Proceedings of International Conference
on Networking and Multimedia

Managing Temporal Knowledge Underlying Multimedia Presentations *

Timothy K. Shih, Hwei-Jen Lin, Chin-Hwa Kuo, Nancy P. Lin, Yule-Chyun Lin
Chua-Cha Wang and Szu-Jan Fu
Department of Computer Science and Information Engineering
Tamkang University
Tamsui, Taiwan
R.O.C.
email: TSHIH@CS.TKU.EDU.TW
fax: Intl. (02) 620 9749

Abstract

Many interactive authoring tools were developed. However, synchronization of a multimedia presentation is still a difficult task partially due to the constant change of a presentation design. In this paper, we use the Z notation to analyze the temporal knowledge underlying a multimedia presentation. Four temporal specification statements are proposed, which specify the relations of temporal intervals. These statements are translated to some internal temporal relations before the final representation of a presentation time frame is generated. The system is implemented with a ICON programming graphical interface, which allows a presentation designer to drag and drop presentation resources obtained from a multimedia database in the process of a presentation design. The early experience of using the system shows that it is feasible to use logic inference rules to assist the design of good multimedia presentations.

Key words: *Z* Notations, Temporal Properties, Multimedia Presentations, ICON Programming

1 Introduction

As multimedia technologies largely increase communication effectiveness between human and computers, the importance of efficient multimedia authoring tools brings the attention to both researchers and software venders. Many presentation or authoring tools were developed for presenters or artists in various fields. Some researchers developed domain specific presentations using artificial intelligence techniques. For example, COMET (COordinated Multimedia Explanation Testbed) [7, 8] uses a knowledge base and AI techniques to generate coordinated, interactive explanations with text and graphics that illustrates how to repair a military radio receiver-transmitter. We also proposed a system [3, 13, 14, 15] that uses object-oriented techniques to incorporate Expert System inference-mechanisms into multimedia presentation designs. The system supports interactive designs of presentation layouts and navigation, incorporated with an underlying inference and learning system that deduces useful outputs for intelligent presentations. In this paper, we propose some new results of our research, especially in multimedia inter-stream synchronizations. Articles related to solving multimedia synchronization problems using interval temporal relations/temporal logic includes those discussed in [2, 9, 5, 4, 11, 10]. Other researchers [12, 1, 10] use timed Petri nets. Our approach, based on the thirteen relationships between two time intervals proposed in [2], considers multimedia resource properties as important issues in order to make good multimedia presentations. Other authoring systems discuss synchronization of multimedia streams can be found in [17, 6].

The purpose of our presentation generator is to reduce a designer's load as much as possible while still allowing the designer to make high quality presentations. We support the stepwise refinement of a presentation design. As long as the user knows the relation between two resources, he/she can start the initial design. Other resources are added one by one to the initial design. We also provide an efficient mechanism for the user to precisely control the synchronization points by means of a "synchronizes" specification statement. Multimedia presentations can be designed in our proposed specification language. Or, for the convenience of the user, we also develop a ICON programming graphical user interface. A multimedia presentation specification, according to our system, contains three parts:

- *Resource Specification* carries information of multimedia resources to be used in the presentation, which is

obtained from our multimedia resource database via a resource browser.

- *Temporal Specification* describes the temporal relations among resources, which is specified in a predicate format.
- *Spatial Specification* provides the layout of a presentation.

Our presentation generator takes as input the above specifications, uses logic inference rules defined in our system, and generates presentation implementation information for a *presentation frame* [3, 14] (e.g., screen layouts, presentation schedule, or possible error diagnosis). A presentation contains a number of presentation frames. These frames activate each other via message passing. The navigation of a presentation is based on the user's interaction which introduces messages. The interaction, as well as the temporal synchronization requirements and the spatial layouts of multimedia resources, are the three important elements construct a multimedia presentation.

This paper is organized as the following. Section 2 introduces some temporal operators and functions in order for us to represent the temporal information of a multimedia resource within a presentation. Section 3 presents our specification statements and two translation functions in Z for the generation of multimedia presentation schedule. Finally, section 4 highlights our contributions and points out some difficult problems as our further research direction. Due to the limitation of space, we omit the introduction of the Z notation/language [16].

