Consistency Checking for Interactive Multimedia Presentations

Susan Elias
Department of Computer Science
and Engineering
Sri Venkateswara College of Engineering
Chennai - 602 105, India.
susan@svce.ac.in

K. S. Easwarakumar
Department of Computer Science
and Engineering
Anna University
Chennai - 600 025, India.
easwara@cs.annauniv.edu

Lisa Mathew
Department of Mathematics
Vellore Institute of Technology
Vellore - 632 014, India.
lisa_mat@rediffmail.com

Richard Chbeir
Computer Science Department
LE2I - Bourgogne University
BP 47870 21078, Dijon CEDEX France.
rchbeir@u-bourgogne.fr

Abstract

Multimedia presentations are generally specified in terms of spatial and temporal relations between the media objects involved. Creation of these presentations, and interactions with them during their play-out, require an effective mechanism for handling the specifications dynamically. Further, these specifications could result in inconsistencies, which need to be checked and resolved. Our main contribution is the design of an algorithm which aids in resolving these consistency issues dynamically and efficiently. The other salient features of our approach are a new composite spatio-temporal operator and an effective relaxation policy. Thus, this paper presents an interactive multimedia presentation mechanism, which maintains a consistent and complete set of constraints during authoring and play-out of the presentation.

1 Introduction

Interactive presentations that involve multimedia objects are used in diverse fields including education, entertainment and advertising. The creation of an effective presentation requires imagination and skill. These presentations involve explicit spatial and temporal relations which are specified in the form of constraints while authoring. However, some of these constraints could contradict each other and hence lead to inconsistencies. An example is given in the next section to show how inconsistencies arise while defining constraints. When the number of objects and / or constraints is large, the author of the presentation may not be aware of the inconsistencies while creating the presentation. Hence these inconsistencies have to be identified and removed by the presentation process prior to the play-out. Consistency checking issues are also discussed in [1, 2, 3] , an analysis of the consistency models for distributed interactive multimedia applications is presented in [4], and in [5] a mechanism has been proposed to resolve inconsistencies dynamically.

In this paper, we have enhanced the dynamic consistency checking approach proposed in [5]. The relaxation policy employed by our approach uses the classification of relations introduced in [6] to effectively resolve inconsistency. As a result we have achieved a considerable reduction in time complexity. Mechanisms for creation and interaction with multimedia presentations have been studied extensively. While [7] deals with interaction in terms of navigation of the presentation, we focus on the interaction with the presentation, as discussed in the reference model presented in [8]. Moreover, the composition of the spatial and temporal relations as discussed in [9] has been modeled using a spatio-temporal operator and this helps in the effective generation of the presentation layout. Our proposed approach would allow the user to change the spatial and temporal layout i.e it permits interaction with the presentation during authoring as well as during play-out, while still effectively resolving spatio-temporal consistency issues.

The rest of the paper is organised as follows. In section 2, we illustrate the concepts of consistency and completeness of constraints. Section 3 provides an overview of the related work in the area of multimedia constraint checking. Section 4 is devoted to our proposed method for checking consistency and completeness of multimedia constraints. The last section summarises the paper and provides an overview of our future work.

2 Motivation

Multimedia presentations require the specification of the exact location in time and space for each participating media object prior to its play-out. However, instead of specifying the actual starting and ending time, and spatial locations of each object, it would be convenient to specify the relationship between them. For instance, consider two media objects *A* and *B*. Temporal and Spatial relations between *A* and *B* may be given in the form of constraints such as:

T_1*: B starts 5 seconds after A.*

S_1*: The left top corner of A is 50 pixels above and 20 pixels to the left of that of B.*

Consider a new object *C* in addition to *A* and *B* specified earlier. Now, we add two new temporal constraints:

T_2*: C starts 10 seconds after B* and

T_3*: A and C start together*.

