Interval-Based Conceptual Models for Time-Dependent Multimedia Data[1]

T.D.C. Little[†] and A. Ghafoor[‡]

[†] Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
tdcl@bu.edu

[‡] School of Electrical Engineering
Purdue University, West Lafayette, Indiana 47907, USA

MCL Technical Report 05-07-1993

Abstract–Multimedia data often have time dependencies that must be satisfied at presentation time. To support a general-purpose multimedia information system, these timing relationships must be managed to provide utility to both the data presentation system and the multimedia author.

In this paper we propose new conceptual models for capturing these timing relationships and managing them as part of a database. Specifically, we introduce and define *n*-ary and *reverse* temporal relations along with their temporal constraints. These new relations are a generalization of our earlier temporal models and establish the basis for conceptual database structures and *temporal access control* algorithms to facilitate forward, reverse, and partial-interval evaluation during multimedia object playout. The proposed relations are defined to ensure a property of monotonically increasing playout deadlines to facilitate both real-time deadline-driven playout scheduling or optimistic interval-based process playout. Furthermore, we show a translation of the conceptual models to a structure suitable for a relational database.

Keywords: Temporal Modeling, Multimedia Databases, Synchronization, Scheduling.

[1]In *IEEE Trans. on Knowledge and Data Engineering* (Special Issue: Multimedia Information Systems), Vol. 5, No. 4, August 1993, pp. 551-563. Portions of this work were presented at the 1992 Workshop on Multimedia Informations Systems held in Tempe, Arizona on February 7, 1992. The work by T.D.C. Little is supported in part by the National Science Foundation under Grant No. IRI-9211165. The work by A. Ghafoor is supported in part by the National Science Foundation under Grant No. CDA-9121771. The authors also acknowledge the support of the New York State Center for Advanced Technology in Computer Applications and Software Engineering (CASE) at Syracuse University.

1 Introduction

Multimedia refers to the integration of text, images, audio, and video in a variety of application environments. These data can be heavily time-dependent, such as audio and video in a motion picture, and require time-ordered playout during presentation. The task of coordinating sequences of these data requires synchronization among the interacting media as well as within each medium. Synchronization can be applied to the playout of concurrent or sequential streams of data and to the external events generated by a human user including browsing, querying, and editing typical of stored-data applications. This problem of synchronizing time-ordered sequences of data elements is fundamental to multimedia data.

Timing relationships between the media can be implied, as in the simultaneous acquisition of voice and video, or can be explicitly formulated, as in the case of a multimedia document with voice-annotated text. In either situation, the characteristics of each medium, and the relationships among them must be established in order to provide synchronization in the presence of vastly different presentation requirements. Consider the familiar canned *multimedia slide presentation* in which a series of verbal annotations coincides with a series of images. The presentation of the annotations and the slides occur in a sequential manner. Synchronization points correspond to the change of an image and the end of a verbal annotation, representing a *coarse*-grain synchronization between objects. A *fine*-grained example of synchronization is the lip-sync of audio and video which usually requires 25 or 30 synchronization points per second.

A multimedia system must preserve the timing relationships among the elements of the object presentation at these points by the process of multimedia synchronization. A multimedia database management system (MDBMS) must have the capability for managing the aspects of time required for time-dependent media. This problem is different from the provision of historical databases, temporal query languages [25, 27], or time-critical query evaluation [14]. Time-dependent multimedia objects require special considerations for *presentation* due to their real-time playout characteristics as data need to be delivered from the storage devices based on a prespecified schedule. Furthermore, presentation of a single object can occur over an extended duration (e.g., a motion picture). Fig. 1 illustrates such a time dependency between elements of a composite multimedia object. In this example, a sequence of text and image elements are played-out in succession based on such a prespecified schedule.

The tolerance of data to timing skew and jitter during playout varies widely depending on

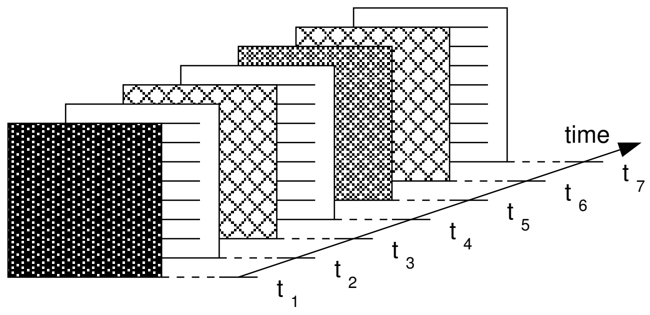

Figure 1: Time-Dependent Data Presentation

the medium. Audio and video require tight bounds on the order of hundreds of milliseconds, whereas synchronous text and images can tolerate skew on the order of seconds. Furthermore, audio and video can tolerate different absolute timing requirements during playout as the human ear can discern dropouts in audio data more readily than of video. Based on the data's tolerance to skew and jitter during playout, two approaches to providing synchronous playout of time-dependent data streams have been proposed. These consist of a real-time scheduling approach [19], and an optimistic interval-based process approach as proposed in this paper.

In addition to simple linear playout of time-dependent data sequences, other modes of data access are also possible due to the unique nature of the multimedia data objects and should be supported by a MDBMS. These include,

- Reverse
- Fast-forward
- Fast-backward
- Midpoint suspension
- Midpoint resumption

These *Temporal Access Control* (TAC) operations are feasible with existing technologies; however, when non-sequential storage devices are used with complex data compression algorithms, and random communication delays are introduced due to data distribution, the provision of these capabilities can be very difficult. Examples include viewing a motion picture backwards or reversing an animation of a series of images (sequence reversal), rapid viewing of a long sequence of time-dependent data by increasing the rate of presentation or by skipping some data (fast-forward or fast-backward), and stopping and starting of a motion picture at an arbitrary point (midpoint suspension, resumption and partial interval evaluation).

In this paper, we propose temporal-interval-based models and constraints which provide a basis for a proposed conceptual data representation and algorithmic support of the aforementioned TAC functionality. The work represents major extension and generalization of our earlier models presented in [17]; however, we do not consider the dynamic properties of user interaction (e.g., Stotts and Furuta [24]). The uncertainty created by random user interaction is an additional complexity in managing time in multimedia information systems.

The remainder of the paper is organized as follows. In Section 2, we review related work on time-dependent data storage. In Section 3, we describe interval-based modeling schemes, including our proposed models for establishing a conceptual database structure. Section 4 describes a conceptual data representation based on the new temporal models, including an example using a relational implementation. Section 5 describes algorithms for accessing the proposed conceptual models in the context of a database. We discuss characteristics of the overall modeling methodology in Section 6, and in Section 7 we conclude the paper.

2 Background and Related Work

The primary requirements for the support of time-dependent data playout in an MDBMS include the means for the identification of temporal relations between multimedia data objects, temporal conceptual database schema development, physical schema design, and synchronous access for data retrieval. In this section we briefly describe these requirements, introduce various terminology, and describe related work.

As indicated in the introduction, time-dependent data differ from historical data which do not specifically require timely playout. Typically, time-dependent data are stored using mature technologies possessing mechanisms to ensure synchronous playout (e.g., VCRs or audio tape recorders). With such mechanisms, dedicated hardware provides a constant rate of playout for homogeneous, periodic sequences of data, and concurrency in data streams is provided by independent physical data paths. When this type of data is migrated to more general-purpose computer data storage systems (e.g., disks), many interesting new capabilities are possible, including random access to the temporal data sequence and time-dependent playout of static data (animation). However, the generality of such a system eliminates the dedicated physical data paths and the implied data structures of sequential storage. Therefore, a general MDBMS needs to support new access paradigms including a retrieval mechanism for large amounts of multimedia data, and must provide conceptual and physical database schemata to support these paradigms. Furthermore, a MDBMS must also accommodate the performance limitations of the computer.

At the conceptual level, the temporal aspects of data must be modeled. Data can have *natural* or *implied* time dependencies, (e.g., audio and video recorded simultaneously). These data streams often are described as continuous because recorded data elements form a continuum during playout, i.e., elements are played-out contiguously in time. Static data, which lack time dependencies, can have *synthetic* temporal relationships (e.g., Fig. 1). The combination of natural and synthetic time dependencies can describe the overall temporal requirements of any pre-orchestrated multimedia presentation. Temporal information can be encapsulated in the description of the data using the object-oriented paradigm (e.g., Gibbs [10], Herrtwich [12]). Using such schemes, temporal information such as a time reference, playout time units, temporal relationships, and required time offsets can be maintained for specific multimedia objects. If the data are periodic, this approach can define the time dependencies for an entire sequence by defining the period or frequency of playout (e.g., 30 frames/s for video). For mixed-type, time-dependent data, there have been several proposals for their conceptual modeling and interchange format specification, most based on temporal-interval-based schemes [5, 8, 13, 23]. However, these works either neglect to consider the implications on the development of conceptual database structures to support TAC operations or do not comprehensively model time-dependent data.

Once a conceptual temporal model is established for a multimedia object, the multimedia data must be mapped to the physical system to facilitate database access and retrieval. For time-dependent multimedia data this presents some interesting challenges. Problems arise due to the strict timing requirements for playout of time-dependent data. The multimedia types of audio and video require very large amounts of storage space and must be maintained in secondary storage. In order to meet the presentation requirements for these data, various physical storage organizations have been proposed, such as storing data in contiguous blocks on a disk in the same order as playout. Some recent work on data placement on physical storage for audio data retrieval is described by Yu et al. [28, 30], Gemmell and Christodoulakis [9], Rangan and Vin [22], and Christodoulakis and Faloutsos [7]. We do not address physical storage organizations here.