2 Temporal Notations

In order to represent the temporal information of multimedia presentation resources, a representation of time model is necessary. The time model of our presentations is discrete. That is, a presentation consists of a number of continuous time intervals (or cycles). Next, we need a representation to embed presentation resources within these time intervals. Thus, a number of temporal operators and functions are defined in our inference system for the representation of resource temporal information. The concatenation operator, "^", is to connect two sequential resource streams. The silent operator, "*", while applying to a number, denotes a silent stream of many cycles. For instance, "*10" is a stream of no action which lasts 10 cycles. Note that a silent stream can be concatenated with another stream using the concatenation operator. The extension operator, "~", extends a resource stream, according to the *synchronization extent* of that resource (e.g., repeat, keep the last frame, no extension, etc.). Synchronization extent of multimedia resources describes how the resources should be presented after its regular ending. We discuss this concept in section 3. The truncate operator, associated with a number, can be used in two ways. "r 10!" says that resource r is played for the first 10 cycles only. And "!10 r" denotes r is played after cutting the first 10 cycles. These two operators can be applied together to a resource if the total cutting time is smaller than or equal to the duration of that resource. Otherwise, the presentation of that resource is omitted. The concurrent function, "$", is an overloaded function accepts one or more parameters. All resources with their names specified as parameters of the concurrent function start concurrently. The sequential function, denoted by "-", is also an overloaded function. The resources specified as parameters of a sequential function are presented one by one. There is no semantic difference between the sequential function and the concatenation operator. However, sequential functions serve as the principal functors of the final representation of a multimedia presentation. The concatenation operators are used in the intermediate process, or used as parts of the final representation of a presentation. The last function is the identical function "&". This function is similar to the concurrent function with a further restriction indicates that all resources end at the same time as well.

A multimedia resource, when displayed or played by a multimedia computer, produces a serious of presentation values. These values are pieces of video clips, sound effects within some time intervals, or text displayed for a duration of time. A presentation thus contains one or more tracks, which carries temporal presentation values. In the discussion, we omit the detail discussion of an individual presentation value, denoted by a *given type*, V, in the following definition [16]. And we denote a serious of presentation values as TV[1]:

$$[V]$$
$$TV == \mathrm{seq}_1\ V$$

To ease our discussion, we define a selection operator "." to extract a property out of a resource specification statement. We use an axiomatic description [16] to define this operator:

$$\begin{array}{|l} r : R \\ t_r : T \\ n_r : N \\ te_r : TE \\ se_r : SE \\ d_r : D \\ rs_r : RS \\ rd_r : RD \\ \hline r = \langle t_r, n_r, te_r, se_r, d_r, rs_r, rd_r \rangle \Leftrightarrow \\ \quad r.t = t_r \wedge r.n = n_r \wedge r.te = te_r \wedge r.se = se_r \wedge \\ \quad r.d = d_r \wedge r.rs = rs_r \wedge r.rd = rd_r \end{array}$$

[1] In Z, $\mathrm{seq}_1\ X$ denotes a non-empty sequence of objects of type X.

The duration function *dur*, when applied to a resource specification statement, returns the temporal extent of the statement:

$$dur : R \rightarrow \mathbb{N}$$
$$\forall r : R;\ te_r : \mathbb{N} \bullet dur(r) = r.te$$

The temporal operators and functions are defined in the following *generic description* [16], with some of the restrictions applied to them:

$$\_\hat{}\_ : TV \times TV \rightarrow TV$$
$$*\_ : \mathbb{N}_1 \rightarrow TV$$
$$\_\hat{}\_ : TV \times \mathbb{N}_1 \rightarrow TV$$
$$\_\ \_! : TV \times \mathbb{N} \rightarrow TV$$
$$!\_\ \_ : \mathbb{N} \times TV \rightarrow TV$$
$$\$\_ : \mathrm{seq}_1\ TV$$
$$-\_ : \mathrm{seq}_1\ TV$$
$$\&\_ : \mathrm{seq}_1\ TV$$

