CHAPTER III
CHAPTER III

Synchronizing Multimedia Data

Multimedia denotes a temporal, spatial and/or logical relationship between objects, data entities or media streams[154]. Together, a group of data objects and the temporal structure in which they are bound comprise what and when of multimedia composition. The major challenge in multimedia information management is how to synchronize various types of data, both, in space and time in order to compare complex multimedia objects. The synchronization specifications are a part of the description of a multimedia object and are captured at the time of object creation. This specification describes all spatial and temporal dependencies of the related objects and sub-objects. Synchronization requirement plays a key role while designing a multimedia data base. The traditional data model is not capable of handling synchronization aspects and the heterogeneous nature of multimedia data. Hence in this chapter we describe a technique to specify the temporal interrelationships among media objects as a part of document structure model developed in the previous chapter. The objective of these specifications is to ultimately predict the presentation times of data units resulting in synchronized playout. The instant is our basic unit of time and we used event-based technique to model temporal relations.

This chapter has been divided into three sections. The first section gives some definitions, Allen's relations and explains the concept of multimedia
synchronization. In the second section we describe how our model incorporates the three essential features required by a temporal model and in the third section we give a representation of the temporal relations given by Allen in terms of triggers.

3.1 Preliminaries

3.1.1 Definitions

A temporal scenario is an instance of a set of activities that are in some way related in time. Temporal models capture, describe and specify a temporal relation i.e. they completely describe a temporal synchronizations.\[131\]

Events are arbitrary distributed points in time used for the ordering of actions and their coordinated access to shared resources. Events may be regarded as a special kind of information passing mechanisms between objects. The information is passed at start/end of action. [165] Rambaugh et al. have defined event as a 1-way transmission of information from one object to another. It is not like a subroutine that returns a value.\[137\]

A time-dependent media object is presented as a media stream and there exists temporal relation between consecutive units of media stream.\[25\] If the presentation duration of all units of a time-dependent media object are equal, it is called continuous, eg, video consists of fixed presentation duration frames. A time-independent media object is one where the semantic of the content does not
depend upon the presentation or the time-domain eg image, text. Time-dependent media are sometimes called periodic and time-independent media are called aperiodic\[6].

3.1.2 Allen's relations

Allen[8] gave a temporal interval algebra for representing the temporal relations between two intervals. A diagram to this effect is given below:

Before

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

Meets

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

Parallel-start

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

During

\[\begin{array}{c}
  \text{B} \\
  \text{A}
\end{array}\]

Parallel-end

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

Overlap

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

equal

\[\begin{array}{c}
  \text{A} \\
  \text{B}
\end{array}\]

\textit{fig.3.1}
A temporal interval is identified as the basic anchored specification of time during which an activity takes place. The elements of the algebra are sets of seven basic relations (as shown in fig. 3.1) that can hold between two intervals and their inverses.

3.1.3 Synchronization

It is known that integrated media processing is an important characteristic of a multimedia system. The emphasis on the integrated processing is due to the fact that there are inherent dependencies between the various media objects. These dependencies must be reflected in integrated processing whether it is storage manipulation, communication, capturing and specially presenting of media objects. Thus it implies that whatever be the type of processing it has to be coordinated (or synchronized) keeping in view the dependencies as the constraints or parameters of coordination (or synchronization). Some authors use the term synchronization in a general sense comprising of spatial synchronization and temporal synchronization[154,165]. A brief description of these is given below:

3.1.3.1 Temporal synchronization

Temporal synchronization ensures that events are carried out in a particular order with respect to time. It deals with the temporal dependencies between media objects. Temporal dependencies are more prominent in case of time dependent objects. Temporal synchronization is defined as the occurrence of multiple events
at the same instance of time[127]. According to Gibbs, Dami and Trichtzis the objective of multimedia synchronisation are starting and stopping of multimedia objects at desired points in time[57]. The basic characteristics possessed by interacting objects are: the number of involved objects, the behaviour of individual objects while waiting for synchronisation, the influence of those objects on others already synchronising, to order synchronisation events, the relationship between interacting objects and coverage of real-time aspects[153]. Temporal synchronization is specified in terms of temporal relations between objects. Examples of temporal relations are the relation between a video and audio object; relation between audio narration and presentation of slides in a slide show.

