}, n_k))$ in G', with $n_1 = n_k \wedge k > 3$
  1.1 : for each $i, 1 \leq i \leq k-2$
    1.1.1 : set $(n_i, n_{i+2}).rs =$ $RelComp((n_i, n_{i+1}).rs, (n_{i+1}, n_{i+2}).rs)$
  1.2 : $rs = RelComp((n_k, n_{k-2}).rs, (n_{k-2}, n_{k-1}).rs)$
  1.3 : if $(n_k, n_{k-1}).r \notin rs$ then
    1.3.1 : ask user to choose an $r' \in rs$
    1.3.2 : set $(n_k, n_{k-1}).r = r'$

In function *RelComp*, the reasonable set computed must be the union of all possible combinations of the pair of relations obtained from the two input reasonable sets, namely, rs_1 and rs_2. The function uses a table lookup function to obtain a set of reasonable relations. The *TableLookUp* function (definition omitted) uses the relation composition table discussed in [1], if the algorithm is to compute relations of objects in a 1-D space. We have another table for objects in a 2-D space discussed in section 4. However, it is the same algorithm to compute the reasonable relation domain. Only the amount and type of relations are changed (i.e., changes from 1-D relations to 2-D relations). The following is the function computes a reasonable set:

Algorithm : *RelComp*
Input : rs_1, rs_2
Output : rs
Preconditions : *true*
Postconditions : *true*
Steps :
1 : $rs = \bigcup_{\forall r_1 \in rs_1, \forall r_2 \in rs_2 \bullet (r1,r2) \in rs_1 \times rs_2}$
$TableLookUp(r_1, r_2)$

In the actual implementation of a multimedia presentation system, depending on the user's specification, directions of user edges are easily decided and represented in the implementation.

The first algorithm assumes that a set of relations is provided by the user. However, in an interactive system, the user may incrementally adds user edges. The set of relations is unknown before the completion of the multimedia specification. In the next algorithm, we allow the user edges to be added to the graph one by one. Thus, our algorithm is more realistic for a system of interactive multimedia presentation designs.

Algorithm 2

The restricted relation domain is a tree. The reason for using a tree is to avoid cycles which may introduce conflicts. A multimedia presentation contains a number of objects. When a new object is added to the presentation, an user edge and a node representing the new object is added. A number of derived edges are also inserted. Adding a new node to the complete graph K_n requires adding n new edges, where n is the number of nodes, to complete K_{n+1}. Since a complete graph is strongly connected, there exists an edge between each pair of nodes. When a new user edge is added, we can compute other derived edges from checking the composed relations between an existing edge and this new user edge. Algorithm *AddUL* adds an user edge $l = (a, b)$ to a complete graph K_n:

Algorithm : *AddUL*
Input : $l = (a, b), K_n = (K_n V, K_n E)$
Output : $K_{n+1} = (K_{n+1} V, K_{n+1} E)$
Preconditions : $l \notin K_n E \wedge a \in K_n V \wedge b \notin K_n V$
Postconditions :$\mid K_{n+1} V \mid = \mid K_n V \mid + 1 \wedge$
$\mid K_{n+1} E \mid = \mid K_n E \mid + n$
Steps :
1 : $K_{n+1} E = K_n E \cup \{ l \}$
2 : for each $e = (c, b) \wedge c \neq a \wedge c \in K_n V$
2.1 : $e.rs = \bigcap_{\forall d \in K_n V, (c,d) \in K_n E, (d,b) \in K_n E}$
$(RelComp((c, d).rs, (d, b).rs))$
2.2 : $K_{n+1} E = K_{n+1} E \cup \{ e \}$
3 : $K_{n+1} V = K_n V \cup \{ b \}$

The following is the second algorithm which adds all edges in the restricted relation domain T to the complete graph K_n:

Algorithm : *ComputeRD2*
Input : $T = (TV, TE)$
Output : $K_n = (K_n V, K_n E)$
Preconditions : T does not contain any cycle
Postconditions : $TV = K_n V \wedge TE \subset K_n E$
Steps :
1 : $K_n V = \{ a \} \wedge a \in TV \wedge K_n E = \emptyset$
2 : for each $e \in TE \wedge e \notin K_n E$
2.1 : $K_n = AddUL(e, K_n)$

3.2 Computing the POSets

After the reasonable relation domain is computed. Conflicts in a user specification, if any, are eliminated. The next step is to use these information to assist a multimedia presentation system to generate presentation schedule and layout. There are two further steps for the automation. Firstly, the order of presentation objects, whether as a time line or as a two dimensional layout, must be decided. Secondly, the schedule and layout must be generated. The second step is discussed in section 3.3. The first is achieved via function *ComputePOSet* discussed in this section.

