# An Approach to Checking Consistency in Multimedia Synchronization Constraints

Huadong Ma[a,b] and Kang G. Shin[b]
[a] College of Computer Sci.& Tech., Beijing University of Posts & Telecomm., Beijing 100876, China
[b] RTCL, EECS Dept., The University of Michigan, Ann Arbor, MI 48109-2122, USA

*Abstract*—**This paper proposes a new method for analyzing multimedia synchronization constraints based on the constraint graph and classification, which is essential in developing efficient support tools for constraint-based authoring systems. We specify the temporal relations between multimedia objects, and use a directed graph to represent the constraints among the objects in a multimedia scenario. We then analyze synchronization constraints based on graph theory, solving multiple problems in a unified theoretical framework: completeness and/or consistency checking, constraints relaxation and automatic temporal layout generation. We also discuss the effects of user interactive authoring. Compared to the other methods, it is simpler, more efficient, and easier to implement.**

## I. Introduction

One of the peculiar characteristics of multimedia data is that they are tightly-coupled through both spatial and temporal constraints. Thus, a key problem in multimedia systems is how to model spatial and temporal synchronization. The temporal relation is considered as the master relation in a multimedia scenario, and the spatial relation is a slave to the temporal relation. So we focus on the temporal synchronization.

The temporal relations can be specified by two basic models: time-line and constraint-based models. Constraint-based models have a definite advantage over the traditional time-line approach due mainly to their flexibility. Although the constraint-based authoring systems have several distinct advantages, they must provide a mechanism for users to easily manipulate the underlying structures so as to implement completeness and consistency checking, constraints relaxation and layout generation.

The temporal synchronization was addressed in [3,4]. Temporal-synchronization models can be classified as a graph model [5,7], Petri-net based model [9], object-oriented model [8], or language-based model [2,6,10]. In a graph model, constraints are represented as edges in a graph, whereas a Petri-net model captures the temporal constraints by means of places and transitions. In an object-oriented model the constraints are modeled as object attributes, while a language-based model uses programming language constructs to express constraints.

An important issue, which has not yet been fully investigated, is consistency checking. Especially for multi-user collaborative authoring of a multimedia scenario, there usually exist inconsistencies in constraints specified by different users. In spite of its importance to the production of a correct presentation, only a few methods, such as Firefly [5], TCSP [2] and Elastic Time Model [7], explicitly provide such mechanisms. Another important issue is how to deal with the case when the satisfaction of synchronization constraints cannot be guaranteed, i.e., when an inconsistency is found. In such a case, it may be useful to provide a strategy to relax the constraints so as to obtain a feasible presentation. The simplest way is to explicitly require the insertion or the deletion of some constraints from the specification. Alternatively, constraints can be divided into mandatory and optional parts. Or one might provide some form of constraint relaxation by shrinking or stretching the duration of each object when a presentation is infeasible.

To our best knowledge, there has been no report in literature on checking for the incompleteness of synchronization constraints, hence no unique layout of presentation. We propose a new synchronization constraint analysis model, which differs from Firefly and Elastic Time models, for completeness and consistency checking, constraint relaxation and layout generation (either interactively or automatically). Main contributions of this paper are summarized as follows. We first propose the concept of constraint completeness and divide the constraints into different categories, then introduce the priority number to identify them so as to take different policies in removing inconsistencies, and use a directed graph to express the constraints among multimedia objects. Our algorithms improve efficiency in checking consistency with a simpler model. Moreover, we solve the above problems in a unified theoretical framework.

## II. Temporal Synchronizations Constraints

### A. Temporal relations

Assume that object A and B are scheduled in the interval $[b_A, t_A]$, $[b_B, t_B]$, respectively. As shown in Table 1, there are 13 possible temporal relations between A and B, that is, *equals, before, meets, starts, finishes, during, overlaps* and their inverse relations [1].

Table 1. Interval relations

| Relations | Symbols | Reverse | Definitions |
|---|---|---|---|
| A equals B | $e(A,B)$ | $e(B,A)$ | $b_A = b_B < t_A = t_B$ |
| A before B | $b(A,B)$ | $b^{-1}(B,A)$ | $b_A < t_A < b_B < t_B$ |
| A meets B | $m(A,B)$ | $m^{-1}(B,A)$ | $b_A < t_A = b_B < t_B$ |
| A starts B | $s(A,B)$ | $s^{-1}(B,A)$ | $b_A = b_B < t_A < t_B$ |
| A finishes B | $f(A,B)$ | $f^{-1}(B,A)$ | $b_B < b_A < t_A = t_B$ |
| A during B | $d(A,B)$ | $d^{-1}(B,A)$ | $b_B < b_A < t_A < t_B$ |
| A overlaps B | $o(A,B)$ | $o^{-1}(B,A)$ | $b_A < b_B < t_A < t_B$ |

In order to simplify the representation of temporal relations, we merge some of them, and introduce new notations as follows.