**Intra-media vs intermedia synchronization**

Since temporal synchronization represents relative placement of data items with respect to time, in order to clarify mechanisms supporting temporal relations, temporal synchronization is classified into intra-media(or intra-object) and inter-media(or inter-object) synchronization. Intra-object synchronization refers to the time relation between various presentation units of a single media objects[44,153]. Intra-object or intra-media synchronization is applicable in case of continuous media like video and audio. An example is the time relation between the frames of a video sequence. Figure 3.2 below illustrates the intra-media synchronization between frames of a video shot depicting motion of a ball.
The timeline (x-axis) is divided into time frames of equal length (30ms). The position of the ball in the subsequent frames is changing in each frame as time progresses. Now if these frames are displayed at the rate of 25 frames/sec (this value has been experimentally obtained in [154]) the viewer will perceive it as a moving ball. But if the rate changes the motion will be jerky and will not convey desired effect to the user. This time coordination between frames is intra-media synchronization.

The figure 3.3 below shows intermedia synchronization requirements for a presentation consisting of audio, video followed by a few graphics and then animation sequence and text.
In case a multimedia object is composed of various streams a different kind of synchronization -intermedia synchronization- needs to be considered. The multimedia object is composed of Audio1, Video1, G1, G2, G3, Animation and Text. The relative placement of these sub-objects is shown in fig. 3.3. to achieve intermedia synchronization. Thus, intermedia synchronization refers to timing constraint among a set of multimedia streams as depicted above.

**synthetic vs live synchronization**

Another classification of synchronization is synthetic or live. In case of synthetic synchronization, temporal relations have been assigned to media objects that were created independently of each other. For example, in a multimedia slide presentation, synchronization points correspond to the change of an image and the end of verbal annotation, representing coarse-grain synchronization or synthetic synchronization. Here, the specification is done explicitly. This synthetic synchronization is often used in presentation and a multimedia system must preserve the timing relationships among the elements of the object by the process of multimedia synchronization. For synthetic synchronization it is necessary to use a model for specification and manipulation of temporal conditions and operations. In case of fine-grain or live synchronization the temporal relations are defined implicitly. Example of a fine-grained example is lip synchronization in case of a movie. Further, Mourlas have classified multimedia synchronization as low level and high level. Low level synchronization refers to the hardware capability
for maintaining temporal features of a multimedia composition within a range. High level synchronization refers to sequence of actions on multimedia objects & handling of events that occur & cause other actions. A temporal model helps in specification of temporal synchronization constraints between two multimedia objects[131]. Thus the temporal model forms an integral form of a multimedia database.

3.1.3.2 Spatial synchronization

Spatial synchronization refers to the layout relationships that are used for the presentation of a media object on an output device at a certain point of time in a multimedia presentation. The output device could be two-dimensional(paper or monitor) or three-dimensional(three-dimensional projector). Like the temporal model, spatial model is also an essential part of an abstract MMDBMS model. Spatial properties pertaining to objects meant for spatial coordination include points, lines, polygons and volumes. Spatial relations have been classified into three types. One, topological relations that describe neighborhood and incidence(overlap, disjoint etc.); second, directional relations that describe order in space(eg south, northeast etc.); and third, distance relations that describe ranges of distance between objects(eg. far and near). We discuss the first two types only because distance relations are domain dependent. Eight fundamental topological relations have been defined in literature.[49] These relations are computed using four intersections over the concepts of boundary and interior of point-sets.
between two regions embedded in a two dimensional space as described in the
next para.

Let \( A^0 \) and \( B^0 \) be the interiors of objects A and B respectively; and \( A^* \) and \( B^* \) be
the boundaries of objects A and B respectively. Then the following combinations
of \( A^0, B^0, A^* \) and \( B^* \) give the eight topological relations. (Checking whether each
intersection is empty or not empty produces 16 possible results, however, only 8
are meaningful)

\[
A^* \cap B^* = \emptyset, A^0 \cap B^0 = \emptyset, A^* \cap B^0 = \emptyset, A^0 \cap B^* = \emptyset \quad \text{A disjoint B}
\]

\[
A^* \cap B^* \neq \emptyset, A^0 \cap B^0 = \emptyset, A^* \cap B^0 = \emptyset, A^0 \cap B^* = \emptyset \quad \text{A touch B}
\]

\[
A^* \cap B^* = \emptyset, A^0 \cap B^0 \neq \emptyset, A^* \cap B^0 \neq \emptyset, A^0 \cap B^* = \emptyset \quad \text{A inside } B^*
\]

\[
A^* \cap B^* \neq \emptyset, A^0 \cap B^0 = \emptyset, A^* \cap B^0 \neq \emptyset, A^0 \cap B^* \neq \emptyset \quad \text{A overlap B}
\]

\[
A^* \cap B^* \neq \emptyset, A^0 \cap B^0 = \emptyset, A^* \cap B^0 = \emptyset, A^0 \cap B^* \neq \emptyset \quad \text{A cover } B^*
\]

\[
A^* \cap B^* \neq \emptyset, A^0 \cap B^0 = \emptyset, A^* \cap B^0 = \emptyset, A^0 \cap B^* = \emptyset \quad \text{A equal B}
\]