$$\forall r1, r2 : R \bullet dur(r1\hat{}r2) = dur(r1) + dur(r2)$$
$$\forall n : \mathbb{N}_1 \bullet dur(*n) = n$$
$$\forall r : R;\ n : \mathbb{N}_1 \bullet dur(r\tilde{}n) = dur(r) + n$$
$$\forall r : R;\ n : \mathbb{N}_1 \bullet dur(r\ n!) = n$$
$$\forall r : R;\ n : \mathbb{N}_1 \bullet dur(!n\ r) = dur(r) - n$$
$$\forall r1, r2, r3 : R;\ n1, n2 : \mathbb{N}_1 \bullet$$
$$\$\langle r1, r2\rangle \wedge \$\langle r1, r3\rangle \Leftrightarrow \$\langle r1, r2, r3\rangle \wedge$$
$$*n1\hat{}*n2 \Leftrightarrow *(n1 + n2) \wedge$$
$$r1\hat{}r2 \wedge r2\hat{}r3 \Leftrightarrow r1\hat{}r2\hat{}r3 \wedge$$
$$\$\langle r1, r2\rangle \wedge r1.te = r2.te \Leftrightarrow \&\langle r1, r2\rangle \wedge$$
$$\&\langle r1, r2\rangle \wedge r2\hat{}r3 \Leftrightarrow r1\hat{}r3 \wedge$$
$$\&\langle r1, r2\rangle \wedge r3\hat{}r2 \Leftrightarrow r3\hat{}r1 \wedge$$
$$\$\langle r1\hat{}r2, r1\hat{}r3\rangle \Leftrightarrow r1\hat{}\$\langle r2, r3\rangle \wedge$$
$$\$\langle r1\hat{}r2, r3\hat{}r2\rangle \Leftrightarrow \$\langle r1, r3\rangle\hat{}r2$$

3 Multimedia Presentation Specifications

In this section, we propose a number of presentation specification statements as well as some inference rules for the automatic generation of multimedia presentations. The specification statements are used in our system as internal representations. They are quite difficult to be used directly by the user.

3.1 Resource Specification

To create a high quality multimedia presentation, not only a good presentation designing environment is essential, but good multimedia resources are the key reasons. Multimedia resources are recorded or captured via camera, tape recorder, or video camera, converted to their digital formats, and saved on the disk. These resource files can be reused in different presentations. A resource is associated with a number of properties. The kinds of properties we consider here are essential for presentation generations. For instance, properties related to time and space are included. Other properties, such as key words, are not included while the user is issuing the resource specification. Only those resources used in a presentation are specified in the resource specification by the resource browser. And only those attributes related to the automatic generation of presentations will be included in the resource specification statements. Each resource is given a unique name, which maps to a resource descriptor (e.g., a file name, or a database entry). Temporal endurance, indicated by an integer as the number of cycles, specifies how long does a resource last in the presentation. the reserved word ∞ represents a permanent temporal endurance. For example, a picture can last as long as possible until it is dropped from its presentation window. Synchronization extent specifies how a resource is extended if requested. Some resources may not be extended. And some resources could end with a fade out effect[2]. Detectability indicates how sensitive a resource attracts the user. Resources occupy screen space will be given their resolutions. Otherwise, a resolution of 0*0 is given (e.g., for sound, or music). Note that, we introduce a silent resource as a time slot holder which takes no action in a presentation.

Resource Properties

Type	Name	Temporal Endurance	Sync Extent
Sound	..	integer	no | repeat
Animation	..	integer	last frame
Video	resource	integer	last frame
Picture	names	∞	yes
Text	..	∞	yes
MIDI	..	integer	no | repeat
Silent	..	integer	yes

Detectability	Resolution	Resource Descriptor
s | m | l	0*0 Units	"fn.wav"
s | m | l	X*Y Units	"fn.fli"
s | m | l	X*Y Units	"fn.avi"
s | m | l	X*Y Units	"fn.bmp"
s | m | l	X*Y Units	"fn.rtf"
s | m | l	0*0 Units	"fn.mid"
none	0*0 Units	"fn.tmp"

These properties are defined as various domains in the *Z* notation. We use a *given type declaration* to define the type of ASCII strings. As indicated in the *Z* expression below, a given type is specified in between two square parentheses:

$[ASCII\_STR]$

[2]We also consider other special effects such as "rain", "tile", etc.

A resource specification statement, R, thus consists of a resource type (T), a resource name (N), a temporal endurance (TE), a set of possible synchronization extents ($\mathbb{P}\ SE$), a detectability (D), a resolution (RS), and a resource descriptor (RD). These domains are defined as *Z abbreviation definitions*, or as *Z free types* [16]:

$$
\begin{aligned}
&R == T \times N \times TE \times \mathbb{P}\ SE \times D \times RS \times RD \\
&T ::= sound \mid animation \mid video \mid picture \mid text \mid midi \\
&N == ASCII\_STR \\
&TE == \mathbb{N} \\
&SE ::= no \mid yes \mid repeat \mid last\_picture \mid fade\_out \\
&D == 1 \mid 2 \mid 3 \\
&RS == \mathbb{N} \times \mathbb{N} \\
&RD == ASCII\_STR
\end{aligned}
$$