$Cov(A,B,d1,d2)(d1,d2 \geq 0)$ means the presentation interval of B is covered by that of A. It is used to specify concurrent presentations of objects instead of *e,s,f,d* .

$Ovl(A,B,d1,d2)(d1,d2 > 0)$ means there is a partial overlap in presenting A and B. It is used to specify cross presentations of objects in the same way as $o$.

$Seq(A,B,d)(d \geq 0)$ means the presentation of B is after that of A with delay $d$. It is used to specify sequential presentations of objects instead of *m,b*.

The work is supported by the National Natural Science Foundation of China, UNU/IIST Off-shore Project and US NSF under Grant EIA-9806280. The first author is a visiting scholar with RTCL,University of Michigan under the support of CSC. Correspondence email: {mahd,kgshin}@eecs.umich.edu.

Table 2 summarizes the definitions of three relations.

Table 2. New interval relations

| Relations | Symbols | Definitions |
|---|---|---|
| Sequential | *Seq(A,B,d)* | $b_A < t_A \leq b_B < t_B$ <br> $d = b_B - t_A$ |
| Cover | *Cov(A,B,d1,d2)* | $b_A \leq b_B < t_B \leq t_A$ <br> $d1 = b_B - b_A, d2 = t_A - t_B$ |
| Overlap | *Ovl(A,B,d1,d2)* | $b_A < b_B < t_A < t_B$ <br> $d1 = b_B - b_A, d2 = t_B - t_A$ |

**Theorem 1.** The relation set consisting of *Cov, Ovl* and *Seq* is a complete set of temporal relations.

We can easily prove this theorem by representing all of 13 temporal relations using *Cov,Ovl* and *Seq*.

In our approach, they are used as temporal synchronization constraints illustrated in Figure 1.

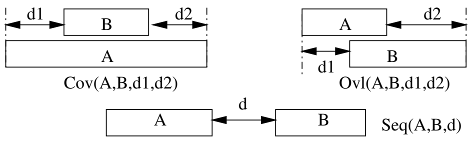


Figure 1. The diagram of temporal constraints

Authoring a multimedia scenario with a constraint-based system, a user must distribute the constraints among the media objects. Actually, he may have preference to some constraints, and in such a case, the system should assure them to be satisfied. For the convenience in removing the inconsistencies in constraints, we divide the constraints into three categories and give them different priority numbers. For instance, 1-guaranteed, 2-negotiable, 3-negligible; 1 means higher priority than 2 and 3.

**Example 1**: Consider a multimedia scenario which first presents a $10s$ logo consisting of Log-Ani (LA) and Logo-Music (LM) and then plays $20s$ Audio A1 with Video V1; Text T1 will be displayed while playing V1; at end, it plays $10s$ Exit-Music (EM) with Exit-Ani (EA), V1 may overlap with EM. Seven constraints are used to specify the scenario: $R_1$=*Cov*(LA,LM,0,0); $R_2$=*Cov*(V1,T1,4,0); $R_3$=*Cov*(A1,V1,0,0); $R_4$=*Seq*(A1, EM, 0); $R_5$=*Cov*(EM, EA, 0, 2); $R_6$=*Cov*(A1, T1, 0, 4); $R_7$=*Ovl*(V1, EA, 15, 5). We assign the priority number 1 to $R_1, R_3, R_5$; 2 to $R_2, R_4, R_6$; and 3 to $R_7$.

### B. The constraint analysis model

**Definition 1.** The Temporal Constraints Graph (TCG) is defined as a three tuple TCG=(*V,E,W*), where *V*=$\{O_1, \ldots, O_N\}$ is the set of vertices representing *N* media objects, $E \subset V \times V$ is the set of edges specifying the temporal constraints between objects. There is a directed edge $e(i,j)$ from $O_i$ to $O_j$ if there is a constraint $C(O_i, O_j)$ between them, where $C$ is one of three primitive constraints. *W:* $E \rightarrow \{1, 2, 3\}$ is a mapping from *E* to integer for defining a priority number for each constraint.