* A inside B is same as B contains A

**A covers B is same as B covered-by A

Figure 3.4 shows the 8 topological relations derived above.
Spatial relationships between objects are very important in multimedia databases because they implicitly support fuzziness (i.e., certain level of variance from user queries) by their qualitative property. A temporal interval algebra as the one given by Allen [8] can be used to interpret topological relations in one-dimensional space.
as is shown in the diagram above. This order given by Allen can also be used to capture the directional aspects in addition to the topological relations. We can classify the directional relationships into three categories, namely, strict directional relations (north, south, east and west), mixed directional relations (northeast, southeast, northwest and southwest), and positional relations (above, below, left and right). These are given in the table below:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A st B</td>
<td>A is South of B</td>
</tr>
<tr>
<td>A nt B</td>
<td>A is North of B</td>
</tr>
<tr>
<td>A wt B</td>
<td>A is West of B</td>
</tr>
<tr>
<td>A et B</td>
<td>A is East of B</td>
</tr>
<tr>
<td>A nw B</td>
<td>A is Northwest of B</td>
</tr>
<tr>
<td>A ne B</td>
<td>A is Northeast of B</td>
</tr>
<tr>
<td>A sw B</td>
<td>A is Southwest of B</td>
</tr>
<tr>
<td>A se B</td>
<td>A is Southeast of B</td>
</tr>
<tr>
<td>A lt B</td>
<td>A is Left of B</td>
</tr>
<tr>
<td>A rt B</td>
<td>A is Right of B</td>
</tr>
<tr>
<td>A bl B</td>
<td>A is Below B</td>
</tr>
<tr>
<td>A ab B</td>
<td>A is above B</td>
</tr>
</tbody>
</table>
It can be noticed that the definition of A above B includes A north of B (A nt B), part of A northwest of B (A nw B), and part of A northeast of B (A ne B) according to the definitions. Though the complete discussion is beyond the scope of this work, below we illustrate with help of few figures that give some insight into how the spatial relations are not disjoint.

The figure above shows six cases of A northwest of B. Out of these first three fall in the category of disjoint topological relations and the next three in the category of touch topological relation. Another example for A north of B is given in figure 3.6.
In this chapter we will discuss the temporal synchronization aspects. Thus in the present context multimedia synchronization (sometimes referred to as orchestration) is a concept/technique which coordinates the presentation of various media streams in a time domain.
3.2 Proposed Model

Though the problem of multimedia system has been addressed by various applications but no formalization has been proposed for a general description for synchronization in document structure. In this section we develop a model to specify temporal relation and incorporate this temporal information with the structural model already developed in the last chapter. Here we deal with synthetic and high level synchronization in an appropriate data model as described previously[143]. Also, now onwards we use the terms synchronization specification, temporal specification and temporal relation interchangeably as our focus in only on temporal synchronization.

A general multimedia synchronization problem has two aspects. First, the modeling, representing and specifying the structural as well as temporal aspects of the objects; and second, to achieve the synchronized presentations using these temporal specifications. Presentation consists of temporal scenarios. Temporal scenario represents an instance of a set of activities that are in some way related in time. An example of a temporal scenario is a slide show having audio and visual components. These two activities are related in time as well as dependent on time. This dependency/relations must be captured by the model i.e. temporal synchronization is realized in terms of a temporal model.

A temporal model is completely characterised by 3 related components[131]:

72
1. the basic time unit

2. contextual information associated to the basic time units and

3. type of time representation technique

Our temporal model is based on integration of temporal aspects within the structural model. As a result, synchronization reduces to the rendering of objects according to temporal specifications.

The document is our unit of information. The structure of a document is represented as an aggregation hierarchy of parts or components. It consists of nodes and edges (branches). The nodes represent the logical components of the document. The edges at this level represent consist-of relationship indicating an inclusion relation between composite and subordinate node. The root of the tree is a general node that describes the summary structure of a document[143].