A presentation consists of multiple streams carry multiple resources. Some streams, due to the limitation of hardware, may not be played concurrently. In some occasions, two resources are not appropriate to be played at the same time, since it is hard for a person, for instance, to watch two video simultaneously. Or, according to the every day presentation experiences, some streams are encouraged to be played concurrently. For instance, it is nice to have a MIDI music background for an animation play. Our approach is to reduce the load of the user by given these good suggestions. However, if the user strongly against our suggestion, it is still possible for he/she to change the final presentation generated. Given two type of resources in a specification, we consider the following relations:

- **mutual exclusive** (represented by an X): two resources can not be played at the same time
- **possible concurrent** (represented by a ?): two resources could be played at the same time, however, it is the decision of the user to play them simultaneously.
- **mutual inclusive** (represented by an O): two resources are encouraged to be played concurrently

The following *generic definition* [16] summarizes these relations used between two type of resources:

$$
\begin{aligned}
&\_\ \mathtt{X}\ \_ : R \times R \\
&\_\ \mathtt{O}\ \_ : R \times R \\
&\_\ ?\ \_ : R \times R \\
&\hline
&\forall\, r1, r2 : R \bullet \\
&(r1.t = sound \land r2.t = sound \Rightarrow r1\ \mathtt{X}\ r2) \lor \\
&(r1.t = sound \land r2.t = animation \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = sound \land r2.t = video \Rightarrow r1\ ?\ r2 \lor r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = sound \land r2.t = picture \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = sound \land r2.t = text \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = sound \land r2.t = midi \Rightarrow r1\ ?\ r2 \lor r1\ \mathtt{X}\ r2) \lor \\
&(r1.t = animation \land r2.t = animation \Rightarrow \\
&\quad r1\ ?\ r2 \lor r1\ \mathtt{X}\ r2) \lor \\
&(r1.t = animation \land r2.t = video \Rightarrow \\
&\quad r1\ ?\ r2 \lor r1\ \mathtt{X}\ r2) \lor \\
&(r1.t = animation \land r2.t = picture \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = animation \land r2.t = text \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = animation \land r2.t = midi \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = video \land r2.t = video \Rightarrow r1\ \mathtt{X}\ r2) \lor \\
&(r1.t = video \land r2.t = picture \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = video \land r2.t = text \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = video \land r2.t = midi \Rightarrow r1\ ?\ r2 \lor r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = picture \land r2.t = picture \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = picture \land r2.t = text \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = picture \land r2.t = midi \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = text \land r2.t = text \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = text \land r2.t = midi \Rightarrow r1\ \mathtt{O}\ r2) \lor \\
&(r1.t = midi \land r2.t = midi \Rightarrow r1\ \mathtt{X}\ r2)
\end{aligned}
$$

Note that, in some situation we suggest more than one relations (e.g., $r1\ ?\ r2 \lor r1\ \mathtt{O}\ r2$). In this case, the user is asked to provide the decision first. If a decision is not given, the second relation (i.e., O) is used.

When two types of resources are presented in a same interval, if the hardware is able to carry out only one, we need to decide which resource to play. We consider different type of resources have different presentation priorities. Generally, dynamic resources, such as sound and video, have higher priorities since they can easily catch the attention of the user. Also, resource detectability (i.e., the degree of the resource attract a listener) is considered in presentation priority calculations. Generally, if a resource has a high detectability, it should not be omitted from the presentation. The use of priority is to assist the inference rules to make a better presentation. The following generic definition gives the priorities of resource types. The higher the number, the higher the priority.

$$
\begin{aligned}
&pri : R \rightarrow \mathbb{N} \\
&\hline
&\forall\, r : R \bullet \\
&(r.t = video \Leftrightarrow pri(r) = 2) \lor \\
&(r.t = animation \Leftrightarrow pri(r) = 2) \lor \\
&(r.t = sound \Leftrightarrow pri(r) = 1) \lor \\
&(r.t = midi \Leftrightarrow pri(r) = 1) \lor \\
&(r.t = picture \Leftrightarrow pri(r) = 0) \lor \\
&(r.t = text \Leftrightarrow pri(r) = 0) \lor \\
&(r.t = silent \Leftrightarrow pri(r) = 0)
\end{aligned}
$$