For each time interval, represented as a 1-D object, there is a starting point and an ending point. Assuming that there is no conflict, it would be nice to use a partial order relation to formulate a set of time intervals. However, intervals may overlap or embrace one another. An interval X with starting point before the one of interval Y may have its ending point after the ending point of Y. To represent the order, we use two partial order sets (POSets), one for the starting points and another for the ending points. For each pair of objects, after their relation is given by the user, the order of points are decided, as illustrated in figure 2. In order to state the relations between these two POSets, a *duration label* for each object is used. These labels

connect starting and ending points. When the duration or sizes of each multimedia object is obtained, duration labels are attached with a number.

According to the temporal interval relations given in [1], the order of starting and ending points are decided. For example, if "X equals Y", the starting and ending points of X and Y are equal. If "X during Y", Y's starting point is before the one of X. But their ending points are in an opposite direction. The *initiation connections* and the *termination connections* given in figure 2 picture the order of points for seven relations. Similarly, those for the six inverse relations can be constructed.

Function *ComputePOSet* takes as input a reasonable relation domain and generates two POSets. An arbitrary node of $K_n V$ is chosen. The duration label of this node is added. Next, the algorithm performs a depth first search (or breadth first search) on the reasonable relation domain via user edges. Note that, for each user edge (X, Y), one can always find an inverse user edge (Y, X) denoting the inverse relation of that user edge. Thus, graph traversal is allowed in each direction of an user edge. Nodes and edges in the initiation POSet and termination POSet are added one by one. And duration labels are decided. Since there is no conflict cycle and the graph is connected, the two POSets are constructed after n iterations, where n is the number of objects. However, the number of nodes in the POSets are not necessary equal to n, as initiation or termination connections may join nodes. When a set of object relations is given, the initiation POSet and termination POSet are generated by the following algorithm:

Algorithm : *ComputePOSet*
Input : $K_n = (K_n V, K_n E)$
Output : $InitPOSet = (IV, IE)$,
$TermPOSet = (TV, TE)$
Preconditions : K_n is a reasonable relation domain
Postconditions : *true*
Steps :
1 : choose an arbitrary node $v \in K_n V$
1.1 : let $IV = \{ v \} \wedge TV = \{ v \} \wedge$
$IE = \emptyset \wedge TE = \emptyset$
1.2 : add duration label of v
2 : for each user edge $ue \in K_n E$, perform DFS on K_n
2.1 : add init connection of ue to *InitPOSet*
2.2 : add term connection of ue to *TermPOSet*
2.3 : add duration label of the new node from ue

The temporal order of objects is represented in two POSets, one for the starting points and another for the ending points. Using a similar concept, the spatial order of 2-D objects is captured in four POSets, two for the starting and ending boundaries in the X coordinate and the other two for the Y coordinate. In general, for an arbitrary n-dimensional space, we need n pairs of POSets.

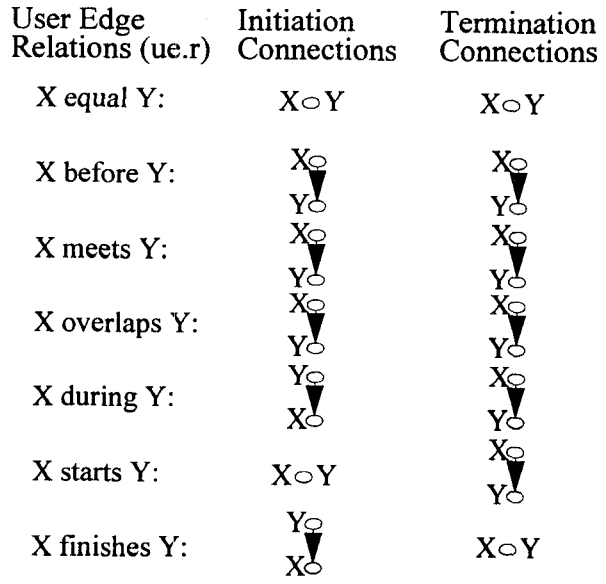

Figure 2: The orders of starting and ending points

3.3 Generating Layout

After the initiation POSet and termination POSet are constructed, the next step is to construct the presentation schedule or layout. We present an algorithm for scheduling first. And the algorithm is then extended for layout construction.