As per the temporal constraints T_1 and T_2, *C* should start 15 seconds after *A*, in contradiction to T_3, which says that they start together. Obviously, all three constraints cannot be satisfied at the same time. This situation is referred to as a *temporal inconsistency*. In order to avoid such a situation, the constraints have to be checked thoroughly and one of the constraints (for instance T_3) has to be removed in order to ensure consistency prior to the play out. Similarly, spatial inconsistencies may appear and need to be checked. Suppose in addition to the three objects *A, B* and *C* specified earlier, we add two more media objects *D* and *E* and one more constraint:

T_4*:D starts 5 seconds after E.*

At this point, we do not know anything about the relationship of *D* or *E* with any of the existing media objects *A, B* or *C*. This leads to a discontinuity in the presentation. Hence, we need to specify at least one more constraint T_5 to remove this discontinuity and ensure *completeness* of the set of constraints.

In case of an inconsistency occurring among a group of constraints, one of them may need to be dropped to obtain a consistent set. This is usually done by attaching a *priority value* to each constraint. An edge with a lower priority value is normally considered to be more important than one with a higher value . Thus an edge with priority value 1 has a higher priority than an edge with a priority value 2. The assignment of the priority may be done either by the user, or by the algorithm under the influence of a policy which may be designed to suit the application. Normally, a constraint with the least priority is dropped to resolve inconsistency. However, in some cases especially when the choice is between two or more constraints of the same priority a *relaxation policy* has to be employed to choose one of them. Although this discussion deals with consistency checking of temporal constraints only, it is also applicable to any other type of relation between two media objects, which could be modeled as a constraint.

3 Related Work

3.1 Consistency Checking

A formal approach toward consistency checking of interactive multimedia documents was dealt with in [2], using the formal description technique RT-LOTOS. Here the high level specification of the multimedia documents is automatically translated into the RT-LOTOS specification, and from this a minimal reachability graph is generated. Consistency analysis is then performed on this graph. If the document is found to be consistent the scheduling graph is generated.

In [3], an approach for temporal consistency checking based on the transformation of scenario specification into networks, called constraint network fragments (cnfs) is presented. The rules for building these constraint networks, corresponding to the fundamental functional units of the scenario are defined, and along with temporal constraint verification techniques provide the basis for checking the consistency of the scenario.

A graph-based method was proposed in [1] for analyzing multimedia constraints for consistency and completeness. While authoring a multimedia presentation, the synchronization specifications were modeled as a set of constraints. These constraints were then used to generate a directed graph known as the temporal consistency graph. The vertices of this graph represented the media objects while the edges represented relations between them. For instance, in the earlier example the vertex set was $\{A,B,C,D,E\}$, while the edges were $\{T_1,T_2,T_3,T_4,T_5\}$.

The connectivity of the underlying undirected graph was used to check for completeness. Since every cycle in this graph represented a possible inconsistency, the paper focused on finding a acyclic graph connecting all vertices (i.e. a spanning tree). In order to find the best set of constraints the priorities attached to the constraints were used as weights for finding a minimal spanning tree using Kruskal's algorithm [10]. The constraints that were left out by the Kruskal's algorithm were examined once again for possible inclusion with the help of an appropriate relaxation policy. Thus their work consisted of five phases for dealing with the temporal constraints visualization - preprocessing, completeness checking, construction of minimal spanning tree, removing inconsistencies using a relaxation policy and generation of the temporal layout. This was followed by a similar discussion on dealing with spatial constraints.

The temporal layout generated for a given specification gave rise to a number of presentation intervals. Within each of these intervals the spatial consistency had to be checked independently. The method for checking spatial consistency was similar to that used for temporal consistency checking with the same five phases repeated. Although the paper also talks about interactive authoring, it does not allow for dynamic consistency checking, which would have permitted

the algorithm to respond to inconsistencies while the presentation is being created.

3.2 Dynamic Consistency Checking

In the method discussed above, consistency checking could be performed only after the entire set of constraints were available. This was improved upon [5] and was made dynamic by examining each constraint immediately on input. For the implementation of this method, the functionally complete temporal operator given below was proposed:
TEMPORAL(A, B, $d_1, d_2, priority$*)*
where $d_1 = b_B - b_A$ and $d_2 = t_B - t_A$. Here b_A, t_A and b_B, t_B are the beginning and end times of the media objects *A* and *B* respectively and *priority* is the priority value assigned to the constraint.This operator was used to specify the temporal relations as a set of constraints. The consistency checking algorithm in [1] was made more efficient by eliminating some of the redundant steps. The new algorithm was designed to examine each constraint immediately on input for possible inconsistency with those already in the tree. This process helps to construct the spanning tree dynamically.