The overall format of a node has been discussed in the last chapter and is represented below:

<table>
<thead>
<tr>
<th>ident- attributes</th>
<th>synchronization specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>level</td>
</tr>
</tbody>
</table>

fig.3.7
3.2.1 Incorporating temporal model in the structural model

The synchronization specifications (can also be called presentation specifications) of our node format consists of three fields, namely, offset, duration and tr. We now explain how these three attributes are sufficient to characterize the temporal characteristics. According to Little and Perez[131], a temporal model is completely characterized by a basic time unit, contextual information and representation technique. We have chosen instant as our basic time unit. Contextual information is present in the node format in the form of tr attribute which can have values before/after/overlap/equal/parallel-start etc., i.e. any of the thirteen binary relations given by Allen[8]. It is to be noticed that tr attribute gives temporal relation between the children of the nodes in which it appears. In order to remove indeterminacy (which is inherent in Allen’s relations) we have included two more attributes namely, offset and duration. These two attributes are used along with the tr to completely specify temporal relation between two data units. Duration represents the time-distance between two instants pertaining to the same object (the two instants are begin and end of an object display). Offset removes indeterminacy by specifying the time gap between play of two data units related by tr. For the third component, i.e. representation technique we develop the concept of triggers based on events. Thus our representation technique is event-based. This is dealt with in section 3.3. We now give more details about the components of our temporal model.
3.2.1.1 basic time unit (BTU) and events

Two possible basic time units are instant and interval. The basic time unit we have chosen for our model is instant. Instant is a zero-length moment in time. If we choose instant as the basic time unit it can represent both type of events i.e. those occurring instantaneously and those occurring over a period of time i.e. interval. It is possible to represent an interval using two instants $t_1$ and $t_2$ where $t_2 \geq t_1$ and $t_2 - t_1$ represents duration of the interval[105,131,132].

events/synchronization points

An event is an occurrence in time that can be instantaneous or can occur over a period of time.[144] Event is also defined as an atomic unit of action.[132] We call instantaneous happenings events and something happening over a period of time is an activity. The instantaneous events can be modeled through the basic time units i.e. begin-obj(may be beginning of display of an object) is given by some instant and end-obj(end of display of an object) is given by some instant. In order to model an event occurring over a period of time, we use two special instants, begin-event and end-event to represent an interval. Sometimes instantaneous events are also known as synchronization points or reference points[158]. For coarse-grain synchronisation, it is crucial to synchronize the beginning and end of the display of objects (for finer grain of synchronisation we keep breaking into subjects till we reach atomic object - similar to as suggested by Hoepner[132]).
Thus, for us two events of interest are begin and end (begin and end refer to display of the data item they are associated with).

3.2.1.2 Contextual information

The contextual relation in our model is represent by the offset, duration and tr fields in the structure of a node which comprise of synchronization specifications and serves as a means of synchronization[143]. The value of field tr can be any of the thirteen relations given by Allen.

The offset field gives the gap between the temporal placement of two nodes. This is interpreted in association with the temporal relation field given in the parent node. If the temporal relation in the parent of a particular node is before which implies that the left child is to be presented before the next child, then offset field of the left child will give how much before it is to be played before its next sibling. Thus offset gives time gap between two events belonging to two different data units. Diagramatically it can be shown as below:

\[ \text{fig.3.8} \]
In the above diagram \( t_1 \) represents end of object 1 (before \( t_1 \) object 1 is undergoing some activity e.g. being displayed) and \( t_2 \) represents beginning of object 2 (after \( t_2 \) object 2 carries out with some activity). Offset is the time gap between \( t_1 \) and \( t_2 \), where \( t_1 \) and \( t_2 \) are time instances associated with events of different objects. (In some cases the offset field of the rightmost sibling may be redundant but mentioned for maintaining uniformity of the node structure)

The duration or extent of a media item may be implicit or explicit depending on the media type of the item[132]. For example a video has an implicit duration associated with its media item i.e. a LTU (Logical Time Unit) for a frame[25]. An image needs to be assigned a duration explicitly. This may be explicitly assigned or derived from the surrounding presentation. An example of a derived duration is the image that is displayed when an audio begins and continues until the audio finishes. In order to satisfy relative temporal constraints that derive the duration of an instance or satisfy constraints, individual media items may have to be scaled in time. If the duration is to be lengthened then the object instance can either be played slower or can be repeated until the specified duration is reached. If the duration is to be shortened, then the instance can be played faster or can be cut short[66]. In both the cases, then the duration or extent of the item as incorporated in the presentation may require some other value. This may be in terms of an absolute value or scale factor with respect to other presentation units. As a consequence, start time, finish time (and position for spatial constraints) cannot be stored as attributes because these parameters cannot be specified for a
media item in isolation from the rest of the presentation but need to be specified in relation to other media items or with respect to the presentation as a whole. The start time of an object instance can be given in a number of ways. The most common is to define an instance's start time relative to other object. For our model the reference time is the start of display of the first data unit of the document. The representation of duration in terms of events is given below.