The integration of conceptual and physical data models with system support for data delivery yields the functionality necessary to construct multimedia applications. System support for time-dependent data includes the study of real-time operating systems for supporting audio and video synchronization [3, 6, 20, 21, 26, 29], however, this work is beyond the scope of this presentation. We now describe our proposed conceptual models as a one component required in the construction of a multimedia information system.

3 Interval-Based Conceptual Models

In order to support time-dependent data retrieval from a MDBMS, we must provide both conceptual temporal models for describing the data and models for their storage. In this section, we introduce conceptual models that describe temporal information necessary to represent multimedia time dependencies and synchronization. Specifically, this includes a discussion of the various temporal specification methodologies, our proposed reverse and n-ary temporal relations, and our partial-interval evaluation scheme.

3.1 Basic Temporal Relations

An important and often used representation of time is the temporal interval [1]. Temporal intervals consist of time durations characterized by two endpoints, or *instants*. These interval and instant-based representations are widely investigated in the study of time. A time instant is a zero-length moment in time, such as "4:00 PM." By contrast, a time interval is defined by two time instants and, therefore, their duration. "100 ms" or "eight hours" represent temporal intervals (see Fig. 2), which we formally define as follows [2]:

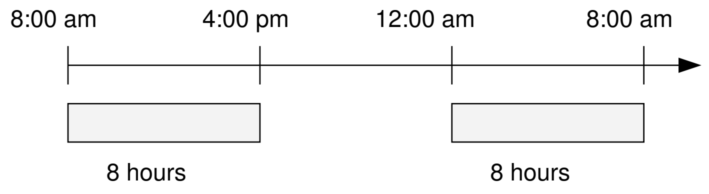

Figure 2: Instants vs. Intervals

Definition 1 *Let $[S, \leq]$ be a partially ordered set, and let a, b be any two elements (time instances) of S such that $a \leq b$. The set $\{x \mid a \leq x \leq b\}$ is called an interval of S denoted by $[a, b]$. Furthermore, any interval $[a, b]$ has the following properties:*

1. $[a,b] = [c,d] \iff a = c \wedge b = d$
2. *if* $c, d \in [a,b], e \in S$ *and* $c \leq e \leq d$ *then* $e \in [a,b]$
3. $\#([a,b]) \geq 1$

Time intervals are described by their endpoints (e.g., a and b in Def. 1 above). The length of such an interval is identified by $b - a$. The relative timing between two intervals can be determined from these endpoints. By specifying intervals with respect to each other rather than by using endpoints, we decouple the intervals from an absolute or instantaneous time reference, leading us to temporal *relations*.

Given any two intervals, there are thirteen distinct ways in which they can be related [11]. These relations indicate how two intervals relate in time; whether they overlap, abut, precede, etc. Using Allen's the representation [1], these relations are shown graphically in Fig. 3, using a timeline representation.

The thirteen relations can be represented by seven cases because six of them are inverses (the equality relation has no inverse). For example, *after* is the inverse relation of *before*, or equivalently, *before*$^{-1}$ is the inverse relation of *before* (α *equals* β is the same as β *equals* α). For inverse relations, given any two intervals, it is possible to represent their relation by using the noninverse relations only by exchanging the interval labels. In Section 3.2, we show how the ordering of deadlines required for the playout algorithms forces the use of some inverse relations. However, this use does not affect our ability to model temporal relations, being only an artifact of existing syntax conventions for temporal relations.

Temporal intervals can be used to model multimedia presentation by letting each interval represent the presentation of some multimedia data element, such as a still image or an audio segment, in what we call temporal-interval-based (TIB) modeling. We define an atomic interval to be one which cannot be decomposed into subintervals, as in the case of the presentation of a single frame of a motion picture. Intervals indicate the start times (π_α, π_β), durations (τ_α, τ_β), and end times for data elements α and β. The relative positioning between them is captured by a delay from the beginning of the first interval (τ_δ), as is their overall duration (τ_{TR}).

Fig. 4 shows audio and images synchronized to each other using the *meets* and *equals* temporal relations. For continuous media such as audio and video, an appropriate temporal representation is a sequence of intervals described by the *meets* relation because intervals abut in time and are non-overlapping, by definition of a continuous medium.

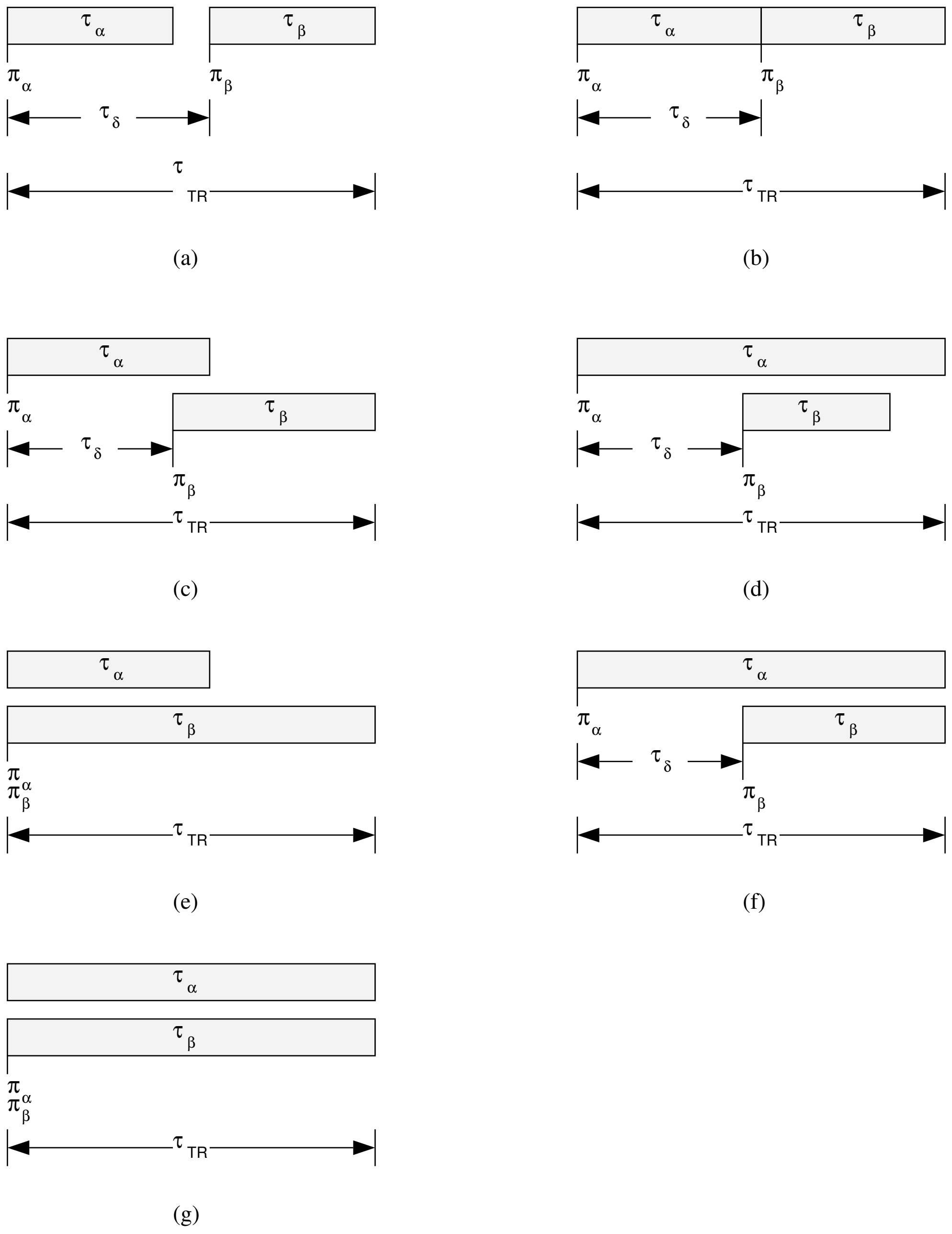

Figure 3: Binary Temporal Relations. (a) α *before* β, (b) α *meets* β, (c) α *overlaps* β, (d) α *during*$^{-1}$ β, (e) α *starts* β, (f) α *finishes*$^{-1}$ β, (g) α *equals* β

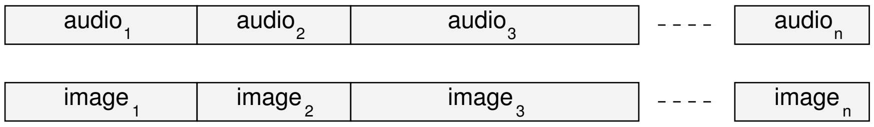

Figure 4: Synchronization of Audio and Images Represented by Temporal Intervals

Table 1: Temporal Parameters of Unified Model (P_α TR P_β)

Relation	τ_α	τ_δ	τ_{TR}
before	$< \tau_\delta$	$\neq 0$	$\tau_\beta + \tau_\delta > \tau_\alpha + \tau_\beta$
meets	τ_δ	τ_α	$\tau_\alpha + \tau_\beta$
overlaps	$< \tau_\beta + \tau_\delta$	$\neq 0$	$\tau_\beta + \tau_\delta < \tau_\alpha + \tau_\beta$
$during^{-1}$	$> \tau_\beta + \tau_\delta$	$\neq 0$	τ_α
starts	$< \tau_\beta$	0	τ_β
$finishes^{-1}$	$\tau_\beta + \tau_\delta$	$\neq 0$	τ_α
equals	τ_β	0	τ_α

In Table 1, a set of constraints indicate the timing parameter relationships between simple binary temporal relations. These constraints, identified for the simple unified Object Composition Petri Net (OCPN), are used to show uniqueness in identification of temporal relations [17]. In particular, these constraints can be used to:

1. Identify a temporal relation from the parameters τ_α, τ_β, τ_δ, and τ_{TR}.
2. Verify that the parameters satisfy a temporal relation, TR.
3. Identify overall interval duration, τ_{TR}, given a temporal relation.