Assuming X and Y are temporal intervals, a number of properties are defined below:

- $X.s$: denotes the starting time of interval X.
- $X.e$: denotes the ending time of interval X.
- $X.d$: denotes the duration of interval X.
- (X, Y).t: is the time difference between the starting time of interval X and the starting time of interval Y.

Similar properties for interval Y are defined. The temporal interval relations given in [1] provide the orders of starting and ending points of two intervals. However, to schedule a presentation, precise timing is required. Thus we add parameters to these temporal relations. We define parameters for three relations (and their inverse relations):

- before(X, Y, n): X is before Y, n is the time difference between starting points of X and Y.
- overlaps(X, Y, n): X overlaps Y, n is the time difference between starting points of X and Y.
- during(X, Y, n): X is during Y, n is the time difference between starting points of X and Y.

However, other relations do not require a parameter. For example, if "X starts Y", we can always find

the durations of objects X and Y from a multimedia resource database. Thus, we know the difference between the two ending points of X and Y. After parameters are added, assuming that $X.s$ is known and $X.d$ and $Y.d$ are obtained from the multimedia resource database, we can always derive $Y.s$. Also, we know that $X.e = X.s + X.d$, $Y.e = Y.s + Y.d$, and $(X, Y).t = Y.s - X.s$. The time difference $(X, Y).t$ is used in the *ComputeRT* algorithm. The following table summarize these computations of relations:

Table 1: Parameterized Temporal Interval Relations

Parameterized Relations	Starting Time
equal(X, Y)	Y.s = X.s
before(X, Y, n)	Y.s = X.s + n
meets(X, Y)	Y.s = X.s + X.d
overlaps(X, Y, n)	Y.s = X.s + n
during(X, Y, n)	Y.s = X.s - n
starts(X, Y)	Y.s = X.s
finishes(X, Y)	Y.s = X.s - (Y.d - X.d)

Next, from the initiation POSet, we can compute a *relative time table* (RT) for the starting points of temporal intervals. Algorithm *ComputeRT* takes as input an initiation POSet, and computes the relative time table, which is denoted by a function mapping from nodes to integers or an undefined symbol (which means that the relative time is not yet computed):

Algorithm : *ComputeRT*
Input : $InitPOSet = (IV, IE)$
Output : $RT = IV \rightarrow Integer \cup \{ Undefined \}$
Preconditions : $\forall v \in IV \bullet RT(v) = Undefined$
Postconditions : $\forall v \in IV \bullet RT(v) \in Integer \wedge V = IV \wedge E = IE$
Steps :
1 : let $v \in IV \wedge RT(v) = 0 \wedge V = \{ v \} \wedge E = \emptyset$
2 : for each $e = (a, b) \in IE \wedge e \notin E \wedge (a \in V \vee b \in V)$
  2.1 : if $a \in V$ then
    2.1.1 : if $b \notin V$ then
      $RT(b) = RT(a) + e.t \wedge$
      $V = V \cup \{ b \} \wedge E = E \cup \{ e \}$
    2.1.2 : else if $RT(b) \neq RT(a) + e.t$ then *Error*
  2.2 : else if $b \in V$ then
    2.2.1 : if $a \notin V$ then
      $RT(a) = RT(b) - e.t \wedge$
      $V = V \cup \{ a \} \wedge E = E \cup \{ e \}$
    2.2.2 : else if $RT(a) \neq RT(b) - e.t$ then *Error*
3 : choose $t \in Integer$ such that $\forall v \in IV \bullet t \leq RT(v)$
4 : for each $v \in IV$, let $RT(v) = RT(v) - t$

Initially, all relative time of nodes are undefined. The algorithm chooses an arbitrary node from the POSet and set the relative time of that node to zero. V and E are node set and edge set keep track of nodes and edges visited. Each time the algorithm finds a new edge, $e = (a, b)$, connected to the part of graph traversed. Depending on whether the new edge is connected from the graph (i.e., $a \in V$) or connected to the graph (i.e., $b \in V$), the algorithm computes the relative time of the new node added to the graph. However, if the node to be added is already in the graph, the algorithm checks for inconsistency and reports a possible error message. After all edges in IE and nodes in IV are added to E and V, respectively, the algorithm finds the smallest relative time of all nodes, and adjusts the relative time table accordingly. The object with a zero starting relative time is the one who should start the multimedia presentation. The schedule of ending points can be computed by the same algorithm by using the termination POSet as input.