The major difference between [1] and [5] is that preprocessing was eliminated totally in [5] and the rest of the phases were combined and performed simultaneously, thereby reducing the total running time of the algorithm. As a result consistency checking could commence immediately on input of the first constraint unlike [1] were the entire set of constraints needs to be input before the procedure could begin. In case a cycle was identified while checking for possible inclusion in a spanning tree, it was recognised as an inconsistency. Then one of the constraints was dropped, using an appropriate relaxation policy. Finally, when all the constraints had been input, the resulting graph was examined for completeness and the user was prompted to specify an appropriate constraint to resolve any possible incompleteness. The functionally complete spatial operator given below, was also proposed in [5] to capture the spatial constraints effectively and is given below:
SPATIAL(A, B, $d_l, d_r, d_b, d_u, d_z, priority$*)*
where $d_l = x_{l_B} - x_{l_A}, d_r = x_{r_B} - x_{r_A}$,
$d_b = y_{d_B} - y_{d_A}, d_u = y_{u_B} - y_{u_A}, d_z = z_B - z_A$ and
$(x_{l_A}, x_{r_A}, y_{d_A}, y_{u_A})$ and $(x_{l_B}, x_{r_B}, y_{d_B}, y_{u_B})$ are the minimum bounding boxes of two objects *A* and *B* respectively, while z_A and z_B represent their depth information and *priority* is the priority value assigned to the constraint. The spatial consistency checking within each temporal interval was dealt with in a manner similar to their respective temporal consistency checking mechanisms.

We illustrate the algorithm with the help of our earlier example. Consider a multimedia presentation involving 5 multimedia objects *A, B C, D* and *E* and 5 temporal constraints as follows:
T_1*:TEMPORAL(A,B,5,10,1)*
T_2*:TEMPORAL(B,C,10,0,1)* , T_3*:TEMPORAL(A,C,0,10,2)*
T_4*:TEMPORAL(A,B,3,5,3)* , T_5*:TEMPORAL(D,E,5,10,3)*
For this presentation we assume that the object A starts at time 0 and its length is 30 seconds.The spanning tree created is shown in Figure 1(a) and 1(b) and the temporal layout in 1(c). The algorithm works as follows:

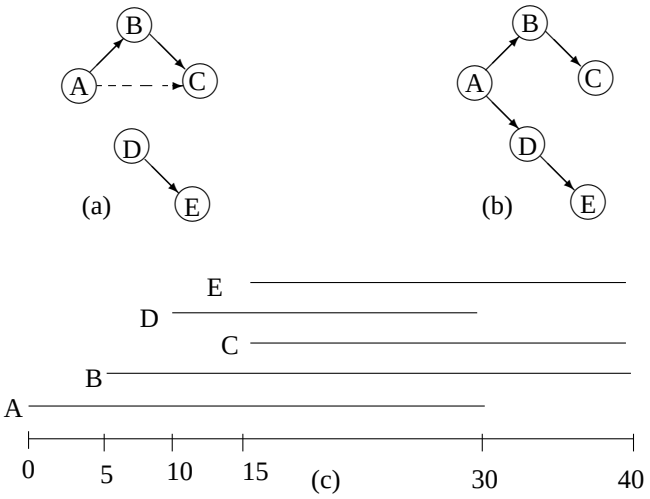

Figure 1. Presentation Layout

Since T_1 and T_2 do not give rise any inconsistency they are included in the spanning tree. Since T_3 is inconsistent with the above constraints it is dropped since it has a lower priority than T_1 and T_2. Again T_4 is dropped since it is inconsistent with T_1. Next T_5 is included. Since all the constraints specified have been checked the algorithm now checks for completeness and prompts the user to specify a new constraint T_6 to generate a complete set of constraints. Suppose the user specifies the new constraint as T_6*:TEMPORAL(A,D,10,10,1)*. At this stage the algorithm finds that the tree is complete and hence it generates the temporal layout. A similar algorithm was used to check the spatial constraints between the media objects present in each interval for consistency.