In general, $|E|$ is much smaller than $N^2$, so we assume that an input graph is represented in adjacency lists, in which $e(i,j)$ is stored as an element of the list $Adj[i]$. There may be more than one constraint between a pair of objects, but it obviously causes inconsistency. In this case, a best acceptable constraint is determined by comparing these constraints and reporting the inconsistency to the user. The best one should have the minimum priority number, or the one the user interactively chooses. The data structure for an edge in the list is as follows.

```
struct Edge {
   Integer dvn; /* directed vertex number*/
   Constraint cot; /* constraint type:Cov,Ovl or Seq*/
   Integer d1,d2; /*constraint parameters*/
   Integer pnum; /*priority number*/
   Edge *nedge; } /*pointer to next edge*/
```

Thus the initial TCG will be structured by a preprocessing algorithm **PreTCG** on the above idea. There is at most one constraint for any pair of vertices in a TCG. Figure 2(a) shows the TCG of the scenario in Example 1.

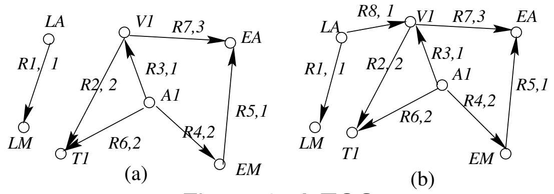


Figure 2. A TCG

## III. Constraint Checking

### A. Completeness Checking

**Definition 2.** The constraints of a multimedia scenario are said to be *complete* if there is a given or implied constraint $C(O_i, O_j)$ or $C(O_j, O_i)$ for any pair of objects $O_i, O_j$.

An undirected graph is *connected* if every pair of vertices are connected. So the concept of an undirected TCG can be used to map the completeness of temporal constraints.

**Definition 3.** If an undirected TCG is connected, it is called a *Complete TCG* (CTCG).

We need to determine if the TCG is a CTCG. If not, the constraints are not enough to fix the temporal relationships among the vertices in TCG. In general, an undirected graph has some connected components. An undirected graph is connected if it has exactly one connected component. We developed an algorithm **JudgeCTCG**, which finds the connected components of TCG. If a TCG has more than one connected component, the constraints are incomplete. Additional constraints should be added the sets of vertices of different components. Actually, the user just needs to add *K-1* constraints if there are $K$ connected components. TCG will become a connected graph after appending constraints. Figure 2(b) is the CTCG of Example 1 by adding $R_8$=*Seq*(LA,V1,0).

### B. Consistency Checking

When checking consistency, a CTCG is considered as a connected, undirected graph. We can prove the following theorem by constructing an acyclic constraint graph (tree).

**Theorem 2.** The layout of presenting $N$ media objects can be fixed by *N-1* constraints among them, where each object appears at least in one constraint.

Inconsistencies are usually due to the redundancy in constraints. If there are more than *N-1* edges in a CTCG representing constraints among $N$ objects, there must exist redundant constraints. In order to find the best *N-1* constraints the user hopes to have, we consider the priority number of each constraint and then find an acyclic subset $T \in E$ that connects all of vertices and whose total weight (priority number) $w(T) = \sum_{c \in T} c.pnum$ is minimized. Since $T$ is acyclic and connects all of vertices, it must be a spanning tree. So, the minimum spanning tree algorithm can be used to solve this problem.

Figure 3(a) gives the minimum priority number spanning tree (MPST) for the CTCG of Example 1. After finding the MPST $T$

of CTCG, we can obtain some constraints that are not selected by *T* if the number of CTCG constraints is greater than *N-1*. These constraints are possibly inconsistent with those in *T*. In order to remove the inconsistencies, we need to determine if the remaining constraints are acceptable. We now describe the consistency checking method and the relaxation strategy.

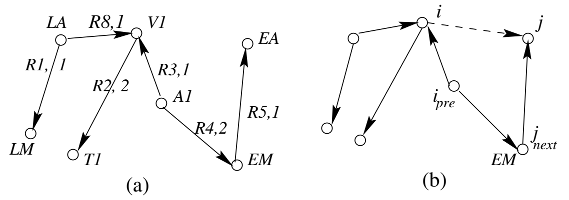


Figure 3. MPST and redundant constraints

At first, take a constraint $C(O_i, O_j) \in E - T$ in nondecreasing order of *pnum*, and join *C* to *T*, then there must be a cycle $\alpha$ shown by Figure 3(b), assume that $\alpha = \{e(i,j), e(j, j_{next}), \cdots, e(i_{pre}, i)\}$, where $e(i,j)$ is produced by *C*.