Using a similar concept, we can decide the spatial order of 2-D objects. The X and the Y coordinates of object starting positions are computed separately. Thus the layout of a multimedia presentation is generated. However, the layout may change according to time. In section 5.1, we will discuss a technique to bridge temporal and spatial information of a presentation.

4 Extending the Algorithms

Allen's work [1] discusses relations for 1-D objects (e.i., time intervals). In this section, we introduce a mechanism to extend the relations of objects to an n-dimensional space. Relations of 2-D objects can be used in screen layout designs. Relations of 3-D objects can be used in 3-D graphics, such as Virtual Reality applications. A cube in 3-D space can be projected onto a 2-D plane. Similarly, a square is projected to a line segment. If we look at two objects in the n-dimensional space, we can project the positional relation between these two objects from n directions to n 1-D space. Thus, an n-dimensional relation can be formularized by a conjunction of n 1-D interval relations.

Let $R1$ denote a 1-D temporal interval relation discussed in section 2. The relation composition table discussed in [1] is a function maps from the Cartesian product of two $R1$s to a set of $R1$s (denoted by $\mathbf{P}\ R1$). Assuming that t^1 is the mapping function interpreting Allen's table, we can compute t^2, the relation composition table of 2-D objects, and t^3, the one for 3-D objects, from t^1. There are 13 relations for 1-D objects. A conjunction of two 1-D relations, which denotes a 2-D relation, has 13^2 variations. Similarly, there are 13^3 3-D relations. Fortunately, 4-D relations are not quite applicable and the memory space required for 2-D and 3-D relation tables is manageable by nowaday computers.

Following the notations used in [1], "<" denotes the "before" relation, ">" is the "after" relation, and so on. The set of 1-D relations, $\mathbf{P}\ R1$, is due to [1]. Note that, "e" denotes the "equal" relation. Since a 2-D

relation is a conjunction of two 1-D relations, we use the notation, r_1 x r_2, to denote a 2-D relation, where r_1 and r_2 are two 1-D relations. Thus, t^2 is a mapping from the Cartesian product of two $R1$ x $R1$s to $\mathbb{P}$ $R1$ x $R1$. In $\mathbb{P}$ $R1$ x $R1$, there are 169 2-D relations. Similarly, t^3 is represented. The following are signatures of these relation tables:

$$t^1 = R1 \times R1 \rightarrow \mathbb{P}\ R1$$
$$\mathbb{P}\ R1 = \{ <, >, d, di, o, oi, m, mi, s, si, f, fi, e \}$$
$$t^2 = R1\ \mathbf{x}\ R1 \times R1\ \mathbf{x}\ R1 \rightarrow \mathbb{P}\ R1\ \mathbf{x}\ R1$$
$$\mathbb{P}\ R1\ \mathbf{x}\ R1 = \{ <\ \mathbf{x}\ <, <\ \mathbf{x}\ >, <\ \mathbf{x}\ d, <\ \mathbf{x}\ di, \ldots \}$$
$$t^3 = R1\ \mathbf{x}\ R1\ \mathbf{x}\ R1 \times R1\ \mathbf{x}\ R1\ \mathbf{x}\ R1 \rightarrow \mathbb{P}\ R1\ \mathbf{x}\ R1\ \mathbf{x}\ R1$$
$$\mathbb{P}\ R1\ \mathbf{x}\ R1\ \mathbf{x}\ R1 = \{ <\ \mathbf{x}\ <\ \mathbf{x}\ <, <\ \mathbf{x}\ <\ \mathbf{x}\ >, \ldots \}$$

Tables t^2 and t^3 are computed according to the following formulae. Note that, each element in table t^2 contains a set of 2-D relations, which are computed from the production of two elements of table t^1:

$$\forall\ i_1\ \mathbf{x}\ j_1, i_2\ \mathbf{x}\ j_2 \in \mathbb{P}\ R1\ \mathbf{x}\ R1$$
$$t^2(i_1\ \mathbf{x}\ j_1, i_2\ \mathbf{x}\ j_2) = \prod t^1(i_1, i_2)\ \mathbf{x}\ t^1(j_1, j_2)$$
$$\forall\ i_1\ \mathbf{x}\ j_1\ \mathbf{x}\ k_1, i_2\ \mathbf{x}\ j_2\ \mathbf{x}\ k_2 \in \mathbb{P}\ R1\ \mathbf{x}\ R1\ \mathbf{x}\ R1$$
$$t^3(i_1\ \mathbf{x}\ j_1\ \mathbf{x}\ k_1, i_2\ \mathbf{x}\ j_2\ \mathbf{x}\ k_2) =$$
$$\prod t^1(i_1, i_2)\ \mathbf{x}\ t^1(j_1, j_2)\ \mathbf{x}\ t^1(k_1, k_2)$$
$$where \prod A\ \mathbf{x}\ B = \{\ a\ \mathbf{x}\ b \mid \forall\ a \in A, b \in B\ \}$$
$$\prod A\ \mathbf{x}\ B\ \mathbf{x}\ C = \{\ a\ \mathbf{x}\ b\ \mathbf{x}\ c \mid \forall\ a \in A, b \in B, c \in C\ \}$$