3.3 Limitations of the existing mechanism

- In [5] the temporal and spatial constraints were dealt with separately. After the temporal layout was generated each presentation interval was examined and the spatial constraints between the objects appearing in them were examined for consistency. Although this approach allowed dynamism in dealing with temporal constraints, the spatial constraints could be dealt with only after all the temporal constraints had been processed. Hence dynamism could not be extended to the spatial constraints. For instance, the temporal layout in Figure 1(c) has five presentation intervals (a presentation interval represents an interval of time during which the participating media objects remain the same). Now, the algorithm accepts spatial constraints for each of the five intervals and generates spanning trees for each of them. Thus, for the above illustration the algorithm would require the generation of six (one for temporal constraints and five for spatial con-

straints) spanning trees and their respective layouts to generate a consistent presentation schedule.

- For presentations with rapid changes in media objects, the existing approaches would not be efficient. Moreover, spatial specifications between pairs of media objects appearing in more than one interval need to be repeated in each such interval. Also incompleteness in each of the intervals could result in the algorithm prompting the user for more constraints. Although this method could be extended to support interactions with the presentation during play-out it is seen that each time a temporal constraint is modified by the user all the spanning trees for the spatial constraints may have to be regenerated. As a result, this approach would be inefficient in terms of the response time during interaction.

In this paper, we have overcome these limitations by designing an integrated operator for representing both the temporal and spatial relations. As a result the entire process would require just one spanning tree regardless of the number of presentation intervals generated. This leads to the development of a consistency checking mechanism which is highly responsive to interactions with the presentation during play-out and which achieves a considerable reduction in the time complexity in comparison with [1] and also with [5].

4 Proposed Approach

In order to overcome the drawbacks in [1] and [5], spatial and temporal relations between two objects need to be combined and represented using the same binary relation. For this purpose we introduce a new composite spatio-temporal operator ST, which helps to capture all aspects of the relationship between two media objects in an effective way. It is defined as

ST(A, B, $d_{t_1}, d_{t_2}, d_{t_3}, d_{t_4}, d_l, d_r, d_b, d_u, d_z$*, flag, priority)*

where $d_{t_1} = b_B - b_A$, $d_{t_2} = t_B - t_A$, $d_{t_3} = b_B - t_A$, $d_{t_4} = t_B - b_A$, $d_l = x_{l_B} - x_{l_A}, d_r = x_{r_B} - x_{r_A}$, $d_b = y_{d_B} - y_{d_A}, d_u = y_{u_B} - y_{u_A}, d_z = z_B - z_A$

The flag in the operator is used to indicate, the presence or absence of the spatial relationship between *A* and *B*, using 1 or 0 respectively. The absence of a spatial relation occurs when either *A* or *B* or both of them are audio elements. The priority is assigned by the algorithm and is used by the relaxation policy to resolve inconsistency.

4.1 Algorithm for consistency checking of spatio-temporal constraints

The composite operator defined above enables the algorithm to generate a consistent set of constraints by building a single spanning tree for the entire presentation. Thus temporal and spatial constraints are checked simultaneously . In algorithm 1, *V* represents the set of nodes of the generated spanning tree *T*, while *E* represents the set of its edges. We make use of the the following subroutines: *MAKE-SET(i)*: creates a new set with representative *i*. The only member of this new set is the element *i*.(A representative of a set is any one element chosen from the set to uniquely identify the set. This element could be chosen using any arbitrary rule. Here for instance, we just consider the first element in the lexicographic ordering of the elements of the set, to be the representative).*FIND-SET(i)*: returns a pointer to the representative of the set containing *i*.*UNION(i,j)*: merges the two sets corresponding to the elements *i* and *j* and assigns the representative of one of the two sets as the representative of the new set. *LIST(S)*: enumerates the elements of set S. *RELAX*(path): implements the relaxation policy. *DFS(S,i,j)*: performs a Depth-First-Search and returns a path that starts at *j* and ends at *i*. Here, it is also used to keep track of the edge with the largest priority value encountered.*BFS(T)*: performs a Breadth-First-Search of the spanning tree.In the following, we explain the three steps of our algorithm:

Algorithm 1: Integrated_ConCheck()