This functionality proved valuable for describing the temporal component of composite multimedia objects as shown by Little and Ghafoor [17]. We reiterate an important result for the determination of durations of related temporal intervals in Lemma 1 below.

Lemma 1 *For binary temporal relations, the duration, τ_{TR}, of two related intervals τ_α and τ_β can be uniquely determined by the durations of intervals τ_α, τ_β, τ_δ, and the temporal relation,* TR, *between them.*

This lemma and Table 1 result in the following equations discerning the durations for sequential and parallel temporal relations.[2] For the sequential cases,

$$\tau_{TR} = \tau_\beta + \tau_\delta \geq \tau_\alpha + \tau_\beta \tag{1}$$

and, for the parallel cases,

$$\tau_{TR} = max\{\tau_\alpha, \tau_\beta + \tau_\delta\} < \tau_\alpha + \tau_\beta. \tag{2}$$

Using these equations and the OCPN modeling scheme, complex timeline representations of multimedia object presentation can be delineated. In the next section we introduce a major generalization of the binary temporal interval modeling approach which permits a more uniform representation of temporal intervals and supports the aforementioned TAC operations.

3.2 n-ary Temporal Relations

Binary temporal relations are sufficient for the temporal characterization of simple or complex multimedia presentations at the level of orchestration. By introducing a relationship among many intervals which we define as a *n-ary* temporal relation, we can generalize the binary temporal relations and ultimately simplify the data structures necessary for maintaining the synchronization semantics in a database. The deficiency of the binary construction process [17] is evident when many objects are to be synchronized by a single kind of temporal relation. Although a TIB scheme can easily model this case, a general approach is desired so that an efficient conceptual model for data storage results permitting simple algorithmic data retrieval and other TAC paradigms.

We therefore propose a new kind of homogeneous temporal relation for describing this case. The new temporal relation on n objects, or intervals, is defined as follows:

Definition 2 *Let P be an ordered set of n temporal intervals such that $P = \{P_1, P_2, ...P_n\}$. A temporal relation,* TR, *is called an n-ary temporal relation, denoted* TR^n, *if and only if*

$$P_i \ \text{TR} \ P_{i+1}, \ \forall i (1 \leq i < n)$$

[2]In both cases $\tau_{TR} = max\{\tau_\alpha, \tau_\beta + \tau_\delta\}$, assuming $\tau_{delta} \geq 0$.

Like the binary temporal relations, there are thirteen possible n-ary temporal relations, which reduce to the seven cases indicated in Fig. 5, after eliminating their inverses. When $n = 2$, the n-ary temporal relations simply reduce to the binary ones, i.e., for $n = 2, P_1$ TR P_2. Like the binary case, each interval indicates the start time (π_i), duration (τ^i), and end times for a data element i. The relative positioning and time dependencies are captured by a delay (τ_δ^i), as is their overall duration (τ_{TR^n}).

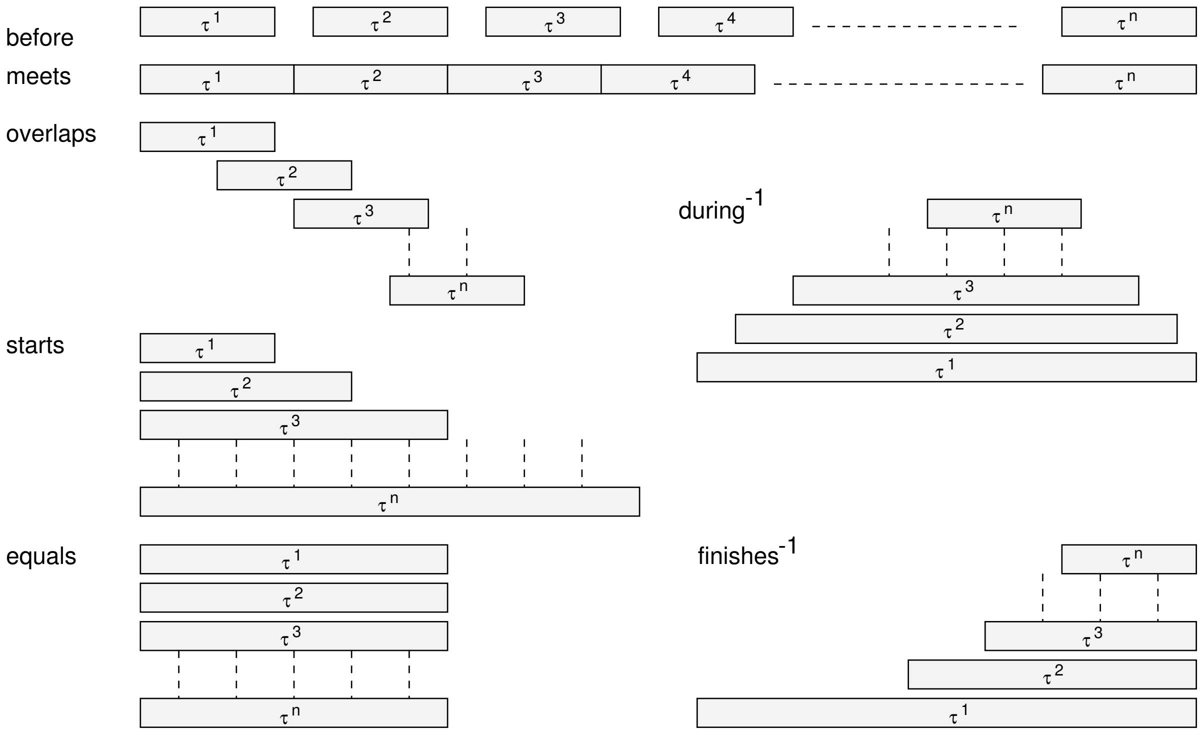

Figure 5: n-ary Temporal Relations

We now investigate the properties of the n-ary temporal relations including implications of multiple playout deadlines and temporal constraints.

3.2.1 Property of Monotonic Playout Times

If we constrain ourselves to n-ary relations with the property that

$$\pi_i \leq \pi_j,\ (1 \leq i < j \leq n)$$

then we ensure the characteristic of monotonically increasing playout deadlines, which simplify presentation algorithms and the generation of playout deadlines [17, 18]. Therefore, we concentrate on the relations *before, meets, overlaps, during*$^{-1}$, *starts, finishes*$^{-1}$, and *equals.* This does not affect our ability to model temporal relations, i.e., we have chosen a set of the 13 temporal relations such that an ordering relationship is always implied and is easily identified in both the forward and reverse directions.

3.2.2 Deadline Determination

A complex multimedia data object consists of many subobjects, each with characteristic time dependencies, and can be evaluated for the purpose of identifying the exact playout deadlines of each subobject. This task is necessary for real-time scheduling of the retrieval of objects in the presence of significant system delays [18]. The following theorem describes the relative playout time (deadline) for any object or start point of a temporal interval [19].

Theorem 1 *The relative playout deadline π_k for interval k for any n-ary temporal relation, is determined by*

$$\pi_k = c,\ (k = 1)$$

$$\pi_k = c + \sum_{i=1}^{k-1} \tau_\delta^i,\ (1 < k \leq n)$$

where c is a constant time offset.

Proof: Because timing is relative to start of the set of intervals, we let $\pi_1 = c$. $\pi_2 = c + \tau_\delta^1$ since $\tau_\delta^1 = \pi_2 - \pi_1$, by definition of delay for the binary case, and $\pi_1 = c$. Suppose, $\pi_m = c + \sum_{i=1}^{m-1} \tau_\delta^i$, for some m. We find the $m + 1$th deadline noting that for intervals P_m and P_{m+1}, P_m TR P_{m+1} by Def. 2. Therefore, $\tau_\delta^m = \pi_{m+1} - \pi_m$, or $\pi_{m+1} = \pi_m + \tau_\delta^m$, but $\pi_m = c + \sum_{i=1}^{m-1} \tau_\delta^i$, so $\pi_{m+1} = c + \sum_{i=1}^{m} \tau_\delta^i$.$\Box$

Theorem 1 specifies how to generate playout times from an n-ary temporal relation. In Section 5.2 we show this theorem applied to an algorithm for identifying playout times from our proposed temporal model.