Table generated by the above formulae are stored in memory to reduce run-time computation load. These tables are used in the algorithm discussed in section 3.1 (i.e., the *TableLookUp* algorithm), depending on which dimension of objects the algorithm is computing.

5 The Automation

We use our algorithms to design a system for multimedia presentation designs. One dimensional relations are used for scheduling and two dimensional relations are for layout generation. We try to provide a set of relation ICONs and a graphical user interface for the user to click, drag and drop multimedia resource relations. Thus, the necessary information of generating schedule and layout is declared. When a user is about to change the relation between two resources, the user does not need to worry about relations of other objects. This is the most important advantage of using our system. However, considering the amount of relations, it is quite difficult for us to design and for the user to use 13 1-D relation ICONs and 169 2-D relation ICONs. Therefore, synthesizing relations is a need. Assuming $X.d$ denotes the duration of interval X, and $Y.d$ is the one for Y, and according to the parameterized relations discussed in section 3.3, we combine the following five relations into the *parallel*(X, Y, n) relation:

Table 2: Combining Parallel Relations

Parameterized Relations	Conditions
equal(X, Y)	$n = 0 \wedge Y.d = X.d$
during(X, Y, n)	$n > 0 \wedge Y.d > n + X.d$
starts(X, Y)	$n = 0 \wedge Y.d > X.d$
finishes(X, Y)	$n > 0 \wedge Y.d = n + X.d$
overlaps$^{-1}(X, Y, n)$	$n > 0 \wedge Y.d < n + X.d$

For example, given the relation *parallel*(X, Y, n), if $n > 0$ and $Y.d > n + X.d$, then the relation *during*(X, Y, n) is used by the system for scheduling. Note that *parallel*(X, Y, n) is specified by the user via an ICON. But $X.d$ and $Y.d$ are retrieved from the resource database. Similarly, we combines two relations into the *sequential*(X, Y, n) relation:

Table 3: Combining Sequential Relations

Parameterized Relations	Conditions
before(X, Y, n)	$n > 0$
meets(X, Y)	$n = 0$

We design two ICONs for the *parallel* and the *sequential* temporal relations, as shown in figure 3. The boxes denoted by "**X**" and "**Y**" are to keep resource ICONs which represent multimedia resources selected by the user via our resource browser. The small boxes denoted by "**n**" are filled with integers by the user for the timing parameter of those two relations. Since the two temporal relations (and their inverse relations) cover all 13 relations introduced in [1], using our two ICONs and the graphical user interface enables the user to specify any temporal relations between two multimedia resources.

To design spatial ICONs is a little complicate. Based on the two temporal ICONs, we derive four basic spatial ICONs, as shown in figure 3. Note that, the concepts of parallel and sequential relations are used in the spatial ICONs for controling object overlaping. Two rectangles can be parallel to each other in the X or in the Y coordinate. Only if both coordinates overlap, the two rectangles overlap physically, as shown in the parallel-parallel spatial ICON. The other three basic spatial ICONs contain rectangles either overlap in zero or one coordinate. Thus, those rectangles do not overlap with each other physically.

For the two temporal ICONs, inverse relations can be specified by swapping the two multimedia resources. However, in the case of spatial ICONs, inverse relations occur either in the X or in the Y coordinate (or both). Thus, the swapping of resources in X and Y coordinates are dependent. Therefore, for each basic spatial ICON in figure 3, there are four possible cases. Considering the X box (i.e., resource X) fixed in coordinate, the Y box (i.e., resource Y) can be on the north-west, the north-east, the south-west, or the south-east of the X box. However, the north-west case and the south-east case are inverse relations, as well as the north-ease case and the south-west case. Thus, only four spatial ICON

variations are used (see figure 3). Similar to specifying a timing parameter of a temporal relation, spatial ICONs use box "**m**" and "**n**" for spacing information in the X and the Y coordinates. Two dimensional relations are extended from those of 1-D's. Since the four basic spatial ICONs are extended from the parallel and the sequential temporal ICONs, and all inverse relations can be derived, we found that the 8 spatial ICONs covers all cases of spatial relations.