Input: E = NULL, V = NULL
Output: T, I

1 **while** *user generates constraints* **do**
2 ST (i, j, $d_{t_1}, d_{t_2}, d_{t_3}, d_{t_4}, d_l, d_r, d_b, d_u, d_z$, flag, priority) ⇐ constraint
3 **if** *(i ∉ V)* **then**
4 move i to V
5 do MAKE-SET(i)
6 **end**
7 **if** *(j ∉V)* **then**
8 move j to V
9 do MAKE-SET(j)
10 **end**
11 **if** *(FIND-SET(i)≠ FIND-SET(j))* **then**
12 E=E ∪ e(i,j)
13 UNION(i,j)
14 **end**
15 **else**
16 S = FIND-SET(j)
17 path= DFS(S,i,j)
18 RELAX(path)
19 **end**
20 **end**
21 R=V
22 i=1
23 **while** *R≠ NULL* **do**
24 S= FIND-SET(i)
25 R=R-LIST(S)
26 Choose $j \in R$
27 input constraint e(i,j)
28 E= $E \cup e(i,j)$
29 $i = j$
30 **end**
31 Input dimensions of the first object i
32 T = (V,E); path = BFS(T) /*Starting at i*/
33 **for** *each e(i,j) ∈ path* **do**
34 determine length of j /* where i is the object whose dimensions are known*/
35 schedule the begin and the end events for j
36 **end**
37 Sort all events in temporal order
38 Generate the spatio-temporal layout

1. **Consistency checking**: is done in steps 1 to 20. The spanning tree is created as follows: A collection of disjoint sets of vertices is initialized by invoking the subroutine *MAKE-SET* each time a new vertex appears. In order to check whether there is a path in *T* between two vertices *i* and *j*, we use the subroutine *FIND-SET* on each of them. If the the sets

containing vertices *i* and *j* have the same representative , it indicates that *i* and *j* belong to the same set and also that there exists a path between them. Hence in case at some point of time, while processing an edge *e(i,j)* we find that *i* and *j* are not connected by any path (i.e they do not have the same representatives) then, we can safely add the edge to get a new graph which is still acyclic. However, once this is done, we should merge the sets containing these two vertices so that any future invocation of *FIND-SET* will return the same representative for both sets. This is done by using *UNION(i,j)*. On the other hand, if *i* and *j* are already connected by a path, *FIND-SET(i)* in co-ordination with *LIST* and *DFS* helps to identify the path which along with the newly input edge forms a cycle. In order to ensure consistency and maintain the acyclic nature of T, one of the edges on this path has to be dropped. The choice of which edge is to be dropped is made by using *RELAX*(path) which implements the relaxation policy discussed in the next section. The *DFS* subroutine also keeps track of the edge with the lowest priority encountered.

2. **Completeness checking of constraints**: is done in steps 21 to 30 using *FIND-SET* and *LIST* to find all the vertices in the set containing the first media object. If this generates the whole of V, the set of constraints is complete indicating that the media objects that belong to the presentation are directly or indirectly related to the first media object. Otherwise, the algorithm prompts the user to supply an appropriate constraint to make the set of constraints complete.
3. **Temporal layout generation**: is handled by steps 31 to 38. The user is prompted to give the length of the first media object. A Breadth First Search (*BFS*) beginning at this vertex is performed . The *start* and *end* events for each node in the path returned by *BFS* is next generated. These events are sorted and temporal layout of the entire presentation is formulated from this sorted list of events.

4.1.1 Relaxation policy

In [6], the authors proposed a meta-model which was used to classify relations. This model helped spatial and temporal relations to have more expressive power. For instance, the authors have identified the existence of 33 temporal relations between two intervals instead of 13 that are traditionally used by several applications. On the basis of the proposal provided in [6], a relaxation policy was designed to resolve inconsistencies. In order to classify the temporal relations as given below a threshold value is chosen.