Let the following timeline represent the presentation of an object starting from a time $t_1$, continuing for a duration $d$ and finishing at time $t_2$. $t_1$ and $t_2$ are the instants corresponding to the begin and end events with reference to the playout of this object.

![fig.3.9](image)

The **temporal relation** between the media objects is given by tr attribute. Here we are talking about temporal relations within the sub-components of a document. We treat temporal relation as attribute of a node. This attribute is ignored in case of leaves. This attribute is explained further in the contextual information subsection later. The relation is specified implicitly during capturing of media objects, if the goal of presentation is to present the media in the same way as it
was captured eg. audio/video recording and playback. In the case of presentations that are composed of independently captured or otherwise created media objects eg. slide show, the temporal relation is specified explicitly.

The leaves provide finest grain of synchronization whereas the levels towards the roots of the hierarchy represent composite objects having more coarse inter object synchronization[27]. The synchronisation information is inherited by the lower levels i.e., the temporal relation between the children is specified in their parent object. We have no special class for synchronizable objects as all objects in our model are synchronizable. In our document model, class node represents a synchronizable object because of the presence of attribute tr[143]. The main purpose of the presentation/synchronization specifications is to predict the presentation time for each of the data units[142].

We have mentioned about basic time units(BTU), duration, offset and temporal relation but till now we have not taken care of the start time. In our model we will have no absolute value of the start time, instead, the start time will be relative to the beginning of the playout of the first data unit of the document. With only the knowledge of the presentation time of the first object, the presentation times of all other objects can be calculated based on their temporal relations. This further implies that the particular time instant at which a presentation can start easily be modified without affecting the entire schedule. The third component of the temporal model i.e. representation technique is dealt with in section 3.3.
3.2.1.3 Interpreting tr field

By virtue of our document structure the tr field represents n-ary relations as this field specifies the relation between the siblings on the next level (i.e. between the children of the particular parent). We describe this using an example. (In the nodes we show only those attributes relevant to the temporal aspects)

![Diagram](image)

**fig.3.10**

The first number in the node is for object id, the numbers in parentheses give offset and duration respectively; and the last field gives temporal relation between the children of the specific node. It is important to note that for the composite nodes (i.e. non-leaves) the value of duration is denoted by (d). This is because the duration of the composite node will depend on the duration of its children and the temporal relation between them. It is possible that duration of the composite is not same as calculated by combining the durations of its children according to the
temporal relation between them (in a bottom-up fashion). For example the value of (d) in node 2 should be 8 but it may actually be less or more. There are two approaches to solve this conflict: one, update the duration of node 2, or second, extend/truncate the presentation on last child of this node that is node 6 by invoking appropriate action. At this point it is beyond the scope of this work to go into further details about this conflict, nevertheless, it is discussed in the next chapter [142].

In the second level i.e. next to root there are two nodes with ids 2 and 3; they are related by 'before' (mentioned in node 1). Similarly at the next level nodes 5 and 6 are related by 'before'; 7, 8 and 9 'par-start'. The indeterminate factor is taken care of by the field offset which gives the quantitative separation between two events. Diagramatically these objects related by n-ary temporal relations can be represented as:

```
Before  2     3

Before  5     6

parstart  7
          8
          9
```

*fig.3.11*
An algorithm to obtain the above representation from the document tree structure is given below:

The document has been stored as n-ary tree. Each node has two pointers. The first pointer (first) points to the leftmost child and the second pointer (next) points to the next sibling. The tree traversal starts from the root node.

```
algo displaytemprel(root);
/*this algorithm takes root of the document tree as input and gives the set of temporal relations as output*/
enqueue(root);
while queue(not empty) do
/*the tree is traversed in breadth-first fashion and each node is added to the queue to be processed later*/
    ptr := dequeue(queue);
    parent := ptr;
    /*for a non-leaf node all its children are put in a list and also each child is added to the queue*/
    if parent.first <> nil
        ptr := parent.first;
        while ptr.next <> nil do
            appendlist(ptr.id)
            enqueue(ptr);
```
ptr := ptr.next;

end while;

appendlist(list, ptr.id);

enqueue(ptr);

/*temporal relations and the nodes related by it are displayed as output*/

write parent.tr, list

end while;

end algo;

Procedure enqueue adds an entry to the queue; dequeue removes the entry at the head of the queue; appendlist creates a list of node ids related by the same temporal relation as given in the parent node.