3.2.3 Temporal Constraints

Like the binary temporal relations described in Section 3.1, a set of constraints can be identified for the timing parameter relationships among intervals of the n-ary case. Considering only the parallel and sequential n-ary cases, rather than each relation individually, we can see that for sequential cases,

$$\tau_{TR^n} = \sum_{i=1}^{n-1} \tau_\delta^i + \tau^n \geq \sum_{i=1}^{n} \tau^i \tag{3}$$

and, for the parallel cases,

$$\tau_{TR^n} = max\{\tau^1, \sum_{i=1}^{n-1} \tau_\delta^i + \tau^n\} < \sum_{i=1}^{n} \tau^i \tag{4}$$

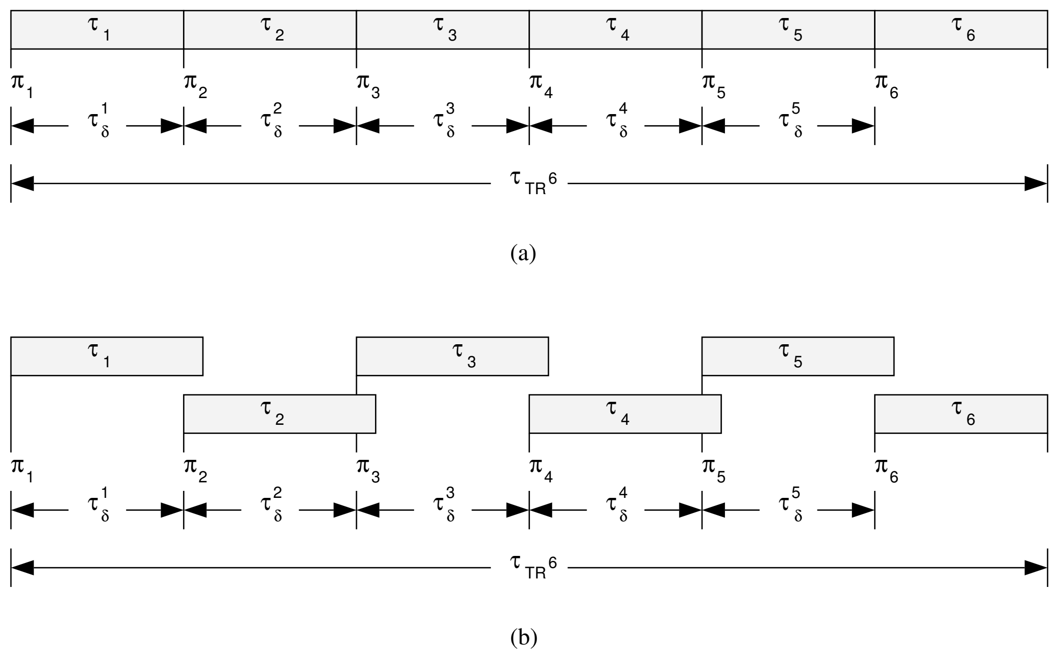

Figure 6: Illustration of the Timing Parameters for the (a) *meets* and (b) *overlaps* Relations

Therefore, similar to Equs. 1 and 2 of the binary case, we can determine whether an n-ary temporal relation is parallel or sequential, or if the temporal relation is known, verify its consistency in terms of temporal parameters.[3] Fig. 6 illustrates these parameters for

[3]Similar to the binary cases, $\tau_{TR^n} = max\{\tau^1, \sum_{i=1}^{n-1} \tau_\delta^i + \tau^n\}$ for both the parallel and sequential relations, assuming $\tau_{delta}^i \geq 0$.

the *before* and *overlaps* relations. A more general set of constraints can be identified for the n-ary relations as described in the following theorem.

Theorem 2 *There exist a set of temporal constraints for the n-ary temporal relations ($n \geq 2$) as shown in Table 2.*

Table 2: n-ary Temporal Constraints

Relation	τ^i, $(1 \leq i < n)$	τ_{TR^n}
$before$	$< \tau_\delta^i$	$\sum_{i=1}^{n-1} \tau_\delta^i + \tau^n$
$meets$	τ_δ^i	$\sum_{i=1}^{n} \tau^i$, $\sum_{i=1}^{n-1} \tau_\delta^i + \tau^n$
$overlaps$	$< \tau^{i+1} + \tau_\delta^i$	$\sum_{i=1}^{n-1} \tau_\delta^i + \tau^n$
$during^{-1}$	$> \tau^{i+1} + \tau_\delta^i$	τ^1
$starts$	$< \tau^{i+1}$, $(\tau_\delta^i = 0)$	τ^n
$finishes^{-1}$	$\tau^{i+1} + \tau_\delta^i$	τ^1
$equals$	τ^{i+1}, $(\tau_\delta^i = 0)$	τ^i, $(1 \leq i < n)$

Proof: The τ^i are determined by Def. 2 and by noting that adjacent intervals (ith to $i+1$th) form binary relationships for which we can apply the relationships of Table 1. The τ_{TR^n} are proven by induction using base cases from Table 1 as follows. For the *before, meets*, and *overlaps* relations, assume $\tau_{TR^k} = \sum_{i=1}^{k-1} \tau_\delta^i + \tau^k$. The $k+1$th relation/interval is formed by adding another interval (Fig. 7) as,

$$\begin{aligned}
\tau_{TR^{k+1}} &= \tau_{TR^k} + \tau_\delta^k - \tau^k + \tau^{k+1} \\
&= \sum_{i=1}^{k-1} \tau_\delta^i + \tau^k + \tau_\delta^k - \tau^k + \tau^{k+1} \\
&= \sum_{i=1}^{k} \tau_\delta^i + \tau^{k+1} \\
&= \tau_{TR^{k+1}}
\end{aligned}$$

For the *during*$^{-1}$ case, clearly $\tau_{TR^n} = \tau^1$, since $\forall i, \tau^i > \tau^{i+1}$. For the *starts* case, $\tau_{TR^n} = \tau^n$ and $\forall i, \tau^i < \tau^{i+1}$. For the *finishes*$^{-1}$ case, $\tau_{TR^n} = \tau^1$, since $\forall i, \tau^i > \tau^{i+1}$. For the *equals* case, $\tau_{TR^n} = \tau^1$, since $\forall i, \tau^i = \tau^{i+1}$. $\square$

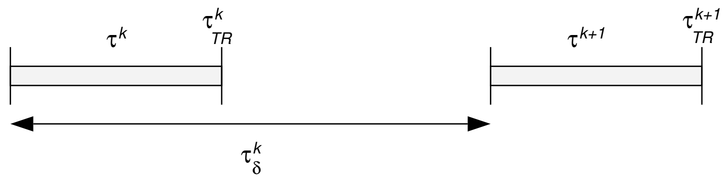

Figure 7: Interval Relations for Proof of Theorem 2

The notion of temporal intervals can also support reverse and partial playout activities, i.e., reversing the direction in time of presentation, or beginning the presentation of an object at a midpoint rather than at the beginning or end. For this purpose, reverse temporal relations are proposed in the next section. These relations, derived from the forward relations, define the ordering and scheduling required for reverse playout.

3.3 Reverse Temporal Relations

As mentioned earlier, in addition to simple linear playout of time-dependent data sequences, other modes of information access are also viable, and should be supported by a MDBMS. These TAC operations include reverse playout and partial interval evaluation for midpoint suspension and resumption. In this section we describe temporal models for satisfying these aforementioned requirements. In order to facilitate reverse playout, we first characterize reversal of time in temporal intervals, and then show their n-ary extension.

3.3.1 Reverse Binary Temporal Relations

A unique use of temporal interval processing proposed in this paper is the application of temporal relations to provide reverse presentation of objects. We introduce reverse temporal relations as distinct from inverse temporal relations, which are described by commuting the operands, i.e., $a * b = b *^{-1} a$. This characterization is essential for reverse playout of time-dependent multimedia objects, and allows us to playout, in reverse time, as defined by changing the direction of time evaluation of a temporal relation. The following definitions and lemmas characterize reverse temporal intervals and relations:

Definition 3 *A reverse interval is the negation of a forward interval, i.e., if* $[a, b]$ *is an interval, then* $[-b, -a]$ *is the reverse interval.* $[-b, -a]$ *is clearly an interval since* $a \leq b$ *and therefore* $-b \leq -a$.

Definition 4 *A reverse temporal relation* $\textsc{tr}_r$*, is defined as the temporal relation formed among reverse temporal intervals. Let* $[a, b]$ *and* $[c, d]$ *be two temporal intervals related by* $\textsc{tr}$*, then the reverse temporal relation* $\textsc{tr}_r$ *is defined by the temporal relation formed between* $[-b, -a]$ *and* $[-d, -c]$*.*

Lemma 2 *The reverse relation is a temporal relation.*

Proof: Because $[a, b]$ and $[c, d]$ are intervals, $[-b, -a]$ and $[-d, -c]$ are also intervals. Given two intervals, a relation $\textsc{tr}$ exists between them.$\square$

In relation to the other temporal models, we deal with relative timing rather than absolute endpoints. As we are more interested in the durations and properties of relative ordering, we can normalize the intervals with respect to the negative values. Consequently, the following lemma relates to the duration of a reversed interval.

Lemma 3 $\#([a, b]) = \#([-b, -a])$*, i.e., reverse intervals have identical durations as their forward counterparts.*

Proof: $\#([a, b]) = b - a = -a - (-b) = \#([-b, -a])$.$\square$

Table 3: Temporal Parameter Conversions (P_α $\textsc{tr}$ P_β to $P_{\alpha-r}$ $\textsc{tr}_r$ $P_{\beta-r}$)

Forward	Reverse	$\tau_{\alpha-r}$	$\tau_{\beta-r}$	$\tau_{\delta-r}$
before	*before*	τ_β	τ_α	τ_δ + τ_β - τ_α
meets	*meets*	τ_β	τ_α	τ_β
overlaps	*overlaps*	τ_β	τ_α	τ_δ + τ_β - τ_α
during^{-1}	*during*	τ_β	τ_α	τ_α - τ_β - τ_δ
starts	*finishes*$^{-1}$	τ_β	τ_α	τ_β - τ_α
finishes^{-1}	*starts*	τ_β	τ_α	0
equals	*equals*	τ_α, τ_β	τ_α, τ_β	0

The reverse relations are summarized in Fig. 8, noting that the reverse intervals are the reflection across a line on the time axis. To identify reverse temporal parameters ($\tau_{\alpha-r}$, $\tau_{\beta-r}$, $\tau_{\delta-r}$, and $\textsc{tr}_r$) from the forward temporal parameters (τ_α, τ_β, τ_δ, and $\textsc{tr}$), the conversions summarized in Table 3 can be used, which are derived from the consistency formulae of Table 1 and by inspection of the binary reverse relations of Fig. 8. These parameters represent

