

Temporal Properties Underlying Multimedia Presentations with Z Notations

TIMOTHY K. SHIH, HUAN-CHAO KEH, YING-HONG WANG AND YING-FENG KUO*

*Department of Computer Science and Information Engineering
Tamkang University*

Tamsui, Taiwan 251, R.O.C.

E-mail: tshih@cs.tku.edu.tw

**Department of Electrical Engineering*

China Junior College of Technology and Commerce

Taipei, Taiwan 115, ROC.

E-mail: yfkuo@cc.cjctc.edu.tw

Formal specifications use mathematical notations to precisely describe **what** properties a system need to have instead of **how** these properties are implemented. The Z notation is a formal specification language widely used in Europe. In this paper, we use the Z notation to analyze the temporal knowledge underlying a multimedia presentation. We then present a system based on the four temporal specification statements that we propose in this paper. The system visualizes the design of a multimedia presentation. Our early experience in using the system shows that it is feasible to use logic inference rules to assist the design of good multimedia presentations.

Keywords: Z notations, multimedia presentations, logic inference rules

1. INTRODUCTION

As multimedia technologies largely increase communication effectiveness between human and computers, the importance of efficient multimedia authoring tools has attracted the attention to both researchers and software vendors. Many presentation and authoring tools have been developed for presenters or artists in various fields. Some researchers have 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 illustrate 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 design of presentation layouts and navigation, incorporated with an underlying inference and learning system that deduces useful outputs for intelligent presentations. In this paper, we present some new results of our research, especially in multimedia inter-stream synchronization. Articles related to solving multimedia synchronization problems using interval temporal relations/temporal logic includes those discussed in [2, 4, 5, 9-11].

Received June 11, 1996; accepted March 19, 1998.
Communicated by Shing-Tsaan Huang.

Other researchers [1, 10, 12] have used 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 producing good multimedia presentations. Other authoring systems for synchronization of multimedia streams are found in [6, 17].

The related works addressed above are mostly academic researches. On the other hand, we also looked at some commercial products related to multimedia authoring or presentation design:

1. Authorware Professional by Macromedia, Inc.
2. Multimedia Viewer by Microsoft
3. Multimedia Toolbook by Asymetrix Corporation
4. Hypermedia System by ITRI (Taiwan)
5. Action! by Macromedia, Inc.
6. Audio Visual Connection by IBM
7. Astound by Gold Disk Inc.
8. Director by Macromedia, Inc.

Authorware uses an event control flow diagram which allows the presenter to specify presentation objects and controls, which can be decomposed into several levels in a hierarchical structure. The system also provides a simple script language for calculation and data manipulation. Other systems (i.e., 2, 3, and 6 above) also provide script languages and API (application program interface) functions. Hypermedia System, Action!, and Director use a time line table which allows actions or objects to be dropped in a particular time slot. Most systems allow users to cut and paste presentation objects or actions via button clicking and drawing. Multimedia Viewer also provides a set of media editing tools. Presentation objects produced by these tools can be linked together by a script language which supports functions, data structures, and commands.

Even through the above presentation design systems provide fancy user interfaces and flexible presentation control mechanisms, have some drawbacks:

- At the beginning of a presentation design, it is possible that the designer may not have a complete idea of what the detail presentation schedule should look like. For instance, when a designer uses Director in the early stage of a design, it is difficult for him/her to precisely lay out the temporal relations among all the multimedia resources. As a result, the designer has to modify the time table many times. Each modification requires some changes in timing of other resources not directly related to the modification. This will certainly increase the burden on the designer.
- If a user uses Authorware to design a presentation, it is difficult for the user to precisely specify the synchronization point between two resources or among three or more resources. As a result, it is difficult to design, for instance, a presentation with a special sound effect which occurs at a particular time point in the presentation.

The purpose of our presentation generator is to reduce the designer's effort as much as possible while still allowing the designer to make high quality presentations. We support stepwise refinement of a presentation design. As long as the user knows the relation between two resources, he/she can start work on 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. In addition, for the convenience of the user, we have also developed an 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 are 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. Navigation in a presentation is based on user interaction, which introduces messages. This interaction, as well as the temporal synchronization requirements and the spatial layouts of multimedia resources, are the three important elements used to construct a multimedia presentation.

This paper is organized as follows. Section 2 introduces some temporal operators and functions used 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 generation of multimedia presentation schedule. In section 4, we discuss an ICON programming mechanism and interface which the user can use to design presentation easily. Section 5 gives a comparison of our system with others. Finally, section 6 highlights our contributions and points out some difficult problems which indicate our further research direction. Since to understand a formal specification specified in the *Z* notation requires some background study of the *Z* language, we present some notations of *Z* in appendix A. (Due to the limitation of space, we omit the introduction of the *Z* notation language.)

2. TEMPORAL NOTATIONS

In order to represent the temporal information of multimedia presentation resources, a time model representation 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 used to connect two sequential resource streams. The silent operator, “*”, applied 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 the regular ending. We discuss this concept in section 3. The truncate operator, associated with a number, can be used in two ways. “ $r\ 10!$ ” means that resource r is played for the first “ $!10\ r$ ” cycles only. “ $!10\ r$ ” denotes that 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, presentation of that resource is omitted. The concurrent function, “ $\$$ ”, is an overloaded function. This 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 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 which indicates that all resources end at the same time.