5.1 Linking the Temporal and the Spatial Relations

A presentation schedule is a collection of temporal intervals. Each interval has a starting point and an ending point. These points change the state of a presentation. A *state change point* in a presentation, in our definition, is a time point which contains at lease one starting or ending point of temporal intervals, except the ending point(s) of the last interval(s). Since multimedia presentations are dynamic, the definition of presentation layout is with respect to a time point. It is the state change points that we consider for these time points. The layout of presentation with respect to a state change point is defined to be the collection of multimedia resource locations on the screen, starting from the state change point and ending at the next state change point (exclusive), or the end of the presentation.

The relative time table discussed in section 3.3 contains the starting point of intervals. The ending points are easily computed by using the relative time table and the duration labels. Therefore, a set of state change points is computed. For each point in the set, we maintain links to the objects in the spatial relation POSets. Thus, the layout design with respect to each state change point is constructed and stored.

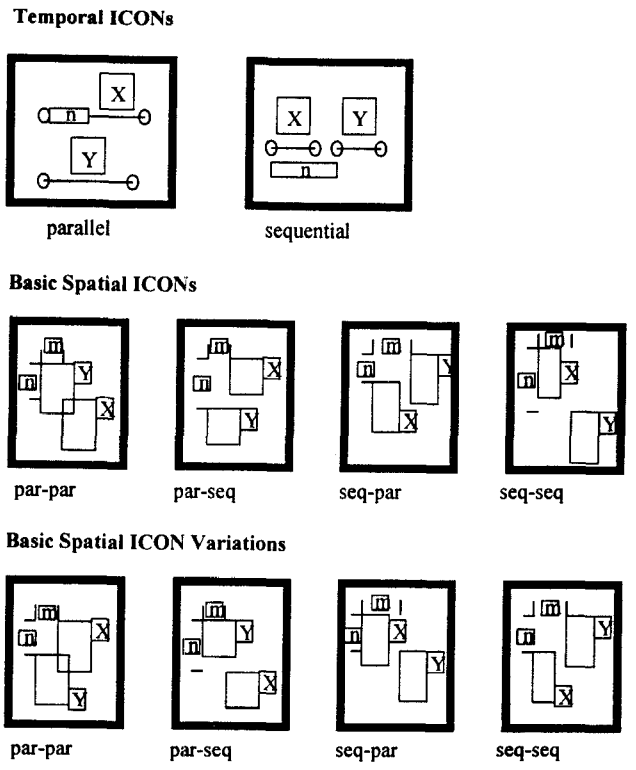

Figure 3: Temporal and Spatial ICONs

6 The Graphical User Interface

Before a user is about to design his/her presentation, multimedia resources are collected. A preliminary script of the presentation is also decided. However, it is quite difficult for the user who is not familiar with computer programming to design a presentation using spatial or temporal relations. For the convenience of the user, we develop an ICON programming technique as well as its graphical user interface.

When a presentation designer is about to design his/her presentation, after the presentation topics and a rough schedule are designed, the designer will collect resources, convert them to some digital forms, and store these resources with the properties discussed in our multimedia database. Figure 4 shows a resource browser allows the presenter to insert/delete resources, and to retrieve multimedia resources according to properties