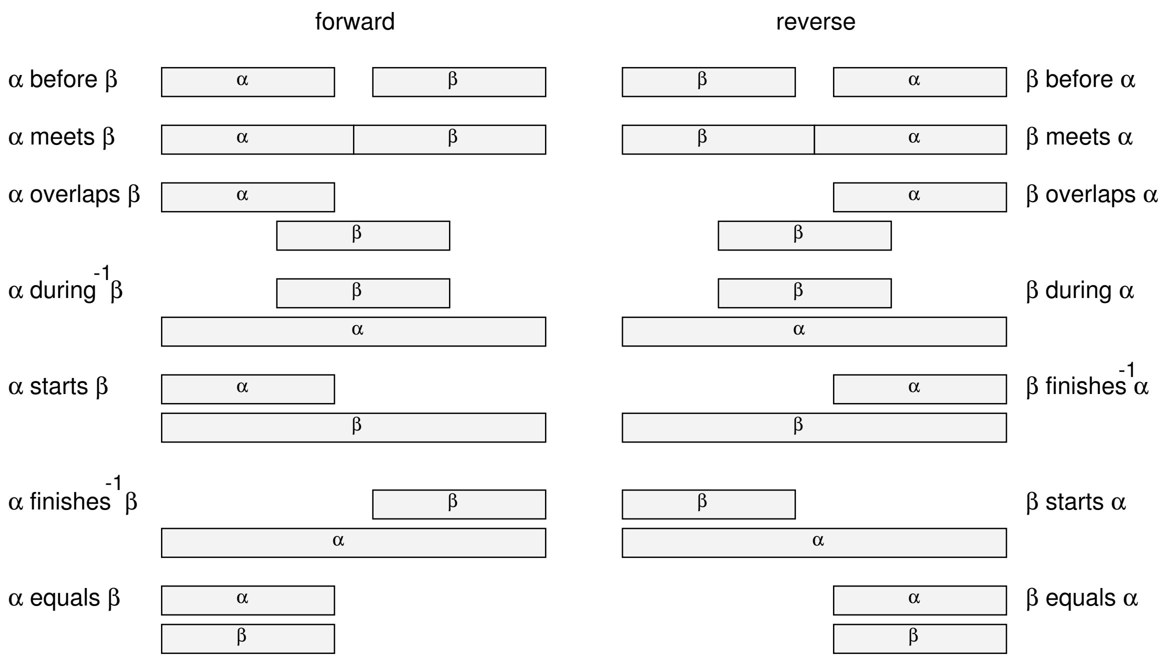

Figure 8: Forward and Reverse Relations and Intervals

the new parameters formed by the new relation when viewed in reverse.[4] Reverse temporal intervals must obey rules for temporal intervals.

3.3.2 n-ary Reverse Temporal Parameters

Like the n-ary case, we can define reverse n-ary temporal relations as follows:

Definition 5 *Let P be an ordered set of n temporal intervals such that $P = \{P_1, P_2, ...P_n\}$, and let* TR *be a temporal relation with reverse relation* TR_r. *If* TR^n *is a n-ary temporal relation on P, then a temporal relation* TR_r^n *is called a reverse n-ary temporal relation, and is defined as,*

$$P_i \ \text{TR}_r \ P_{i-1}, \ (1 < i \leq n),$$

where TR_r *can be found from Table 3.*

[4]The *during*$^{-1}$ conversion results in a non-inverted *during* relation outside of our complete set of seven relations (shown in the table). By commuting the operands we close our conversion with respect to the seven relations (and monotonically increasing deadlines) but result in an exception in the handling of the interval ordering.

Given the temporal parameters of an n-ary temporal relation, we can determine the corresponding reverse temporal parameters in a similar manner as is shown in Table 3. Ultimately these parameters enable reverse presentation timing given the forward timing parameters. To identify reverse temporal parameters for the n-ary cases (τ_r^i, $\tau_{\delta-r}^i$, and TR_r^n) from the forward temporal relations (τ^i, τ_δ^i, and TR^n), we can apply the following theorem:

Theorem 3 *Conversion from the forward temporal parameters to the reverse parameters is achieved with the equations shown in Table 4.*[5]

Table 4: n-ary Temporal Parameter Conversions

Forward	Reverse	τ_r^i, $(1 \leq i \leq n)$	$\tau_{\delta-r}^i$, $(1 \leq i < n)$
before	*before*	τ^{n+1-i}	τ_δ^{n-i} + τ^{n+1-i} - τ^{n-i}
meets	*meets*	τ^{n+1-i}	τ_δ^{n-i} + τ^{n+1-i} - τ^{n-i}, τ^{n+1-i}
overlaps	*overlaps*	τ^{n+1-i}	τ_δ^{n-i} + τ^{n+1-i} - τ^{n-i}
during^{-1}	*during*	τ^{n+1-i}	τ^i - τ_δ^i - τ^{i+1}
starts	*finishes*$^{-1}$	τ^{n+1-i}	τ^{n+1-i} - τ^{n-i}
finishes^{-1}	*starts*	τ^{n+1-i}	τ^{n-i} - τ^{n+1-i} - τ_δ^{n-i}, 0
equals	*equals*	τ^{n+1-i}, τ^i	0

Proof: The reverse interval durations, τ_r^i, are implied by Def. 5 of a n-ary reverse temporal relation. The reverse delay durations, $\tau_{\delta-r}^i$, are found by noting that adjacent intervals (ith to $i+1$th) are instances of the binary case for which conversions are tabulated in Table 3.□

3.3.3 Partial Interval Evaluation

A further enhancement for multimedia presentation is the ability to playout only a fraction of an object's overall duration. This operation is typical during audio and video editing, in which a segment is repeatedly started, stopped, and restarted. Another example of partial playout occurs when a viewer stops a motion picture then later restarts at some intermediate point (or perhaps an earlier point to get a recap). In this section we show the basis for achieving partial evaluation or fractional playout for a composite (n-ary) temporal interval. Later, in Section 5, we show an algorithm for partial interval evaluation based on this scheme.

[5]The same difficulty arises again for the *during*$^{-1}$ relation. The conversion of this relation results in a relation outside of our set of seven. The commuting of the operands implies that the correct reverse values on *during*$^{-1}$ are equal to τ^i, however, this requires a transposition of the interval indices.

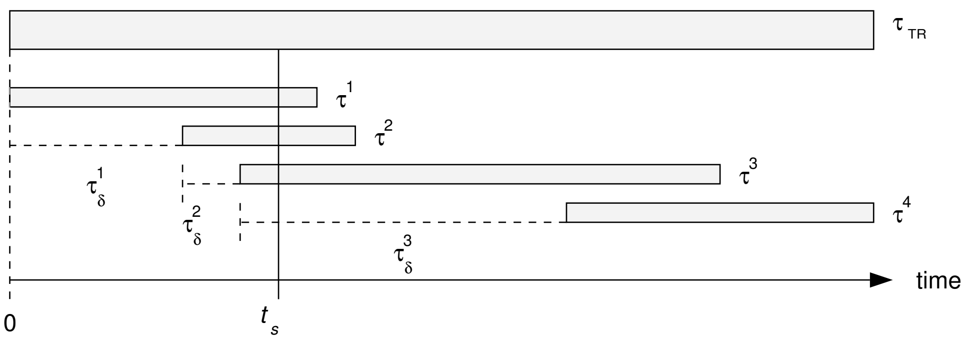

Figure 9: Partial Interval Evaluation

Consider a single temporal interval that represents the overall duration, τ_{TR^n}, of a complex n-ary temporal relationship, as shown in Fig. 9. We seek to present some fraction of this temporal interval beginning at a relative time called t_s. If $t_s < 0$, then it is too soon to consider τ_{TR^n}. If $t_s = 0$, then the whole interval and corresponding n-ary intervals must be considered. For $(0 < t_s < \tau_{TR^n})$, a fractional part of the n-ary relation must be evaluated, and finally, for $t_s \geq \tau_{TR^n}$, the interval need not be considered for evaluation at all.

A non-decomposable, or atomic, interval does not have an n-ary decomposition, and can represent the presentation of a data element. For the fractional part, we must consider both non-decomposable and decomposable intervals. For a non-decomposable interval, partial evaluation implies that the data has already been presented and need only be terminated upon expiration of the temporal interval. For a decomposable interval, the problem is to determine where to begin evaluation of the sub-intervals.

There are two cases, one for parallel temporal relations and the other for sequential relations. The following theorems characterize partial interval evaluation for these cases.

Theorem 4 *For a sequential n-ary temporal relation (before, meets), playout at t_s implies beginning playout at a subinterval k such that,*

$$\sum_{i=1}^{k-1} \tau_\delta^i \leq t_s < \sum_{i=1}^{k-1} \tau_\delta^i + \tau^k, \ (1 \leq k < n).$$

Proof: Consider the first interval. If $t_s < \tau^1$, then some time is left in τ^1. If $\tau^1 \leq t_s < \tau_\delta^1$, then t_s falls in the delay period between intervals τ^1 and τ^2, and no k exists. For $t_s \geq \tau_\delta^1$, t_s

occurs during the next interval τ^2. From Theorem 1, we know that the kth interval starts at $\sum_{i=1}^{k-1} \tau_\delta^i$, and ends at $\sum_{i=1}^{k-1} \tau_\delta^i + \tau^k$.$\square$

Theorem 5 *For a parallel temporal relation (overlaps, during^{-1}, starts, $finishes^{-1}$, and equals), playout at t_s indicates playout of all subintervals k such that,*

$$\sum_{i=1}^{k-1} \tau_\delta^i \leq t_s < \sum_{i=1}^{k-1} \tau_\delta^i + \tau^k, \ (1 \leq k < n)$$

$Proof$: For intervals k such that $t_s < \pi_k + \tau^k$, there is clearly time left. However, the ones that will be active are indicated by $\pi_k \leq t_s < \pi_k + \tau^k$, but $\pi_k = \sum_{i=1}^{k-1} \tau_\delta^i$, by Theorem 1.