3.3 Representation of temporal relations

In this section we use two terms begin-trigger and end-trigger to represent the temporal relation tr between two (or more) objects belonging to a multimedia document.

Though each event is a unique occurrence, but we classify them in two categories—begin events and end events. Begin event is an instant which signifies the starting of (play of) an object and end event denotes end of (play of) an object. Our
temporal specification focuses on the relationships between four pair of events, i.e., begin-begin, begin-end, end-end, and end-begin[144].

These two events - begin and end (associated with some object) are the points in time where other synchronizable objects can be attached. When any one of these is reached the controller object or source (i.e. the object with which these events are associated) makes a call to called object or destination via one of the latter's synchronization points. The alignment of actions (play or wait in our case) at synchronization points is done by applying some contextual information which are the temporal relations in our case.

3.3.1 Triggers

An event $E_1$ is defined as a trigger of another event $E_2$ if the occurrence of $E_1$ also indicates occurrence or happening of $E_2$. As mentioned earlier we have two events of interest i.e. begin and end. Thus begin-trigger is the event that specifies the beginning of another event and end-trigger is an event which forces end of an already ongoing event. Diagramatically begin-trigger will be represented by $\rightarrow$ (from the source to the destination) and end-trigger by $\longrightarrow$ (from source to destination). Let $X$ and $Y$ be two objects to be displayed. There are two synchronization points associated with each of them, namely, $X_{\text{beg}}$, $X_{\text{end}}$, $Y_{\text{beg}}$ and $Y_{\text{end}}$. $X$ and $Y$ will interact with each other in terms of time via these four events/synchronization points[144]. Few possibilities are listed below.
(1) X and Y start at the same time

\[ X_{\text{beg}} \rightarrow Y_{\text{beg}} \quad X_{\text{beg}} \text{ triggers } Y_{\text{beg}} \]

(2) X after Y

\[ Y_{\text{end}} \rightarrow X_{\text{beg}} \quad Y_{\text{end}} \text{ triggers } X_{\text{beg}} \]

(3) X before Y

\[ X_{\text{end}} \rightarrow Y_{\text{beg}} \quad X_{\text{end}} \text{ triggers } Y_{\text{beg}} \]

(4) X and Y finish at the same time

\[ X_{\text{end}} \rightarrow Y_{\text{end}} \quad X_{\text{end}} \text{ triggers } Y_{\text{end}} \]

Duration of an activity over time for example display of X is represented by pair <X_{\text{begin}}, X_{\text{end}}>.

The interaction between two objects via the synchronization points can be represented by the following state diagram:
The source object is in any state. Begin trigger from source object causes transition from the stop state to play state for destination object and the end trigger does vice versa. We could add another state wait. In the wait state the waiting object performs action as given by its silence action.

3.3.2 Translating Allen’s relations

Allen’s[8] relations can also be represented by using trigger constructs. Here we have given seven relations. Rest six relations can be written in the similar manner by inversing the first six relations. The symbols in parenthesis give the offset which is either 0 or some numeric value. This offset gives the gap between triggering synchronization point of the source and the triggered synchronization point of the destination, for example in the relation before $A_{\text{end}}$ is $y$ time units before $B_{\text{beg}}$. The representation of basic binary relations given by Allen in terms of begin and end triggers is shown in fig.3.13[144].

As can be seen from the illustration below the first two relations - before and meets - have the same representation in our notation. The only difference is the value of offset. Using the same logic we can combine overlap and equal. So our set of relations is reduced to before(includes meet), overlap(includes equal), par-start, par-finish and during. The notation that we have given for binary relations can easily be extended to n-ary relations.
Before