The detectability of a resource, specified in the resource database by the user, can be high, medium, or low, represented by 3, 2, or 1, respectively. Given two resources, $r1$ and $r2$, the priority checking expression (denoted by $r1 >> r2$) is decided by the following:

$$
\begin{array}{|l}
\hline
\_ >> \_ : R \times R \\
\hline
\forall r1, r2 : R \bullet \\
(r1.d > r2.d \Rightarrow r1 >> r2) \vee \\
(r1.d = r2.d \wedge pri(r1) \geq pri(r2) \Rightarrow r1 >> r2) \vee \\
(r1.d = r2.d \wedge pri(r1) < pri(r2) \Rightarrow r2 >> r1) \vee \\
(r1.d < r2.d \Rightarrow r2 >> r1) \\
\hline
\end{array}
$$

Resource Specifications are given in a Prolog predicate format containing seven parameters. Note that, some restrictions may be applied to resource specifications. For instance, there is no screen resolution for sound and MIDI music[3]. Moreover, the temporal endurance for pictures and text files, theoretically, should be infinite. However, we also allow the user to specify a finite temporal endurance for a picture or text. The notion of screen resolution is important. Not only it decides the screen layouts, the resolution information also works with the temporal information in that any screen region can not be occupied by two or more resources at the same time for our current system[4].

3.2 Temporal Specification

The research discussed in [2] proposes thirteen types of relations between two temporal intervals. The thirteen relations cover all possible situations. However, multimedia presentation synchronization needs precise timing information of resources. Some modifications to the relations are necessary in order for the system to achieve synchronization. Some statements are given additional arguments to explicitly indicate a synchronization point. The following table shows our revised relations:

Revised Interval Temporal Relations

- always(r1, n): Always present r1 for n cycles.
- meets(r1, r2): r2 is presented right after r1 finishes.
- before(r1, r2, n): r2 is presented n cycles after r1 finishes.
- starts(r1, r2): r1 and r2 are synchronized at the beginning.
- finishes(r1, r2): r1 and r2 are synchronized at the end.
- overlaps(r1, r2, n): r1 overlaps r2, r1 starts first, r2 starts n cycles after r1 starts, r1 ends before r2.
- embraces(r1, r2, n): r1 embraces r2, r1 starts first, r2 starts n cycles after r1 starts, r2 ends before r1.
- equal(r1, r2): r1 and r2 carry the same duration concurrently.
- simultaneous(r1, n1, r2, n2): The n1-th cycle of r1 and the n2-th cycle of r2 happen at the same time.

Our purpose is to reduce the load of the user by means of automation. We try to make specification statements as general as possible while still maintaining all possible temporal relations between two multimedia resources. Considering our revised relations, it is possible to combine some of them by adding extra parameters. For example, by introducing a delay parameter, we can combine the "*meets*" and the "*before*" relations. If the delay parameter in the "*sequential*$^+$" specification is zero, the specification indicates a "*meets*" relation. Otherwise, it indicates a "*before*" relation. Note that, all timing parameters in our specifications are non-negative. The following are the four specification statements used in our system:

Temporal Specification Statements

- always(r1, n): The system always presents r1 for n cycles. This statement applies to picture or text resources only.
- sequential(r1, r2, n): r2 is presented n cycles after r1 finishes.
- intersects(r1, r2, n): r1 and r2 intersects each other. r1 starts first, and r2 starts after n cycles.
- synchronizes(r1, n1, r2, n2): The n1-th cycle of r1 and the n2-th cycle of r2 are synchronized.

Our specification statements are the generalized version of the revised relations based on Allen's results. Since the timing arguments are provided by the user and the temporal endurance of resources can be obtained from the database, some equivalence relations hold between our statements and the revised relations. These equivalence relations are discussed in the Ψ function (to be discussed). For now, we define the signatures of these specification statements and revised relations below:

[3] Even sound can be recorded in different sampling frequency and 8-bit or 16-bit option, we are not considering the difference in the generated presentation, as long as the underlying hardware works properly.

[4] Overlap regions will be allowed in our next version of software.

$$
\begin{array}{|l}
always^{+} : R \times \mathbb{N} \\
sequential^{+} : R \times R \times \mathbb{N} \\
intersects^{+} : R \times R \times \mathbb{N} \\
synchronizes^{+} : R \times \mathbb{N} \times R \times \mathbb{N} \\
always : R \times \mathbb{N} \\
meets : R \times R \\
before : R \times R \times \mathbb{N} \\
starts : R \times R \\
finishes : R \times R \\
overlaps : R \times R \times \mathbb{N} \\
embraces : R \times R \times \mathbb{N} \\
equal : R \times R \\
simultaneous : R \times \mathbb{N} \times R \times \mathbb{N} \\
\hline
\end{array}
$$