According to the MPST algorithm, it is impossible that $e(i,j).pnum$ is less than the others' in $\alpha$. Thus, it is no more important than others for the user (Table 3, where $p = Max_{e \in \alpha - \{e(i,j)\}}\{e.pnum\}$). But it is necessary to decide if *C* is acceptable.

Table 3. Possible priority number combinations

| Case | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| *C.pnum* | 1 | 2 | 2 | 3 | 3 | 3 |
| $p$ | 1 | 1 | 2 | 1 | 2 | 3 |

From the constraints in $\alpha$ except *e(i,j)*, we can get the temporal relation between $O_i$ and $O_j$ by using the definitions of constraints repeatedly in the order of cycle, and Table 4 shows the formula we used. Note that we should consider the direction of constraint (edge of directed TCG) to find the correct relation.

Table 4. The constraint relation

| Constraint | Relations between objects |
|---|---|
| $Seq(A, B, d1)$ | $b_B = t_A + d1, t_B = l(B) + t_A + d1$ |
| $Cov(A, B, d1, d2)$ | $b_B = b_A + d1, t_B = t_A - d2$ |
| $Ovl(A, B, d1, d2)$ | $b_B = b_A + d1, t_B = t_A + d2$ |

Thus, the relaxation policy of inconsistent constraints can be decided by the user once they are found. We just give a default policy. Given $dx = b_i - b_j$, $dy = t_i - t_j$, where $b_i, t_i$ are the start time and end time of $O_i$, respectively, there are 6 cases in total, which correspond to 6 different relations between $O_i$ and $O_j$, as shown in Table 5.

Table 5. The possible relations

| Case | Conditions | Implied constraint $F$ |
|---|---|---|
| 1 | $dx > 0, dy > 0, b_i < t_j$ | $Ovl(O_j, O_i, dx, dy)$ |
| 2 | $dx < 0, dy < 0, b_j < t_i$ | $Ovl(O_i, O_j, -dx, -dy)$ |
| 3 | $dx > 0, dy \leq 0$ | $Cov(O_j, O_i, dx, -dy)$ |
| 4 | $dx < 0, dy \geq 0$ | $Cov(O_i, O_j, -dx, dy)$ |
| 5 | $b_i \geq t_j$ | $Seq(O_j, O_i, b_i - t_j)$ |
| 6 | $b_j \geq t_i$ | $Seq(O_i, O_j, b_j - t_i)$ |

Now, we take $C(O_i, O_j)$ to compare with the found relation $F$, then decide how to use the strategies to relax the synchronization constraints in order to obtain a feasible specification.

(1) If their constraint types and directions are same, they are compatible. They can be relaxed as follows:

(1a) if $p$ is the same as *C.pnum* ((*C.pnum,p*)=(1,1), (2,2) or (3,3)), the parameters of $C$ are relaxed as follows: *C.d1=(C.d1+F.d1)/2, C.d2=(C.d2+F.d2)/2*(*if necessary*). $C(O_i, O_j)$ is appended to the CTCG, and remove the constraint with priority number *p* which is the nearest to node *i* or *j* in $\alpha$.

(1b) if *C.pnum* is 3((*C.pnum,p*)=(3,1) or (3,2)), then keep the original constraints, and treat $C(O_i, O_j)$ as an inconsistent constraint, so remove it from the constraint set.

(1c) if (*C.pnum,p*)=(2,1), the parameters of $C$ are relaxed as follows: *C.d1=(C.d1+2F.d1)/3, C.d2=(C.d2+2F.d2)/3*(*if necessary*), $C.pnum = 1$. $C(O_i, O_j)$ is appended to the CTCG, and remove the constraint between $O_j$ and $O_{j_{next}}$.

(2) If their types are same but their directions are different, only *Cov* constraints are compatible, i.e., $C(O_i, O_j) = Cov(O_i, O_j, d1, d2)$, *F*=*Cov*$(O_j, O_i, d3, d4)$. The constraints can be relaxed according to their priority numbers.

(2a) if *p* is the same as *C.pnum*, we modify the parameters of $C$ as follows: *C.d1=0,C.d2=0*. $C(O_i, O_j)$ is appended to the CTCG, and remove the constraint which is with priority number *p* and the nearest to node $i$ or $j$ in cycle $\alpha$.

(2b) if *p* and *C.pnum* are not same, then keep the original constraints, regard *C* as an inconsistent constraint, and remove it from the constraint set.