meets

par-end

par-start

during

overlaps

equal

$A_{end}$ triggers $B_{begin} (y)$

$A_{end}$ triggers $B_{begin} (0)$

$A_{end}$ triggers $B_{end} (0)$

$A_{begin}$ triggers $B_{begin} (0)$

$B_{begin}$ triggers $A_{begin} (x)$

$B_{end}$ triggers $A_{end} (y)$

$A_{begin}$ triggers $B_{begin} (x)$

$A_{end}$ triggers $B_{end} (y)$

$A_{begin}$ triggers $B_{begin} (0)$

$A_{end}$ triggers $B_{end} (0)$

fig.3.13
3.3.3 Templates for the reduced set of temporal relations

Temporal structure of a multimedia composition describes the properties of the composition in the temporal dimension. Various objects/data units could be related by the same temporal relation. Because same set of relations is shared by various compositions, it is useful to keep the temporal structures as templates(classes) and substitute different data objects into it. For example a slide show(or a movie) keeps a similar temporal relations regardless of the contents of the objects. This will avoid duplication[144]. Temporal relations are independent of what data is being presented. So we define another layer and for each of the above five relation we define five templates which will be instantiated at the time of presentation by substituting appropriate values of source and object ids. Fig.3.14 gives templates for the five relations described above.(For the rest of the six Allen’s relations which are the inverses of the six relations given above, excluding, equal, templates can be drawn on the similar lines).

<table>
<thead>
<tr>
<th>Before/meets:</th>
<th>Src-id(A)</th>
<th>dest id(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin-trigger</td>
<td>A&lt;sub&gt;end&lt;/sub&gt;( )</td>
<td></td>
</tr>
<tr>
<td>end-trigger</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Fig.3.14*(contd)
The value in parenthesis is offset; 0 in case of meets and a positive integer in case of before; in case of equal offsets are 0 whereas in overlap it is a positive integer.

fig. 3.14
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. We have introduced concepts that describe temporal information necessary to represent multimedia time dependencies and synchronization. Also it is important to note that we have used triggers as one way relationships, for example in case of relation equal one could argue that $A_{\text{begin}}$ is begin trigger for $B_{\text{begin}}$ as well as $B_{\text{begin}}$ is begin trigger for $A_{\text{begin}}$. In order to keep the approach simple and comprehensible we do not consider triggers as two-way relationships.

Considering the above templates, the temporal composition can be represented as:

```
      △
     /   \
  synch obj     synch obj
```

*Fig. 3.15*

The source and the destination objects are the synchronizable elements in this composition. The attributes of the objects of composition are begin and end triggers (which are events identifiable by object ids) and offsets (integer value in terms of BTU).

Templates are declared as nodes having the following structure:
2x2 array where each element of the array is (objid, event) pair.

[1,1] element is the begin-trigger for the source

[2,1] element is the end-trigger for the source

[1,2] element is the begin-trigger for the destination

[2,2] element is the end-trigger for the destination

Below we give an algorithm that generates a list of appropriate temporal templates from the document tree. This list of templates is stored as a linked list called templist. This list will later be used to generate presentation schedules[142].

The document has been stored as n-ary tree; each node has two pointers; the first pointer(child) points to the leftmost child and the second pointer (bros) points to the next sibling. Starting from the root node the tree is traversed from Left to Right at each level, and the temporal templates get instantiated accordingly. For any parent node temporal relation field gives the relative timings of the children of a particular node, for instance, if for a particular node temporal relation is overlap, it implies that the playout of its children are overlapping; as a result the overlap
template will get instantiated. For each parent node we assume that the relation it has with one of its children (i.e., the child with earliest start time) is that of parallel start. Thus the template for parallel start is instantiated. For all the children from left to right with ith child as the source and (i+1)th child as the destination the particular template as given by the temporal relation is instantiated. Another temporal relation parfinish is required between the child with latest finish time and the parent. So parfinish template is instantiated after the last child of the parent is visited (before the next sibling of the parent is visited). Thus for each node traversed \( n = 2 + m-1 \) (2 for parstart and parfinish and \( m-1 \) for \( tr \)) templates are instantiated where \( m \) is the no of children of the particular node. The structure of the temporal templates has been described earlier. It consists of two objects ids i.e., source id and destination id; and a \( 2 \times 2 \) array. Each element of the array gives an (object id, event) pair; event could be begin or end. Each instantiated template constitutes a node. These nodes are linked to form a link list. The creation of link list ends with the tree traversal. The nodes of this link list are in form of source/destination pairs. Each node of the link list looks like\[144]\:

\[
\begin{array}{ccc}
| id_1 | & id_2 |\\
| --- | & --- |\\
| trig[1,1] | & trig[1,2] |\\
| trig[2,1] | & trig[2,2] |
\end{array}
\]

\textit{fig.3.16}

92
id_1 is the object identifier of the first object (source); id_2 is object identifier for the second object (destination).

trig[1,1] gives the (id, event) pair which is begin trigger for object id_1.

trig[1,2] gives the (id, event) pair which is begin trigger for object id_2.

trig[2,1] gives the (id, event) pair which is end trigger for object id_1.

trig[2,2] gives the (id, event) pair which is end trigger for object id_2.

event can take values begin or end.

nxt is the pointer to the node of the same type.