Class 1. Begins together
Ends together
Class 2. Begins / Ends just before the beginning / ending
Begins / Ends just after the beginning / ending
Class 3. Begins / Ends before the beginning / ending
Begins / Ends after the beginning / ending

Combinations of Class 1 are assigned priority 1
Combinations of Class 1 and 2 are assigned priority 2
Combinations of Class 1 and 3 are assigned priority 3
Combinations of Class 2 and are assigned priority 4
Combinations of Class 2 and 3 are assigned priority 5
Combinations of Class 3 are assigned priority 6

Given a constraint the values assigned to $d_{t_1}, d_{t_2}, d_{t_3}, d_{t_4}$ are checked to determine which class they belong to. Then the constraint is assigned a priority as given below.For instance, consider the threshold value to be 5 secs. Then a relation is assigned priority 1 if two of these values are zero (i.e begin and end together). If one of these values is zero and one of the others is less than the threshold , then it is assigned a priority 2. In a similar manner, priorities are assigned in other cases. The edge in consideration is dropped when its priority happens to be equal to or less than the value detected in the path by DFS. Thus the relaxation policy which is employed when a cycle is detected in the spanning tree, has been designed to retain temporal relations that are closer (defined by the threshold) by assigning them higher priority. In case the priority of the edge under consideration is higher than that detected in the path this edge is inserted into the tree and the constraint in the path having the lowest priority is removed to avoid a cycle in the spanning tree, thus resolving inconsistency.

4.1.2 Illustration

We consider again the example given in [1]. The multimedia scenario of 40s first presents a 10s logo consisting of a Logo-Animation(*A*) and Logo-Music(*B*). Then 20s of Audio1(*C*) is played with Video1(*D*). Text(*E*) will be displayed while playing *D*. The presentation ends with 10s of Exit-Animation(*F*) along with an Exit-Music(*G*). A background (*H*) and a caption (*I*) will be present throughout the entire presentation. Assume the screen resolution to be 512 x 512 with the origin at the lower left corner. The background object is assumed to be the source of the presentation, occupying the entire screen having 0 as its z value and commencing at time zero. The above scenario can be specified in the form of the following constraints. The priority that would be assigned by the algorithm is also indicated.

ST1 = ST(A,B,0,0,-10,10,0,0,0,0,0,0,1)
ST2 = ST(D,E,4,0,-16,20,156,-156,456,-16,0,1,2)
ST3 = ST(C,D,0,0,-20,20,0,0,0,0,0,0,1)
ST4 = ST(C,G,20,10,0,30,0,0,0,0,0,0,3)
ST5 = ST(G,F,0,0,-10,10,0,0,0,0,0,0,1)
ST6 = ST(C,E,0,-4,-30,16,0,0,0,0,0,0,2)
ST7 = ST(D,F,15,5,0,30,56,-56,56,-56,0,1,2)
ST8 = ST(H,A,0,-30,-40,10,56,-56,56,-56,20,1,3)
ST9 = ST(A,I,0,30,-10,40,50,-50,-40,-400,-10,1,3)
ST10 = ST(H,D,10,-10,-30,30,0,0,0,0,20,1,6)
ST11 = ST(D,I,-10,10,-30,30,106,-106,16,-456,-10,1,6)
ST12 = ST(H,E,14,-10,-26,30,156,-156,436,-36,20,1,6)
ST13 = ST(H,F,30,0,-10,40,56,-56,56,-56,20,1,3)
ST14 = ST(F,I,-30,0,-40,10,50,-50,-40,-400,-10,1,3)

In the above example, the relaxation policy would be employed by the algorithm six times each time dropping a constraint that is deemed least important. The output spanning tree is given in Figure 2(c) The structure of the tree after the input of *ST6* is shown in 2(a) and after *ST7* in 2(b)

4.2 Support for interactive authoring and playout

The specification set of the presentation may require addition and deletion of objects and constraints. Addition is handled in the

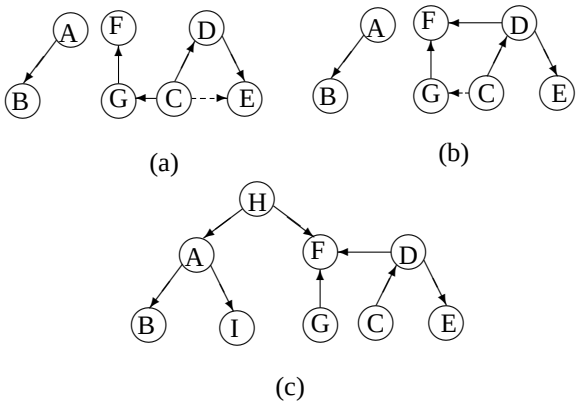

Figure 2. Construction of the Spanning Tree

same way as the algorithm *Integrated_ConCheck* handles any new object or constraint during its execution. Updating of constraints i.e. changing the temporal and spatial values specified in the ST operator can also be permitted. But this would require that appropriate changes be made to all other related constraints followed by a regeneration of the layout as in steps 31 to 38. Deletion of a constraint would require the algorithm to check again for completeness and then generate the layout as given in steps 21 to 38. However, deletion of an object requires in addition the deletion of all the constraints associated with that object.