$$
\begin{array}{l}
TSPEC == \{\ always^{+}, sequential^{+}, \\
\quad intersects^{+}, synchronizes^{+}\ \} \\
TREL == \{\ always, meets, before, \\
\quad starts, finishes, overlaps, \\
\quad embraces, equal, simultaneous\ \}
\end{array}
$$

$$
\begin{array}{|l}
\Psi : TSPEC \rightarrow TREL \\
r, r1, r2 : R \\
n, n1, n2 : \mathbb{N} \\
\hline
\forall s : TSPEC;\ r : TREL \bullet \Psi(s) = r \Leftrightarrow \\
s = always^{+}(r, n) \Leftrightarrow \\
\quad always(r, n) \\
s = sequential^{+}(r1, r2, n) \wedge n = 0 \Leftrightarrow \\
\quad r = meets(r1, r2) \\
s = sequential^{+}(r1, r2, n) \wedge n > 0 \Leftrightarrow \\
\quad r = before(r1, r2, n) \\
s = intersects^{+}(r1, r2, n) \wedge n = 0 \wedge \\
\quad r1.te = r2.te \Leftrightarrow r = equal(r1, r2) \\
s = intersects^{+}(r1, r2, n) \wedge n = 0 \wedge \\
\quad r1.te \neq r2.te \Leftrightarrow r = starts(r1, r2) \\
s = intersects^{+}(r1, r2, n) \wedge n > 0 \wedge \\
\quad r1.te = r2.te \Leftrightarrow r = overlaps(r1, r2, n) \\
s = intersects^{+}(r1, r2, n) \wedge n > 0 \wedge \\
\quad r1.te \neq r2.te \wedge r1.te < n + r2.te \Leftrightarrow \\
\quad\quad r = overlaps(r1, r2, n) \\
s = intersects^{+}(r1, r2, n) \wedge n > 0 \wedge \\
\quad r1.te \neq r2.te \wedge r1.te = n + r2.te \Leftrightarrow \\
\quad\quad r = finishes(r1, r2) \\
s = intersects^{+}(r1, r2, n) \wedge n > 0 \wedge \\
\quad r1.te \neq r2.te \wedge r1.te > n + r2.te \Leftrightarrow \\
\quad\quad r = embraces(r1, r2, n) \\
s = synchronizes^{+}(r1, n1, r2, n2) \wedge n1 = 0 \wedge \\
\quad n2 = 0 \Leftrightarrow r = starts(r1, r2) \\
s = synchronizes^{+}(r1, n1, r2, n2) \wedge n1 = 0 \wedge \\
\quad n2 > 0 \Leftrightarrow r = intersects(r1, r2, n2) \\
s = synchronizes^{+}(r1, n1, r2, n2) \wedge n1 > 0 \wedge \\
\quad n2 = 0 \Leftrightarrow r = synchronizes(r2, n2, r1, n1) \\
s = synchronizes^{+}(r1, n1, r2, n2) \wedge n1 > 0 \wedge \\
\quad n2 > 0 \Leftrightarrow r = simultaneous(r1, n1, r2, n2)
\end{array}
$$

We use two abbreviation definitions to denote the domain of the temporal specification statements ($TSPEC$) and the domain of the revised temporal relations ($TREL$). In order to generate temporal implementation from temporal specification, we have a number of inference rules defined in our system to carry out the derivation. Two translation functions are defined. The Ψ function takes as input a temporal specification and returns a temporal relation. The Υ function takes the temporal relation produced by Ψ, and generates a serious of presentation values (in the domain of TV).

$$
\begin{array}{|l}
\Upsilon : TREL \rightarrow TV \\
\mathrm{n} : \mathbb{N} \\
\hline
\forall r : TREL;\ tv : TV \bullet \\
\quad \Upsilon(r) = tv \Leftrightarrow \\
\quad\quad r = \mathrm{TREL(n)} \Rightarrow \\
\quad\quad\quad tv = \mathrm{TEXP(n)}
\end{array}
$$

Note that, in the definition of the Υ function, we use a number of if-then-else like inference rules. In the inference rules, we use the temporal operators and functions discussed in section 2. The following table gives these inference rules:

The Specification Inference Rules

```
Syntax Format: TREL(n) ==> TEXP(n)
always(r1, n) ==>
  if r1.TE >= n
    then
      -(r1 n!)
    else
      -(r1~n-r1.TE)
meets(r1, r2) ==>
  -(r1, r2)
before(r1, r2, n) ==>
  -(r1, _n, r2)
starts(r1,r2) ==>
  if r1 X r2
    then
      declare_error(mutual_exclusive)
    else
      if r1.TE > r2.TE
        then
          if r1 >> r2
            then
              if r1 0 r2
                then $(r1, r2~r1.TE-r2.TE)
                else $(r1, r2 ^ _r1.TE-r2.TE)
            else
              $(r1 r2.TE!, r2)
        else
          if r1 >> r2
            then
              $(r1, r2 r1.TE!)
            else
              if r1 0 r2
                then $(r1~r2.TE-r1.TE, r2)
```

```
                else $(r1 ^ _r2.TE-r1.TE, r2)
etc. ...
```

A specification shown on the left side of the table is translated, according to the rules specified on the right, to some intermediate representations. These intermediate representations are translated again to the final form of presentation using some implementation inference rules.

Note that, it is possible for a rule to generate two or more intermediate forms from a single specification statement. In this case, we use the "`&&`" operator to represent the situation. Also, alternatives are possible, which is denoted by the "`||`" operator in the inference rules. The `declare_error` function raises exceptions depends on the exception signals (i.e., `mutual_exclusive`, `possible_concurrent`, or `mutual_inclusive`) received. In the exceptional case, the user has to modify his/her presentation specifications to avoid the abnormal situation. Finally, we define a *schema* [16] shows the use of the Ψ and the Υ functions:

Translate

$temporal\_specifications? : \mathbb{P}\ TSPEC$
$temporal\_relations : \mathbb{P}\ TREL$
$temporal\_values! : \mathbb{P}\ TV$

$temporal\_values! =$
$\{\ tv : TV \mid \exists\ tspec : TSPEC;$
$\quad trel : TREL \bullet$
$\quad\quad tspec \in temporal\_specifications? \wedge$
$\quad\quad trel \in temporal\_relations \wedge$
$\quad\quad trel = \Psi(tspec) \wedge tv = \Upsilon(trel)\ \}$

The final temporal implementation of a presentation is a sequence of temporal operations/functions. Our system needs to take as input the sequence and carries out the actual presentation. A presentation schedule can be represented by a time chart. The X-axis represents time cycles, and the Y-axis represents resource streams. Two types of assertions, the *X-assertion* and the *Y-assertion*, are used in constructing the presentation time chart. An X-assertion extends the duration of a stream; a Y-assertion adds a new stream to the presentation.

Algorithm for an X-assertion: Given a stream `S`, and `-(R1, R2)` or `-(R1, R2, R3)`, etc., where `Rn` represents a resource, the system checks for each `R` with `S`. If there exists a `Rn` in `S` such that `Rn` and `Rn+1` are parameters of the sequential function (i.e., `-(..., Rn, Rn+1, ...)`), the system adds `Rn` and its appending `Rs` to stream `S`. If there is a collision in the process, a Y-assertion occurs.

Algorithm for an Y-assertion: Given a stream `S`, and `$(R1, R2)` or `$(R1, R2, R3)`, etc., where `Rn` represents a resource, the system checks for each `R` with `S`. If there exists a `Rn` in `S` such that `Rn` and `Rn+1` are parameters of the concurrent function (i.e., `$(..., Rn, Rn+1, ...)`), the system adds new streams within which `Rn` and `Rn+1` are synchronized.

The identical function `#(..., Rn, Rn+1, ...)` is treated the same as the concurrent function. In the initial process, one or more streams are created depending on the given statement. For each sequence or concurrent statement in the inference result, the system applies one of the assertions. If a statement contains only resources do not occur in any of the existing stream in the gaint chart, the statement represents some isolated resources. The system signals an exception condition to tell the user to provide a new specification for the isolated resources to be integrated to the existing streams.

So far, we have discussed the temporal issues of a presentation. On the other side, the spatial issues of a presentation is also very important. However, using the spatial specification of a presentation to generate the presentation layout automatically is a difficult task. The process needs a lot of heuristics in the inference rules, especially for solving the region overlap problem. We leave this topic for our future research.

4 Conclusions

We propose a mechanism and a system for the automatic generation of interactive multimedia presentations. The mechanism uses interval temporal logic inference rules as a tool to achieve multimedia resource synchronization. These rules incorporate issues such as hardware limitations, properties of multimedia resources, and good presentation principles. The specification statements we proposed covers all temporal relations given in [2]. To precisely specify synchronization points, we annotate the relations with timing parameters. A ICON programming user interface and the presentation generator are developed. We use our prototype system to generate some simple presentations, such as a city tour and an introduction lecture to multimedia PCs.