(3) If they have different types, or have different directions and the same type except *Cov*, they are considered as incompatible. The relaxation cannot be done. If the user doesn't select one when their priority numbers are same, *C* is, by default, an inconsistent constraint, and remove it from the constraint set.

After relaxing all of redundant constraints, we can obtain the final TCG. The final TCG of Example 1 is Figure 3(a) with $R6 = Cov$(A1,T1,2,2) relaxed and $R2$ moved.

## IV. Support for Interactive Systems

### A. Interactive Authoring

Possible authoring operations for a user are as follows.

*Add an object* Addobj$(X)$: addition of a new object also needs a constraint between *X* and an old object (vertex). If there are more than one such constraint, select the one with the minimum priority number first and add it to the CTG. At the same time, its addition to the old MPST forms the new MPST of modified graph. Then, consistency checking for additional constraints on *X* will be done by the algorithm in Section 3.2.

*Remove an object* Rmvobj$(X)$: deletion of a media object also causes inefficient constraints among *X* and other objects (vertices). If there is only one such constraint in the old MPST (*X* is a leaf node), just delete it to form a new MPST. Otherwise (*X* isn't a leaf node), we need to delete all constraints on *X* from the old TCG, and find a new CTCG and a new MPST. Moreover, consistency checking will be done again.

*Add a constraint* Addcon$(C)$: addition of a new constraint possibly causes the change of MPST. A cycle will be formed after adding *C* to MPST. If *C.pnum* is smaller than another constraint, denoted as $C'$, with the maximum priority number, then we can delete $C'$ and add *C* in the old MPST to form the new MPST. Otherwise, consistency checking for *C* will be done.

*Remove a constraint* Rmvcon$(C)$: deletion of a constraint may cause incomplete constraints. Thus, we need to delete this constraint from the old TCG, and find a new CTCG and a new MPST. Moreover, consistency checking is also done again.

*Modify a constraint* Modcon$(C)$: modification of a constraint can be divided into two cases: (1)*C.pnum* is unchanged, MPST will be unchanged and the other parameters of *C* are possibly modified if *C* is in MPST; (2)*C.pnum* is changed. We need to reconstruct the MPST from CTCG with the modified *C*.

### *B. Generation of Temporal Layout*

To generate temporal layout of a scenario, MPST is considered as a directed graph. We can use breadth-first traversal to find the unique paths from a given vertex $O_s$(as the source) to all other vertices. For example, the path from $O_s$ to $O_i$:

$e(s, i_1), e(i_1, i_2), ..., e(i_p, i), (0 \le p \le N\text{-}2)$.

Assume that the presentation interval of object $O_s$ is $[b_s, t_s]$. Once such a path is found, we can get the temporal relation between $O_s$ and $O_i$ according to the constraints in this path

$b[i] = b_s + db[i], t[i] = t_s + dt[i]$,

where $b[i]$,$t[i]$ represent the start and end time of presenting $O_i$, respectively. Thus, we get a list of $2N$ timed events consisting of the start and end events, denoted as $P_b$, $P_t$

$\Gamma : (P_b[s], b_s), (P_t[s], t_s), (P_b[1], b_s + db[1]), (P_t[1], t_s + dt[1]), ..., (P_b[N], b_s + db[N]),(P_t[N], t_s + dt[N])$

then replace $t_s$ by $b_s + l(O_s)$ and sort them in the temporal order. Note that $l(O_s)$ is the length of $O_s$:

$\Gamma : (a_1, b_s + d[1]), (a_2, b_s + d[2]), ..., (a_{2N}, b_s + d[2N])$,

where $a_i$ is a start event or end event of presenting an object($P_b$ or $P_t$). If the first event is scheduled at time 0, take $b_s = -d[1]$, then $b_s$ is replaced in the above list by $-d[1]$, the final layout of presentation is as follows:

$\Gamma : (a_1, 0), (a_2, t[2]), ..., (a_{2N}, t[2N])$.

Figure 4 shows the generated temporal layout for Example 1.

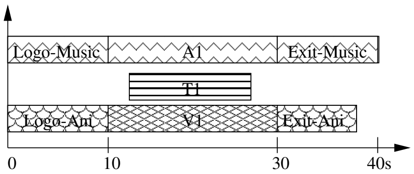


Figure 4. The layout of Example 1

## V. Analysis and Comparison

Suppose that there are $N$ media objects and $M$ constraints among them. Now we consider the efficiency of the algorithm.