The process explained above is given in algo tree-traversal given below:

```
algo treetraversal(root)
/* this algorithm takes root of the document tree as the input and creates a link list called templist after traversing the tree. The nodes of this link list are of the type described in fig.3.16*/

t = root
traverse-tree(t)
display templist()
end main
```
algo traverse-tree (t)

/*this algorithm traverses the tree from left to right in breadth-first fashion and
calles create-node to instantiate the templates*/

while (t <> null)
{
   /*called for the first time to create parstart template between root and is
   first child*/

   create-node(t)

   if t->child <> null

   /*children of node t are put in a queue*/

   { qptr = createquenode(t->child)

     appendinque(qptr)

     t = t->bros

   }

   if queue (not empty)

   /*serve retrieves the first node from the head of the queue*/

   p = serve ( )

   traverse-tree(p)

}

end tree-traverse
algo create-node (t)
/*this algorithm creates the node for templist by setting appropriate values of src
and dest and deciding which template is to be instantiated*/
while (t <> null)
{
    if t->child <> null
    {
        src = t->id
        p = t->child
        dest = p->id
        /*parstart template instantiated between parent and the first child*/
        fresh = instantiate (src, dest, parstart)
        addnodeinlist (fresh)
        while (p->bros <> null)
        {
            src = p->id
            p = p->bros
            dest = p->id
            /*template according to tr field given in the parent node is
            instantiated between the siblings. instantiate sets
            appropriate values for the fields of the node as given in the
            fig.3.14*/
            fresh = instantiate (src, dest, t->tr)
            addnodeinlist (fresh)
        }
        src = p->id
        dest = t->id
        /*parend template is instantiated between the last child of the parent and
        the parent itself*/
        fresh = instantiate(src, dest, parend)
        addnodeinlist(fresh)
    }
}
end create-node
For the tree described in fig. 3.17 the templist is generated as a result of algorithm treetraversal is shown in fig. 3.18

![Diagram of tree with node IDs and templates]

Fig. 3.17

The root (id 1) is passed as parameter to the algorithm treetraversal. Traverse-tree process is invoked with t having the value root. The first procedure to be called from traverse-tree is create-node with t as the parameter and first of all parstart template is instantiated with src as nodeid 1, destination as nodeid 2 and tr having value parstart. Control comes back to traverse-tree. In the next step, two before templates, first with src nodeid 2 and dest nodeid 3; second with src nodeid 3 and dest nodeid 4 are generated by create-node. As node with id=4 is the last child at this level, parend template is generated by invoking create-node with src nodeid 4 and dest nodeid 1. Each time after creating a node addnodelist is called which appends the latest node to the link list. The same process is repeated for all subtrees. Instantiate creates a node of the link list by instantiating the appropriate template as required by the tr field in parent in case of siblings, and parfinish or parstart in case of the last and the first child respectively. The structure of the templates
for various temporal relations is given in fig.3.14. The values of src, dest and tr are passed as parameters. This procedure sets the values of array trig. The array trig is 2x2 and each element is (id, event) pair. id denotes the id of source in the first column and destination in the second column.

If the value of event is 1 it implies begin and value 0 implies end. Tree traversal stops when the queue is empty and finally the list of nodes is displayed.

After algo treetraversal the templist obtained is:

\[ \text{fig.3.18} \]
The values of the fields of the nodes in the above list can be interpreted according to the node structure given in fig.3.16. For instance in case of the first node, 1 is the src node id, 2 is the destination node id and the synchronization between the source and the destination is given by the 2x 2 array. In this case synchronization is represented by one element i.e. first row and second column which is $l_{\text{beg}}$. It implies $l_{\text{beg}}$(beginning of display of node 1) is the begin trigger for node with id 2.

In this chapter we have talked about coarse-grain synchronization. The main purpose of the synchronization model is to predict presentation time for each data unit. We use the templist generated in this chapter to obtain deadline table for presentation. Our model relates events through temporal relation (tr) field included in the document structure node. The offset provides determinacy i.e. how much after or before, but if we add no quantitative information scenario will be totally indeterminate. There is no need for absolute time as the events may be represented by an event hierarchy (this event hierarchy may be independent of corresponding objects hierarchy & thus provides flexibility). Our model though instant-based(associated with event) is capable of representing intervals also because of the attribute duration.