Next, we show how the temporal parameters can be aggregated in a suitable manner for database storage.

4 Database Models for Time-Dependent Media

In Section 3 we have shown a new kind of temporal relation which can describe the timing among sets of multimedia objects. We now show how the proposed relations and their associated formulation lead to the development of conceptual temporal models suitable for storage and retrieval of time-dependent multimedia data.

The OCPN model, used to formally specify sets of synchronized media elements, leads to a specific conceptual database model by selecting synchronized groups of objects and by assigning their temporal parameters to a database elements [17]. Our discussion here parallels this earlier work, however, the new temporal relations provide a more efficient and homogeneous structure and provide new TAC functionality. We design conceptual database structures for the elements of the database which can ultimately be applied to a relational, network, or other data model. We do not assume any specific model for the multimedia database management component, rather, we build a framework which can be integrated, or used in a complementary fashion, with other DBMS schemes. Later, we show examples of this conceptual model applied to a relational data model.

4.1 Proposed Temporal Data Model

To capture the semantics of the n-ary temporal relations and the object orchestration technique, we need to group multimedia objects and identify them with temporal parameters. The first node type template for this purpose is the leaf or *terminal* node as indicated in Fig. 10(a). This node has attributes which indicate media type (text, image, video, etc.) and a pointer which indicates the storage location of the data for presentation. This node type indicates the elements at the finest grain for synchronization. There are no restrictions on the in-degree of this node type to permit links to multiple instances of composite multimedia objects.

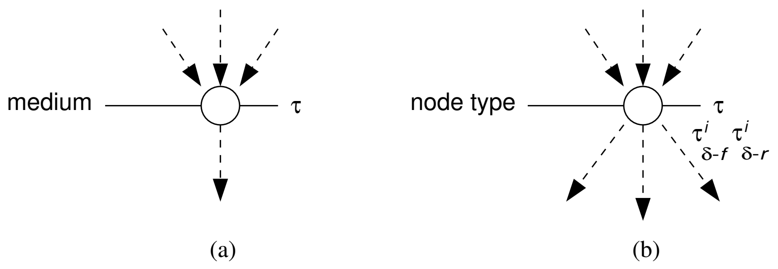

Figure 10: (a) $Terminal$ Node Type and (b) $Nonterminal$ Node Type

$Nonterminal$ nodes have a structure as indicated in Fig. 10(b). Attributes for this template include node type, pointers to children, and temporal parameters (forward delay, $\tau_{\delta-f}$, reverse delay, $\tau_{\delta-r}$, and overall temporal duration, τ_{TR^n}). Again, in-degree can be arbitrary. Note that the forward temporal parameters are necessary and sufficient for reconstituting a temporal relation and deriving the reverse parameters [17]. Clearly these parameters should be maintained in a database to support time-dependent data retrieval.

Using these node types, sets of synchronized elements can be represented in a conceptual temporal framework developed for an OCPN. Consider the *multimedia slide presentation* example to illustrate this development. First, we identify and assign individual images and audio segments to *terminal* nodes along with their locations. We select a parallel relation *equals*, with unique durations for each slide-talk pair, and form a set of synchronized pairs of objects using a *nonterminal* node type. In this case the assigned attributes include the forward, reverse, and overall temporal durations $(\tau_{\delta-f}, \tau_{\delta-r}, \tau_{TR^n})$. Because we have a set of slide-talk pairs, a single *nonterminal* node type is indicated to aggregate the set of sequentially related object pairs. A hierarchical conceptual model for this example is shown in Fig. 11.

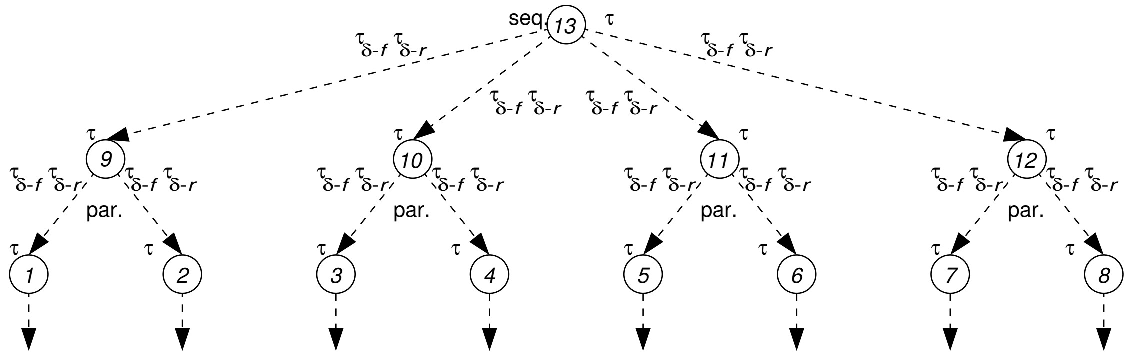

Figure 11: Conceptual Model for the *Multimedia Slide Presentation*

Based on the intended access or query methods, additional attributes can be assigned to the nodes, but these are not considered here. For more complex timing interactions, aggregation of components provides a means of isolating components of a presentation at a coarse-grain of synchronization. The level of *terminals* provides the finest grain of timing whereas levels towards the root of a hierarchy are comprised of composite object aggregations having a more coarse inter-object synchronization grain.

4.2 Database Creation

The construction of a multimedia database is the process of assigning data instances to the developed database model. We assume multimedia data elements are stored and managed by some entity which can be accessed by pointers from *terminal* nodes defined by the conceptual model. Database construction begins by assigning multimedia data to the *terminal* nodes, including the required time durations of the data elements. To establish temporal parameters for *nonterminal* nodes, Lemma 1 is used by combining timing parameters of temporally related terminal elements and assigning them to their connecting *nonterminal* parent nodes. As this process is repeated, timing values are propagated up the database hierarchy until timing values are established at all nodes. Determination of the reverse delay parameters can be performed using the conversion formulae provided in Section 3. This process can be performed at database creation time, or dynamically in the event of a reverse playout TAC operation.

4.3 A Relational Data Model for Temporal Data

Using the conceptual model described above (Section 4.1), data structures can be implemented to capture time-dependent data. These data structures indicate specific choices for implementation of the presentation algorithms. In this section we show how the conceptual model is applied to a relational data structure.

The organization of a temporal hierarchy is naturally tree-like. We can capture this hierarchy using a relational model by defining a relation called NODES to which we can apply temporal constraints. Any *terminal* or *nonterminal* node can be described by a tuple in the relation:

$$\text{NODES}(\mathit{node\_no}, \mathit{node\_type}, \mathit{duration}, \mathit{subject})$$

Each node is uniquely identified by an integer, *node_no*. The remaining attributes, *node_type*, *duration*, and *subject* characterize the node and describe accompanying temporal relationships. *node_type* describes the kind of temporal integration for assembly of subnodes, if any exist. Values of this attribute are *nonterm_s*, *nonterm_p*, and *terminal*, indicating the sequential, parallel, and terminal cases, respectively. The duration attribute describes the overall temporal interval for a possibly complex object. *subject* is an attribute used to describe the content of the node for illustrative purposes only, however, by using a complementary data schema, it is possible to provide content-based attributes as well [16].

To support a hierarchy of conceptual nodes, we define a second relation to represent the children, or SUBNODES, of an arbitrary node described by the relation NODES. For any node in NODES, multiple subnodes are possible, representing subobjects for temporal integration. A tuple is defined to indicate node-to-subnode links as follows:

$$\text{SUBNODES}(\mathit{node\_no}, \mathit{subnode\_no}, \mathit{index}_f, \mathit{index}_r, \mathit{delta}_f, \mathit{delta}_r)$$

In this case, the attributes *node_no* and *subnode_no* uniquely identify the tuples. index_f and index_r are attributes indicating the ordering among a set of subnodes with the same parent for forward and reverse playout, respectively. By using these indices, an arbitrary number of subnodes can be specified as required for an n-ary temporal relation, as discussed in Section 4.1. Lower values of these indices indicate preceding (in time) subnodes, with increasing values indicating directional traverse of the temporal hierarchy for evaluation of the parent's temporal relation. However, ordering is not necessarily identical for the forward and backwards cases, as established by the reverse n-ary relation. $delta_f$ and $delta_r$ establish the durations, $\tau_{\delta-f}$ and $\tau_{\delta-r}$ for forward or reverse temporal relationships between two adjacent subnodes.

Finally, to locate data corresponding to objects for playout, a relation TERMINALS is defined with *node_no* as the unique identifier as follows:

$$\text{TERMINALS}(node\_no, medium, filename)$$

This separate relation is defined apart from the NODES relation because many nodes are not terminal elements and do not require these attributes. Only a subset of all nodes will indicate terminal elements which capture single-medium base types (e.g., single images, audio segments, etc.). Additional information can be defined as part of these relations such as details of formatting requirements, but are not shown here.

Table 5: Tuples of the TERMINALS Relation

node_no	*medium*	*location*
1	image	–
2	audio	–
3	image	–
4	audio	–
5	image	–
6	audio	–
7	image	–
8	audio	–

Referring to the *multimedia slide presentation* example, we now illustrate the creation of a relational database to capture a temporal hierarchy. Suppose the slideshow is on a topic of "Planets." The temporal hierarchy for this example is illustrated in Fig. 11. To map this conceptual model to the relational data structures we first identify each node with a unique identifier, and assign each terminal element to a tuple in the TERMINALS relation along with its location and media type. These tuples are shown in Table 5.

Next, for each node we create a tuple describing whether it is a *terminal*, *nonterm_p*, or *nonterm_s* type. We also identify its duration and a relevant topic or keyword. These tuples are summarized in Table 6.