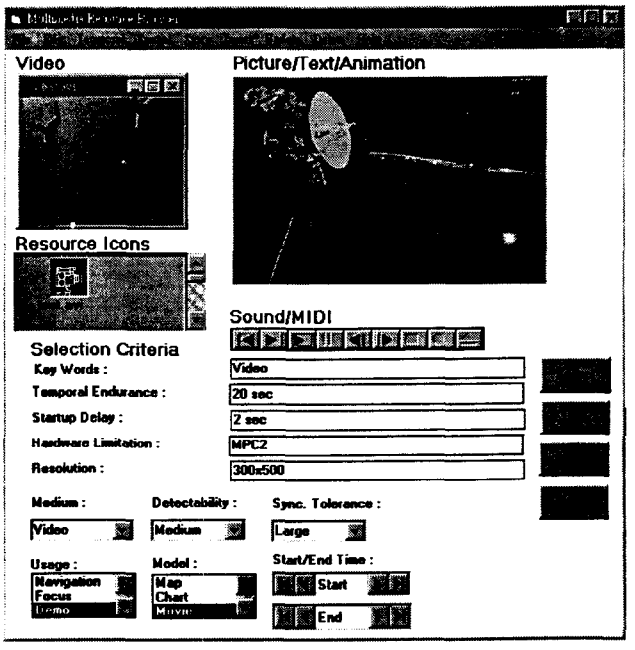

Figure 4: The Multimedia Resource Browser

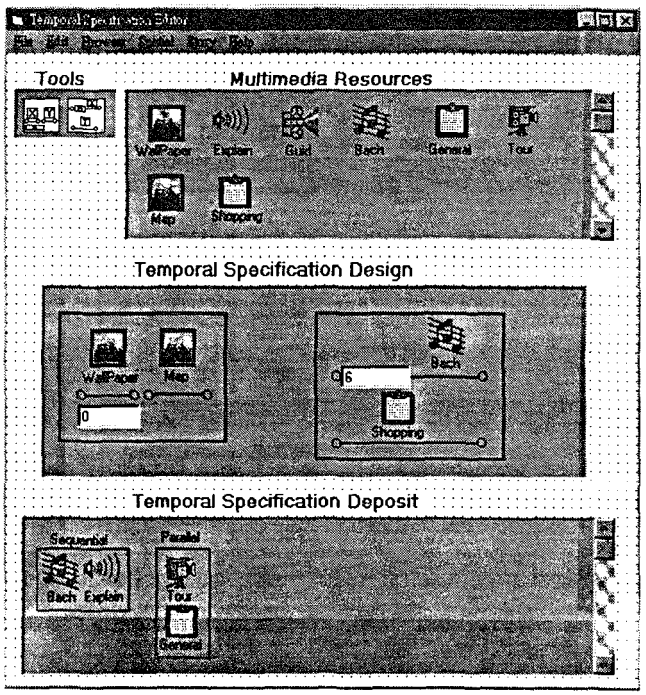

Figure 5: The Temporal Specification Editor

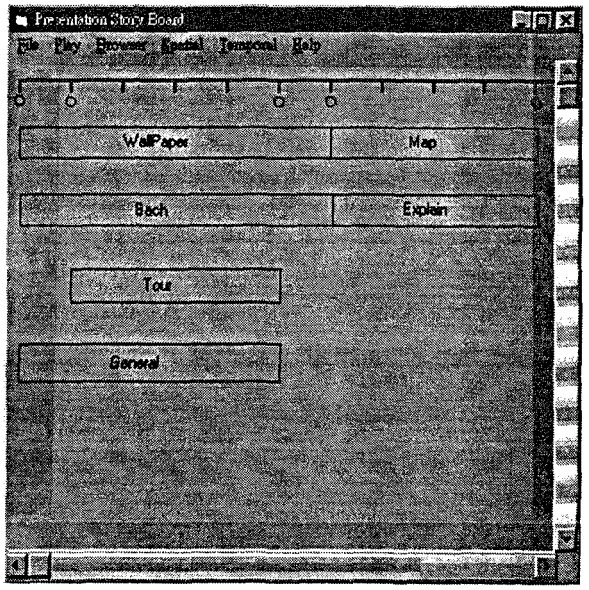

Figure 6: The Presentation Story Board

given in separated text or list boxes. When these properties are decided, the user is able to push the `Select` button which makes a number of resource ICONs shown in the `Resource ICONs` subwindow. The `Video`, the `Picture/Text/Animation`, and the `Sound/MIDI` subwindows are to preview these resources selected before the user pushes the `Use` button to add these resources to a presentation. These resources are used in our presentation generator.

Next, the presentation designer has to use our ICON programming technique to design the schedule of the presentation. We have two temporal ICONs shown on the left side of the temporal specification editor (see figure 5). A square in an ICON represents a place holder for a multimedia resource (indicated by its resource name). A rectangle is for the user to fill in a timing parameter (an integer). After the user selects the resources that he/she plans to use from our multimedia resource browser, in figure 5, when the user pushes one of the temporal ICONs listed in the tool bar, the user is able to drag and drop the temporal ICON in the Temporal Specification Design area. Next, the user has to fill in the resources and timing parameters by dragging resource ICONs from the Multimedia Resource area and typing in integers in the temporal ICONs. Note that, the two resource ICONs used in a temporal ICON have to follow the rule that one resource ICON is new and another is used before, except the first time. After a temporal ICON has all of its parameters filled, the ICON is highlighted. The user thus can drag this temporal ICON to the Temporal Specification Deposit area which contains the final presentation schedule design.