Our approach consists of the following algorithms: **PreTCG** preprocesses the temporal constraint graph taking $O(M * (1 + lg(M/N))$ time. **JudgeCTCG** is used to judge if the CTG is CTCG, and its total time is *O(N+MlgN)*. Finding **MSPT** takes *O(MlgM)* time. **Removing inconsistency** takes *O((M-N+1)N)* time. **Generating temporal layout** includes the tree traversal and sorting timed events which take *O(NlgN)* time. In general, $M \ge$ *N-1* and *M* is about *N-1*. So, the total running time of our approach is *O(MlgM)*.

Many existing models do not provide consistency checking. Compared with several main consistency checking models, our method has the following advantages.

*A1. The integrated modeling and verification of multimedia constraints*. We proposed the concept of completeness of synchronization constraints, which had previously been overlooked. For convenience we divide the constraints into different categories and introduce the priority number to identify them so that we can take different policies in removing inconsistencies. By using a directed graph to express the constraints among objects, we solve the problems on completeness checking, consistency checking, constraint relaxation and layout generation.

*A2. A simple method to develop the analysis tools to support a constraint-based system*. Our method is based on graph theory which makes the problems in constraint-based systems some simple solved problems, i.e., the completeness of constraints is a mapping to the connectivity of TCG, and finding best constraints is a mapping to the spanning tree of TCG. OCPN [9] gives the semantic specification of temporal constraints, TCSP [2], Temporal Logic [10] and Time Automata [6] also efficiently specify the temporal synchronization. Although their proof may be used to check consistency, but it is difficult to develop automatic checking tools for practical multimedia authoring.

*A3. More efficient solution*. Our constraint analysis model is different from Firefly [5] and Elastic Time Model [7], although all of them are based on graphs. Firefly and Elastic Time Model specify the temporal relationship among the start and end events (in presenting objects) by using vertices to represent events and edge to label the relations between the events. In contrast, our model uses vertices to represent objects, and edges to label the constraints between objects. The complexity decreases due to only half of the number of vertices used. Moreover, by using our method, consistency checking will be finished before generating the feasible temporal layout.

*A4. Flexible implementation*. Our method supports the above problems in both interactive and batch authoring systems, especially for multi-user collaborative authoring mode. In an interactive authoring system, the user can interactively select the relaxing policy to facilitate consistency checking, constraint relaxation and layout generation. In a batch authoring system, the default processing will solve these problems automatically.

## VI. Conclusions and Future Work

This paper proposes a new method of analyzing multimedia synchronization constraints based on graphs and classification. This method provides a unified theoretical framework to solve the problems in constraint-based multimedia authoring systems, such as completeness and consistency checking, constraint relaxation and temporal layout generation. Compared to previous methods, the main advantages of our method are simple, efficient, and easy to implement and handle by nontechnical users. Because the consistency checking of spatial constraints is also important, our future work will consider consistency checking of both temporal and spatial constraints.

## References

[1] J. Allen,"Maintaining knowledge about temporal interval," *Communication of ACM*, 1983, 26(11):832-843,

[2] A.F. Ates, *et al.*, "Using timed CSP for specification, verification and simulation of multimedia synchronization,"*IEEE JSAC*, 1996,14(1):126-137.

[3] Elisa Bertino and Elena Ferrari, "Temporal synchronization models for multimedia data," *IEEE TKDE*, 1998, 10(4):612-631.

[4] G. Blakowski and R. Steinmetz, "A media synchronization survey: reference model, specification, and case studies," *IEEE JSAC*, 1996,14(1):5-35.

[5] M.C.Buchanan and P.T.Zellweger, "Specifying Temporal Behavior in Hypermedia Documents," *Proc. ACM Hypertext92*, pp.262-271, Milan,1992.

[6] J. Courtiat, R. Oliveira, "Proving temporal consistency in a new multimedia synchronization model," *Proc. ACM Multimedia96*, pp.141-152, Boston.

[7] Junehwa Song,*et al.*, "Interactive authoring of multimedia documents in a constrain-based authoring system," *Multimedia Systems*, 1999,7:424-437.

[8] Petra Hoepner, "Synchronizing the presentation of multimedia objects," *Computer Communications*, 1992, 15(9):557-564.

[9] T.D.C. Little, A. Ghafoor, "Synchronization and storage models for multimedia objects," *IEEE JSAC*, 1990,8(3):413-427.

[10] H. Ma, S. Liu, "Multimedia data modeling based on temporal logic and XYZ system," *J. Computer Sci. & Tech.*, 1999.14(2):188-193.