Table 6: Tuples of the NODES Relation

node_no	*node_type*	*duration*	*subject*
1	terminal	10	"Venus"
2	terminal	10	"Venus"
3	terminal	5	"Mars"
4	terminal	5	"Mars"
5	terminal	15	"Saturn"
6	terminal	15	"Saturn"
7	terminal	10	"Neptune"
8	terminal	10	"Neptune"
9	nonterm_p	10	–
10	nonterm_p	5	–
11	nonterm_p	15	–
12	nonterm_p	10	–
13	nonterm_s	40	"Planets"

Table 7: Tuples of the SUBNODES Relation

node_no	*subnode_no*	$index_f$	$index_r$	$delta_f$	$delta_r$
9	1	1	2	0	0
9	2	2	1	0	0
10	3	1	2	0	0
10	4	2	1	0	0
11	5	1	2	0	0
11	6	2	1	0	0
12	7	1	2	0	0
12	8	2	1	0	0
13	9	1	4	10	0
13	10	2	3	5	5
13	11	3	2	15	15
13	12	4	1	0	10

Finally, the SUBNODES relation table is created by identifying the children of each node in the temporal hierarchy. For each node, the position indices and forward and reverse delays ($\tau_{\delta-f}$ and $\tau_{\delta-r}$) are identified from the conceptual model. Table 7 shows these tuples.

5 Access Algorithms for Multimedia Databases

We have presented a specification methodology and conceptual models suitable for database storage of time-dependent multimedia data. In conjunction with these models, we have developed access algorithms that we describe in this section. These algorithms provide functionality for partial interval playout, playout reversal, and playout deadline determination as required for real-time scheduling approaches.

5.1 Playout Scheduling Approaches

In formulating an approach to media delivery suitable for satisfying the timing requirements for multimedia data, one must realize that most workstation technology lacks sufficient performance to support the playout of complex multimedia objects. Viable applications will be supported either by ample or marginal system performance. For systems with ample performance, data delivery is simplified by system components without bottlenecks or significant latencies. For systems with marginal performance, techniques must be applied to guarantee delivery of data in a timely fashion. For this reason, we perceive two approaches to data delivery: the real-time scheduling approach and the optimistic, or ideal, interval-based process playout approach.

In a real system, latencies exist in storage devices, communications, and multiprocessing that make enforcement of an ideal timing specifications difficult. These delays can be larger than the playout durations of individual data elements requiring synchronization, and a mechanism must be employed for compensation. Methods of overcoming these delays rely on some form of real-time scheduling (e.g., [15, 19]).

As system performance becomes less marginal, or as playout time durations increase, the system delays become less significant, warranting the development of *ideal* presentation algorithms. Images, text, and audio segments can be effectively synchronized with such an ideal scheme because their presentation durations are large (greater than a few seconds), whereas video and audio frames have a very short presentation interval (less than one second) and require specific resource scheduling.

For both approaches the objects and their temporal hierarchy must be evaluated. For the real-time scheduled approach this requires a recursive traversal of the hierarchy to identify object deadlines, with subsequent scheduling of playout for the deadlines [19]. For the ideal approach, tree traversal includes immediate presentation of component objects without specific consideration of system latencies, i.e., sufficient system performance is assumed to be available. In the remainder of this section we show we show algorithms for TAC operations in the ideal case and an algorithm for generating deadlines for the real-time scheduled approach. Mechanisms for satisfying a playout schedule once generated in this manner are described elsewhere [19].

5.2 Determination of Playout Time Based on Temporal Intervals

The temporal hierarchy can be evaluated for the purpose of identifying the exact deadlines for playout of all objects by applying Theorem 1 for any object or start point of a temporal interval using either forward or reverse parameters. The expression in Theorem 1 can be evaluated in n steps where n is the number of elements indicated by the temporal relation. This is a very important consideration for real-time deadline evaluation when expenditure of time is critical.

Alternatively, the original OCPN model, used to describe a composite multimedia object, can be simulated directly [18]. The following algorithm, based on Theorem 1, finds the playout time, or deadline, for any object or start time of a temporal interval using either forward or reverse parameters. This algorithm uses as input a parameter $t_{elapsed}$ to indicate the current elapsed time throughout its processing, and is initially set to zero.

Playout Deadline Algorithm

```
1. For object $Object$, identify attributes, get elapsed time offset, $t_{elapsed}$.

2. If $Object$ is not a terminal then evaluate its subnodes:

    (a) For each subnode with increasing node index $i$:

        i. Recursively invoke algorithm on the $i$th subnode with $t_{elapsed}$.
        ii. $t_{elapsed} = t_{elapsed} + \tau_{\delta}^{i}$.
```

3. If $Object$ is a terminal then its deadline, π_{object}, is equal to $t_{elapsed}$.

Basically, an object's subnodes are identified in a recursive tree-traversal. As each subnode is identified, its temporal parameters are evaluated via Theorem 1 to generate a set of playout times. It is important to note that for such evaluation, the parallel and sequential information (used in later algorithms presented here) has no significance because deadlines are generated for each interval and a real-time scheduling approach can interpret the deadlines uniformly [19]. Furthermore, the nth occurrence of each τ_δ (forward and reverse) is ignored since it has no interpretation without a subsequent object to which it can relate.

Consider the *multimedia slide presentation* as an example. Applying the deadline generation algorithm initially to node no. 13 results in the identification of a *nonterminal* element. Therefore, its subnodes are evaluated in increasing order of their indices. Node no. 9 yields yet another subnode which then leads to two terminal nodes. For node no. 1, $\pi_1 = t_{elapsed} = 0$. The return of the recursive call on subnode no. 1 causes $t_{elapsed}$ to be set to 0, and $\pi_2 = 0$ for node no. 2. The subsequent return of the recursive call on node no. 9 causes $t_{elapsed}$ to be set to 10, prior to recursively descending the hierarchy for the second set of audio-image pairs. Evaluation continues in this manner resulting in the following set of playout deadlines: $\Pi = \{\pi_i\} = \{0, 0, 10, 10, 15, 15, 30, 30\}$. These deadlines can be generated in either the forward or reverse direction or beginning at an arbitrary point in the temporal hierarchy.

5.3 Retrieval Algorithm for Reverse Playout

To support reverse playout of an object represented by our temporal data model we merely require evaluation of the reverse direction indices. These can be derived from either existing forward temporal parameters or from the temporal relations themselves. Rather than storing identical object durations, indices can be used to establish the playout ordering in the forward and reverse directions, hence the need for the position indices in the relational model. The modification to the original presentation algorithm [17] to support forward and reverse playout is shown as a program fragment as follows (shown later as part of the complete playout algorithm):

1. For each subnode in increasing order of index i_d:

where i_d is either the forward or reverse position index. In the general case, we would like to be able to stop at any point in the temporal hierarchy and restart in either the forward or reverse direction. This functionality is provided though partial-interval evaluation.

5.4 Partial-Interval Evaluation

A further enhancement required for multimedia presentation is the ability to playout partial objects. Partial interval evaluation allows us to stop in the middle of presentation and resume in either the forward or reverse direction.

For the fractional part, we must consider both nondecomposable (atomic) and decomposable intervals. A non-decomposable interval indicates the actual playout of a multimedia data element. When this is the case, partial evaluation implies that the data element has already been presented and need only be terminated upon expiration of the temporal interval. For a decomposable interval, the problem is to determine where to begin evaluation of the subintervals. To stop and then restart during presentation of a decomposable interval, we can save the elapsed playout time, $offset_f$ since starting presentation. For the reverse direction, the reverse elapsed playout time, $offset_r$, can be computed by subtracting the forward elapsed playout time from the overall duration of the object under consideration, i.e.,

$$offset_r = \tau_{object} - offset_f$$

Based on Theorems 4 and 5, we describe the modifications necessary to support partial interval evaluation. These modification require as input a parameter $offset$ indicating the relative time offset from the start of the indicated object (generalized for both forward and reverse playout in the complete algorithm).

For the Sequential Cases

1. For each subnode in increasing order of index i:
   (a) If $0 \leq offset < \tau^i$ then (time left in this node)
      i. Recursively invoke algorithm on the ith subnode with $offset$.
      ii. Delay by $\tau_\delta^i - \tau^i$, $offset := 0$.

   (b) Elseif $\tau^i \leq offset < \tau^i_\delta$ then (some delay left)
      i. Delay by $\tau^i_\delta - offset, offset := 0$.
   (c) Else $offset \geq \tau^i_\delta$ (no delay left)
      i. $offset = offset - \tau^i_\delta$ (try next subnode)

For the Parallel Cases

1. For each subnode in increasing order of index i:
   (a) If $0 \leq offset < \tau^i$ then (time left in this node)
      i. Fork a new process and recursively invoke algorithm on the ith subnode with $offset$
   (b) Else $offset \geq \tau^i$ (no time left in this node)
   (c) If $0 \leq offset < \tau^i_\delta$ then (some delay left)
      i. Delay by $\tau^i_\delta - offset, offset := 0$.
   (d) Else $offset \geq \tau^i_\delta$ (no delay left)
      i. $offset = offset - \tau^i_\delta$ (try next subnode)

With these modifications the playout of an object can be initiated at an arbitrary time within the duration τ_{TR^n} of an object.

5.5 Complete Playout Algorithm

The complete algorithm is shown below, considering all of the proposed modifications to our original presentation algorithm to support TAC functionality. In this algorithm $offset$ indicates either the forward or reverse elapsed time depending on the direction d of playout.