Normally the presentations are generated from the layouts designed. Navigation of the authored presentation can be easily supported in our approach as in [7]. Interaction with the presentation (i.e. making permitted alteration to the temporal and spatial layouts) can also be effectively supported by our method. This is done by ensuring that the spanning tree built during authoring is made available by regenerating it in the background while the presentation is being played. The interactions with the presentation are captured again as constraints and dealt with appropriately for addition, deletion and updation as discussed above.

4.3 Complexity

Consider the algorithm *Integrated_ConCheck*. Assume that the presentation has m constraints and deals with a total of n objects. The algorithm has n *MAKE-SET* operations requiring time *O(n)*, m *FIND-SET* operations requiring time *O(m)*, n-1 *UNION* operations requiring time *O($m\alpha(m,n)$)* [11], m-n+1 *DFS* operations requiring requiring time *O(n(m-n))* and one sort operation requiring time *O(nlogn)*. Hence the total time complexity of the algorithm is *O(mn)* [12].

5 Conclusion

This paper introduces an operator to model spatio-temporal relations to be used as constraints in a multimedia presentation specification. The use of the operator in addition to the simplification of the algorithm, has also helped to achieve a constant time reduction in the time complexity while simultaneously being highly responsive to interactions with the presentation during play-out. We have thus presented an approach for consistency checking of spatio-temporal relations and an effective relaxation policy which supports interactive authoring and play-out. Our future work would include the use of a formal approach for consistency checking and support for interaction to enhance the capabilities of the method to support complex distributed multimedia applications and Quality-of-Presentation guarantees.

References

[1] H. Ma and K. G. Shin, "Checking consistency in multimedia synchronization constraints," *IEEE Trans. Multimedia*, vol. 6, pp. 565–574, Aug. 2004.

[2] P. N. M. Sampaio and J. P. Courtiat, "A formal approach for the presentation of interactive multimedia documents," in *Proceedings of the eighth ACM international conference on Multimedia*, Marina del Rey, California, United States, Oct. 2000, pp. 435 – 438.

[3] I. Mirbel, P. Pernici, and M. Vazirgiannis, "Temporal integrity constraints in interactive multimedia documents," in *Proceedings ACM Multimedia'93*, Anaheim, CA, Aug. 1993, pp. 341–350.

[4] N. Bouillot and E. Gressier-Soudan, "Consistency models for distributed interactive multimedia applications," *ACM SIGOPS Operating Systems Review archive*, vol. 38, no. 4, pp. 20–32, 2004.

[5] S. Elias, K. S. Easwarakumar, L. Mathew, and R.Chbeir, "Dynamic consistency checking for spatio-temporal relations," in *Submitted to The 21st ACM Symposium on Aplied Computing*, Dijon,France, 2006.

[6] R. Chbeir, Y. Amghar, and A. Flory, "High expressive spatio-temporal relations," in *Special Track On Spatiotemporal Reasoning in the 15th International FLAIRS Conference*, Pensacola, Florida, May 2002.

[7] C.-M. Huang and C. Wang, "Synchronization for interactive multimedia presentations." *IEEE Multimedia*, pp. 44–61, Oct. 1998.

[8] B. Rogge, J. Bekaert, and R. V. de Walle, "Timing issues in multimedia formats: Review of the principles and comparison of existing formats," *IEEE Trans. Multimedia*, vol. 6, pp. 910–924, Dec. 2004.

[9] M. Vazirgiannis, Y. Theodoridis, and T. Sellis, "Spatio-temporal composition and indexing for large multimedia applications," *ACM/Springer-Verlag Multimedia Systems Journal*, vol. 6, pp. 284–298, July 1998.

[10] J. B. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem," *Proceedings of The American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[11] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM*, vol. 22, no. 2, pp. 215–225, 1975.

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