An ICON Programming Technique for Multimedia Presentation Designs

Timothy K. Shih, S. H. Yen, C. H. Kuo
Y. H, Wang, C. S. Wang, and A. Y. Chang
Dept. of Computer Science and
Information Engineering
Tamkang University
Tamsui, Taiwan 251, **China**
email: TSHIH@CS.TKU.EDU.TW

Sheng-Tun Li

Dept. of Info. Management
National Inst. of Technology
1 University Road, Yenchao
Kaohsiung, TAIWAN
China

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 2-D space, or Virtual Reality object representations in 3-D space can be constructed. We use our algorithms to design a reasoning system that generates the schedule and layout of multimedia presentations.

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 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. Finally, we implement a system under Windows 95/3.1 to allow presentation designers to construct presentations from temporal/spatial specifications.

In our system, one dimensional relations are used for scheduling and two dimensional relations are for layout generation. We 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. As a result, we propose two generic temporal relations and eight generic spatial relations, with additional spacing/timing parameters. All relations are specified in ICONs by a presentation designer. 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.

This paper is organized as the following. 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. And finally, our conclusions are given in section 5.

2 TEMPORAL 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

0-7803-4253-4/97/$10.00 © 1997 IEEE

spatial/temporal computation researches [3, 7] of multimedia applications, including the one we proposed [8]. Allen's work discussed in [1] includes a table showing the composition of interval temporal relations. Based on the table, we propose 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.

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 [8], 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 [8]. 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. An *user edge* denotes a relation between a pair of objects defined by the user. The relation may be reasonable or non-reasonable. 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:

- 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 algorithms computing the reasonable relation domain.

3 COMPUTING THE REASONABLE RELATIONS

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.

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.

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).

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 : $G = EliminateCycles(G)$
2 : $K_n = G \wedge pl = 2$
3 : repeat until $| K_n E | = | K_n V | * (| K_n V | -1) / 2$
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 : assuming that $((n_1, n_2), (n_2, n_3), ..., (n_{k-1}, n_k))$ is such 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 : $pl = pl + 1$

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 cycle $P = ((n_1, n_2), (n_2, n_3), ..., (n_{k-1}, n_k))$ in G', with $n_1 = n_k \wedge k > 3$
1.1 : for each $i, 1 \leq i \leq k-2$, 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 the user to choose $r' \in rs$
1.3.2 : set $(n_k, n_{k-1}).r = r'$

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 composition of object relations has a direction. To compose r_1 and r_2, if the direction of one of the edge is in an opposite direction, we must firstly find the inverse relation before the composition proceeds. Inverse relations between objects can be easily computed by the following formulae:

$$(r_1 \circ r_2)^{-1} = \{\ r^{-1} \mid r \in r_2^{-1} \circ r_1^{-1}\ \}$$
$$(r_1 \circ r_2 \circ r_3)^{-1} = \{\ r^{-1} \mid r \in r_3^{-1} \circ r_2^{-1} \circ r_1^{-1}\ \}$$
$$(r_1 \circ r_2 \circ \ldots \circ r_n)^{-1} = \{\ r^{-1} \mid r \in r_n^{-1} \circ \ldots \circ r_2^{-1} \circ r_1^{-1}\ \}$$

4 N-DIMENSIONAL RELATIONS

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 $\mathbb{P}\ R1$). 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, $\mathbb{P}\ R1$, 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:

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

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:

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

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 (i.e., the *TableLookUp* algorithm), depending on which dimension of objects the algorithm is computing.

5 CONCLUSIONS

In this paper, we analyze temporal interval relations and propose four domains for relation composition. We provide a set of algorithms for the automatic generation of multimedia presentation from temporal and spatial relations among multimedia resources. Possible conflicts of relations are eliminated. We then extend the algorithms to compute relations of objects in an arbitrary n-dimensional space. The algorithms are used in a system that uses an ICON programming technique for multimedia presentation designs.

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 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.

Acknowledgement

I would like to thank all members of the MINE Lab, especially Mr. Anthony Y. Chang for his many helpful discussions, and Mr. F. Y. Jeng, Mr. Steven K. Lo, Mr. Szu-Jan Fu, and Mr. Julian B. Chang for their implementation of the presentation system.

References

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

[2] Chi-Ming Chung, Timothy K. Shih, Jiung-Yao Huang, Ying-Hong Wang, and Tsu-Feng Kuo, "An Object-Oriented Approach and System for Intelligent Multimedia Presentation Designs," in proceeding of the International Conference on Multimedia Computing and Systems (ICMCS'95), Washington DC, U.S.A., May 15 18, 1995, pp 278 – 281.

[3] Young Francis Day, et. al., "Spatio-Temporal Modeling of Viedo Data for On-Line Object-Oriented Query Processing," in proceedings of the International Conference on Multimedia Computing and Systems, Washington DC, U.S.A., May 15 – 18, 1995, pp 98 – 105.

[4] Cherif Keramane and Andrzej Duda, "Interval Expressions - a Functional Model for Interactive Dynamic Multimedia Presentations," in proceedings of the 1996 International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 17 – 23, 1996, pp 283 – 286.

[5] John Z. Li, M. TamerOzsu, and Duane Szafron, "Spatial Reasoning Rules in Multimedia Management Systems," in proceedings of the 1996 Multimedia Modeling international conference (MMM'96), Toulouse, France, Novemner 12 – 15, 1996, pp 119 – 133.

[6] Thomas D. C. Little and Arif Ghafoor, "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks," IEEE Computer, October 1991, pp 42 – 50.

[7] 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.

[8] 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.

[9] 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.

[10] 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.