Complete Playout Algorithm

1. For object $Object$ identify subnodes, and direction d.
2. If $Object$ is not a terminal then evaluate its temporal relation:
   (a) If temporal relation is sequential then:

```
        i. For each subnode in increasing order of index $i_d$:
           A. If $0 \leq offset < \tau^i$ then
              • Recursively invoke algorithm on the $i_d$th subnode with $offset$.
              • Delay by $\tau^i_\delta - \tau^i, offset := 0$.
           B. Elseif $\tau^i \leq offset < \tau^i_\delta$ then
              • Delay by $\tau^i_\delta - offset, offset := 0$.
           C. Else $offset \geq \tau^i_\delta$
              • $offset = offset - \tau^i_\delta$
    (b) If temporal relation is parallel then:
        i. For each subnode in increasing order of index $i_d$:
           A. If $0 \leq offset < \tau^i$ then
              • Fork a new process and recursively invoke algorithm on the $i_d$th subnode
                with $offset$.
           B. Else $offset \geq \tau^i$
           C. If $0 \leq offset < \tau^i_\delta$ then
              • Delay by $\tau^i_\delta - offset, offset := 0$.
           D. Else $offset \geq \tau^i_\delta$
              • $offset = offset - \tau^i_\delta$
3. If $Object$ is a terminal then present data on the appropriate I/O device for its specified
   duration.
4. Terminate when all recursive invocations have completed.
```

6 Discussion

The proposed n-ary temporal relations offer a more elegant means of representing timing information than our previous binary models. In particular, the n-ary model captures an arbitrary number of component objects with a common temporal relationship without requiring many levels of hierarchy.

We presented a number of consistency formulae in Theorem 2. These formulae can be used at the time of object creation to establish a consistent database of temporal parameters by using a tree traversal procedure akin to the playout algorithms (not shown here).

Alternatively, a timing coercion technique can be applied to incompletely specified timing hierarchies to achieve a cohesive presentation as has been applied by Buchanan and Zellweger [4].

In a similar manner, reverse timing parameters can be computed at object creation time and stored in the temporal database or can be computed dynamically at the time of sequence reversal by using the conversions of Table 4. This process can also rearrange the ordering of intervals that are incorrectly arranged (by the author) for sequence reversal (e.g., arbitrarily ordered intervals with the *starts* relation). The penalty for this computation depends on the complexity of the represented objects and their frequency of reversal. Frequent sequence reversal suggests the use of stored reverse parameters whereas infrequent reversal can be adequately provided by dynamic parameter computation.

We have implemented the aforementioned algorithms of Section 5 using the C programming language and process forking as part of our development of a general purpose distributed multimedia information system in the Multimedia Communications Laboratory at Boston University. The temporal model has also been incorporated into an interactive motion picture database application called the Virtual Video Browser [16]. In this system the relational data structures supporting object timing complement an application-specific schema for a database of motion pictures. The result is a database that supports both TAC functionality as well as content-based retrieval of motion picture components.

7 Conclusion

In this paper we have proposed two new temporal models for time-dependent data retrieval in a multimedia database management system. These models, the n-ary and *reverse* temporal relations, are shown to be useful for facilitating storage and retrieval of time-dependent multimedia data by applying a temporal hierarchy supported by a relational data model.

Because the proposed models are restricted to having the property of monotonically increasing playout deadlines for represented objects, algorithms for synchronous data retrieval are greatly simplified, both for identification of deadlines necessary for real-time scheduling of playout, or for optimistic process playout synchronization. Furthermore, algorithms are shown that allow traversal of the temporal hierarchy in a manner that provides forward, reverse, or partial interval evaluation.

Acknowledgement

The authors wish to thank the anonymous referees for their helpful suggestions and careful review of this paper. We also thank Jehad Shehadeh for his effort in implementing the playout algorithms and Mamoun Filali for his candid examination of our mathematics.

References

[1] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of the ACM*, November 1983, Vol. 26, No. 11, pp. 832-843.

[2] T.L. Anderson, "Modeling Time at the Conceptual Level," *Improving Database Usability and Responsiveness*, P. Scheuermann, Ed., Academic Press, New York, 1982, pp. 273-297.

[3] D.P. Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism," *Computer*, Vol. 24, No. 10, October 1991, pp. 51-57.

[4] M.C. Buchanan and P.T. Zellweger, "Specifying Temporal Behavior in Hypermedia Documents," *Proc. European Conf. on Hypertext'92*, Milan, Italy, December 1992.

[5] D.C.A. Bulterman, G. van Rossum, and R. van Liere, "A Structure for Transportable, Dynamic Multimedia Documents," *Proc. Summer 1991 Usenix Conf.*, Nashville, Tennessee, June 1991, pp. 137-155.

[6] D.C.A. Bulterman and R. van Liere, "Multimedia Synchronization and UNIX, *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[7] S. Christodoulakis and C. Faloutsos, "Design and Performance Considerations for an Optical Disk-based, Multimedia Object Server," *Computer*, Vol. 19, No. 12, December 1986, pp. 45-56.

[8] Committee Draft International Standard, "Information Technology – Standard Music Description Language (SMDL)," ISO/IEC CD 10743, April 1, 1991.

[9] J. Gemmell and S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval," *ACM Trans. on Information Systems*, Vol. 10, No. 1, January 1992, pp. 51-90.

[10] S. Gibbs, "Composite Multimedia and Active Objects," *Proc. OOPSLA'91*, Phoenix, AZ, October 1991.

[11] C.L. Hamblin, "Instants and Intervals," *Proc. of the 1st Conf. of the Intl. Society for the Study of Time*, J.T. Fraser, et al., Ed, Springer-Verlag, New York, 1972, pp. 324-331.

[12] R.G. Herrtwich, "Time Capsules: An Abstraction for Access to Continuous-Media Data," *Proc. 11th Real-Time Systems Symp.*, Lake Buena Vista, FL, December 1990, pp. 11-20.

[13] R.G. Herrtwich and L. Delgrossi, "ODA-Based Data Modeling in Multimedia Systems," *International Computer Science Institute Tech. Rept.* TR-90-043, August 1990.

[14] W.C. Hou, G. Ozsoyoglu, and B.K. Taneja, "Processing Aggregate Relational Queries with Hard Time Constraints," *Proc. ACM SIGMOD Intl. Conf. on the Management of Data*, Portland, OR, June 1989.

[15] L. Li, A. Karmouch, and N.D. Georganas, "Real-Time Synchronization Control in Multimedia Distributed Systems," *Proc. 4th IEEE ComSoc Intl. Workshop on Multimedia Communications (Multimedia '92)*, Monterey, CA, April 1992, pp. 294-305.

[16] T.D.C. Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh, "A Digital Video-on-Demand Service Supporting Content-Based Queries," *Proc. ACM Multimedia'93*, Anaheim CA, August 1993, pp. 427-436.

[17] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. on Selected Areas in Comm.*, Vol. 8, No. 3, April 1990, pp. 413-427.

[18] T.D.C. Little and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," *IEEE J. on Selected Areas in Comm.*, Vol. 9, No. 9, December 1991, pp. 1368-1382.

[19] T.D.C. Little and A. Ghafoor, "Scheduling of Bandwidth-Constrained Multimedia Traffic," *Computer Communications*, Vol. 15, No. 5, July/August 1992, pp. 381-387.

[20] J. Nakajima, M. Yazaki, and H. Matsumoto, "Multimedia/Realtime Extensions for the Mach Operating System," *Proc. Summer 1991 Usenix Conf.*, Nashville, TN, June 1991, pp. 183-198.

[21] M. Pasieka, P. Crumley, A. Marks, and A. Infortuna, "Distributed Multimedia: How Can the Necessary Data Rates be Supported?" *Proc. Summer Usenix Conf.*, Nashville, TN, June 1991, pp. 169-182.

[22] P.V. Rangan and H.M. Vin, “Designing File Systems for Digital Video and Audio,” *Proc. 13th Symp. on Operating Systems Principles (SOSP’91), Operating Systems Review*, Vol 25, No. 5, October 1991, pp. 81-94.

[23] C. Schramm and M. Goldberg, “Multimedia Radiological Reports: Creation and Playback,” *Proc. SPIE Medical Imaging III: Image Capture and Display*, SPIE Vol. 1091, Newport Beach CA, January 1989, pp. 191-201.

[24] P.D. Stotts and R. Furuta, “Petri-Net-Based Hypertext: Document Structure with Browsing Semantics,” *ACM Trans. on Office Automation Systems*, Vol. 7, No. 1, Jan. 1989, pp. 3-29.

[25] R. Snodgrass, “The Temporal Query Language TQuel,” *ACM Trans. on Database Systems*, Vol. 12, No. 2, June 1987, pp. 247-298.

[26] B.I. Szabo and G.K. Wallace, “Design Considerations for JPEG Video and Synchronized Audio in a Unix Workstation,” *Proc. Summer Usenix Conf.*, Nashville, TN, June 1991, pp. 353-368.

[27] A.U. Tansel, M.E. Arkun, and G. Ozsoyoglu, “Time-by-Example Query Language for Historical Databases,” *IEEE Trans. on Software Engineering*, Vol. 15, No. 4, April 1989, pp. 464-478.

[28] J. Wells, Q. Yang, and C. Yu, “Placement of Audio Data on Optical Disks,” *Proc. 1st Intl. Conf. on Multimedia Information Systems ’91*, Singapore, January 1991, pp. 123-134.

[29] L.C. Wolf, “A Runtime Environment for Multimedia Communications,” Presented at the 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video, Heidelberg, Germany, November 1991.

[30] C. Yu, W. Sun, D. Bitton, Q. Yang, R. Bruno, and J. Tullis, “Efficient Placement of Audio Data on Optical Disks for Real-Time Applications,” *Comm. of the ACM*, Vol. 32, No. 7, July 1989, pp. 862-871.