However, it is very difficult to construct good inference rules to assist a presentation designer in the layout of a presentation, especially in solving the resource boundary overlap problems. In the prototype system, we provide an editor for the designer to arrange the presentation layouts. We are still seeking for good heuristics for spatial inference rules.

Another important issue of presentation generation is to allow the user to change the presentation generated. And the system can take the change from the user and synthesize a new specification. Moreover, the use of multimedia resources depends on each other. For instance, a picture showing the city of Paris and a text file describing the history of the city are generally used together. The dependency is important when reuse of resources is considered. We have also developed a multimedia resource database and a browser to support the

reuse of resources and presentations. But this topic is out of the scope of this paper.

Consequently, our contributions in the paper are: firstly, we propose four useful temporal specification statements by showing the mapping between the statements and the interval temporal relations [2]. Secondly, a number of inference rules are developed and a presentation generator is implemented. Thirdly, a ICON programming interface is designed for the convenience of the user. Using our system, the presentation designer is able to specify *what* he/she wants instead of telling the computer exactly *how* the presentation show be scheduled.

References

[1] Yahya Y. Al-Salqan, et. al., "MediaWare: On Multimedia Synchronization" in proceedings of the international conference on multimedia computing and systems, Washington DC, U.S.A., May 15–18, 1995, pp 150–157.

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

[3] Chi-Ming Chung, Timothy K. Shih, Jiung-Yao Huang, Ying-Hong Wang, and Tsu-Feng Kuo, "An Object-Oriented Approach and System for Intelligent Multimedia Presentation Designs," In proceedings of the ICMCS'95 conference, pp 278–281, 1995.

[4] Young Francis Day, et. al., "Spatio-Temporal Modeling of Viedo Data for On-Line Object-Oriented Query Processing," in proceedings of the international conference on multimedia computing and systems, Washington DC, U.S.A., May 15–18, 1995, pp 98–105.

[5] A. John M. Donaldson and Plamen L. Simeonov "Addressing Real Time with Temporal Logic in Multimedia Communications" in proceedings of the Second ISATED/ISMM international conference on Distributed Multimedia Systems and Applications, Stanford, California, U.S.A., August 7–9, 1995, pp 215–219.

[6] Seongbae Eun, Eun Suk No, Hyung Chul Kim, Hyunsoo Yoon, and Seung Ryoul Maeng, "Eventor: an Authoring System for Interactive Multimedia Applications," Multimedia Systems, Vol. 2, 1994, pp 129–140.

[7] S. K. Feiner and K. R. McKeown, "Automating the Generation of Coordinated Multimedia Explanations," In Intelligent Multimedia Interfaces, edited by Mark T. Maybury, American Association for Artificial Intelligence, pp 117–138, 1993.

[8] S. K. Feiner, et. al., "Towards Coordinated Temporal Multimedia Presentations," In Intelligent Multimedia Interfaces, edited by Mark T. Maybury, American Association for Artificial Intelligence, pp 139–147, 1993.

[9] Petra Hoepner "Synchronizing the Presentation of Multimedia Objects - ODA Extensions," ACM SIGOIS, 1991, pp 19–31.

[10] Thomas D. C. Little and Arif Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, April, pp 413–427, 1990.

[11] Thomas D. C. Little and Arif Ghafoor "Interval-Based Conceptual Models for Time-Dependent Multimedia Data," IEEE transactions on knowledge and data engineering, Vol. 5, No. 4, 1993, pp 551–563.

[12] B. Prabhakaran and S. V. Raghavan, "Synchronization models for multimedia presentation with user participation," Multimedia Systems, Vol. 2, pp 53–62, Springer-Verlag, 1994.

[13] Timothy K. Shih, "An Artificial Intelligent Approach to Multimedia Authoring," In proceedings of the second IASTED/ISMM international conference on distributed multimedia systems and applications, Stanford, California, August 7-9, pp 71–74, 1995.

[14] Timothy K. Shih, "On Making a Better Interactive Multimedia Presentation," in Proceedings ofthe International Conference on Multimedia Modeling, Singapore, 1995.

[15] Timothy K. Shih, Chin-Hwa Kuo, and Kuan-Shen An, "Multimedia Presentation Designs with Database Support," in Proceedings of the NCS'95 conference, Taiwan, 1995.

[16] J. Michael Spivey, *The Z Notation: A Reference Manual.* International Series in Computer Science. Prentice Hall, 1989.

[17] Thomas Wahl, et. al., "TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia" in proceedings of the international conference on multimedia computing and systems, Washington DC, U.S.A., May 15–18, 1995, pp 274–277.