After the presentation designer gives the temporal relations, a Presentation Story Board is shown in figure 6. The story board has a timing ruler showing the presentation cycles. The little circles below the ruler indicate the presentation has state change points. Each point starts a segment of the presentation. Clicking on one of the points results in a layout design of the resources with respect to the state change point.

To specify the spatial information, the user has to use the Spatial Specification Editor shown in figure 7. Similar to using the Temporal Specification Editor, the user uses the drag and drop mechanism. However, there are 8 spatial ICONs for layout designs.

The temporal and the spatial specification editors produce a presentation specification based on presentation resource attributes, the temporal specification ICONs the user used, and the layout the user designed. The presentation generator takes this presentation specification, and generates a presentation.

7 Conclusions

The algorithms proposed in this paper can be used in other computer applications, for instance, a project management system. A project contains a number of tasks. Two tasks may be performed either concurrently or sequentially. They may start or end at the same time. Or, the first task may be performed after eighty percents of the second is complete. In such a management system utilizes our algorithms, if the user specifies the temporal relations of tasks, the project schedule can be generated automatically. The main contributions of

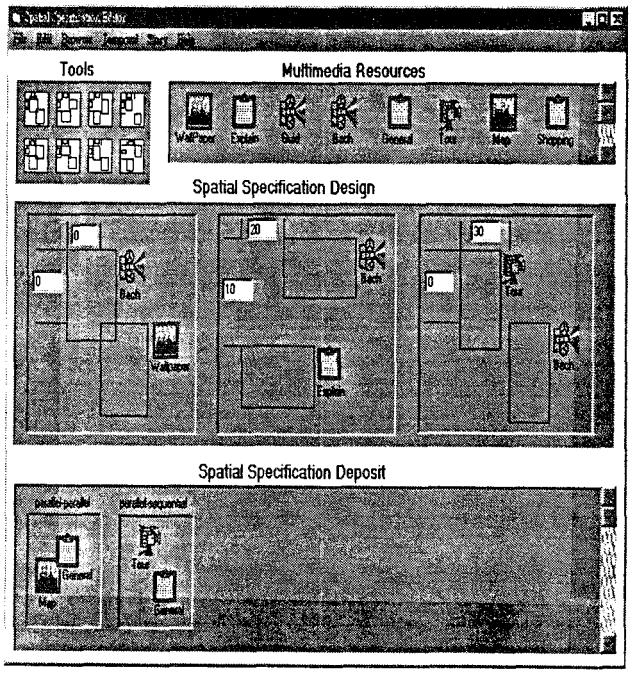

Figure 7: The Spatial Specification Editor

this paper are in its theoretical analysis of interval relation compositions and a systematic approach toward automation. We hope that, with our analysis and algorithms, the knowledge underlying temporal interval relations can be used in many computer applications, especially in multimedia computing.

References

[1] James F. Allen "Maintaining Knowledge about Temporal Intervals," Communications of the ACM, Vol. 26, No. 11, 1983.

[2] Thomas D. C. Little and Arif Ghafoor "Interval-Based Conceptual Models for Time-Dependent Multimedia Data," IEEE transactions on knowledge and data engineering, Vol. 5, No. 4, 1993, pp 551–563.

[3] Timothy K. Shih, Steven K. C. Lo, Szu-Jan Fu, and Julian B. Chang, "Using Interval Temporal Logic and Inference Rules for the Automatic Generation of Multimedia Presentations," in Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 17 – 23, 1996, pp. 425 – 428.

[4] Timothy K. Shih, Chin-Hwa Kuo, Huan-Chao Keh, Chao T. Fang-Tsou, and Kuan-Shen An, "An Object-Oriented Database for Intelligent Multimedia Presentations," in proceedings of the 1996 IEEE International Conference on Systems, Man and Cybernetics, Beijing, China, October 14 – 17, 1996.

[5] Michael Vazirgiannis, Yannis Theodoridis, and Timos Sellis "Spatio - Temporal Composition in Multimedia Applications," in proceedings of the International Workshop on Multimedia Software Development, March 25 – 26, Berlin, Germany, 1996, pp 120 – 127.

[6] Thomas Wahl, et. al., "TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia" in proceedings of the international conference on multimedia computing and systems, Washington DC, U.S.A., May 15–18, 1995, pp 274–277.