A multimedia resource, when displayed or played by a multimedia computer, produces a series 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 carry temporal presentation values. In the following, we omit detailed discussion of an individual presentation value, denoted by a given type, V , in the following definition [16]. We denote a series of presentation values as TV ¹:

$$[V] \\ TV == \text{seq}_1 V.$$

To simplify our discussion, we define a selection operator “ \cdot ” used to extract a property from a resource specification statement. We use an axiomatic description [16] to define this operator:

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

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

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

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

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

$$\begin{array}{l} \textit{Temporal_Operators} \\ \hline \begin{array}{l} _ \wedge _ : TV \times TV \rightarrow TV \\ _ * _ : _ \rightarrow TV \\ _ \sim _ : TV \times _ \rightarrow TV \\ _ \! _ : TV \times _ \rightarrow TV \\ \! _ _ : _ \times TV \rightarrow TV \\ \$ _ : seq_1 TV \\ _ _ : seq_1 TV \\ \& _ : seq_1 TV \end{array} \\ \hline \begin{array}{l} \forall r1, r2 : R \bullet dur(r1 \wedge r2) = dur(r1) + dur(r2) \\ \forall n : _ \bullet dur(*n) = n \\ \forall r : R : n : _ \bullet dur(r \sim n) = dur(r) + n \\ \forall r : R : n : _ \bullet dur(r \! n!) = n \\ \forall r : R : n : _ \bullet dur(!n \ r) = dur(r) - n \\ \forall r1, r2, r3 : R; n1, n2 : _ \bullet \\ \quad \$\langle r1, r2 \rangle \wedge \$\langle r1, r3 \rangle \Leftrightarrow \$\langle r1, r2, r3 \rangle \wedge \\ \quad *n1 \wedge *n2 \Leftrightarrow *(n1 + n2) \wedge \\ \quad r1 \wedge r2 \wedge r2 \wedge r3 \Leftrightarrow r1 \wedge r2 \wedge r3 \wedge \\ \quad \$\langle r1, r2 \rangle \wedge r1.te = r2.te \Leftrightarrow \&\langle r1, r2 \rangle \wedge \\ \quad \&\langle r1, r2 \rangle \wedge r2 \wedge r3 \Leftrightarrow r1 \wedge r3 \wedge \\ \quad \&\langle r1, r2 \rangle \wedge r3 \wedge r2 \Leftrightarrow r3 \wedge r1 \wedge \\ \quad \$\langle r1 \wedge r2, r1 \wedge r3 \rangle \Leftrightarrow r1 \wedge \$\langle r2, r3 \rangle \wedge \\ \quad \$\langle r1 \wedge r2, r3 \wedge r2 \rangle \Leftrightarrow \$\langle r1, r3 \rangle \wedge r2 \end{array} \end{array}$$

3. MULTIMEDIA PRESENTATION SPECIFICATIONS

In this section, we propose a number of presentation specification statements as well as some inference rules for automatic generation of multimedia presentations. The specification statements are used in our system as internal representations. They are quite difficult to use directly. In section 4, we discuss how our graphical user interface is able to make the design of presentations much easier.

3.1 Resource Specification

To create a high quality multimedia presentation, not only a good presentation designing environment, but also good multimedia resources are needed. Multimedia resources are recorded or captured via camera, tape recorder, or video camera, converted to digital formats, and saved on 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 employed by the user is issuing resource specifications. Only those resources used in a presentation are specified in the resource specification by the resource browser, and only those attributes related to automatic generation of presentations are 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 a resource lasts in the presentation. The reserved word ∞ represents 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 can end with a fade out effect². Detectability indicates how sensitive a resource attracts the user. Resources which occupy screen space are assigned 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	Detectability	Resolution	Resource Descriptor
Sound	name	integer	no repeat fade out	1 2 3	0*0 Units	“fn.wav”
Animation	name	integer	last frame fade out	1 2 3	X*Y Units	“fn.fli”
Video	name	integer	last frame fade out	1 2 3	X*Y Units	“fn.avi”
Picture	name	∞	yes fade out	1 2 3	X*Y Units	fn.bmp”
Text	name	∞	yes fade out	1 2 3	X*Y Units	“fn.rtf”
MIDI	name	integer	no repeat fade out	1 2 3	0*0 Units	“fn.mid”
Silent	name	integer	yes	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 string. As indicated in the Z expression below, a given type is specified in between two square parentheses:

[ASCII_STR]

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 (SE), a detectability value (D), a resolution value (RS), and a resource descriptor (RD). These domains are defined as Z abbreviation definitions or as Z free types [16]:

$$R == T \times N \times TE \times SE \times D \times RS \times RD$$

$$T ::= \text{sound} | \text{animation} | \text{video} | \text{picture} | \text{text} | \text{midi} | \text{silent}$$

$$N == \text{ASCII_STR}$$

$$TE ==$$

²We also consider other special effects such as “rain”, “tile”, etc.

$SE ::= no | yes | repeat | last_frame | fade_out$
 $D == 1 | 2 | 3$
 $RS == \times$
 $RD == ASCII_STR.$

A presentation consists of multiple streams which carry multiple resources. Some streams, due to hardware limitations, may not be played concurrently. On some occasions, two resources should not be played at the same time, since it is hard for a person, for instance, to watch two video clips simultaneously. On the other hand, according to the every day presentation experience, some streams should be played concurrently. For instance, it is nice to have a MIDI music background for an animation. Our approach is to reduce the burden on the user by giving these good suggestions. However, if the user is strongly against our suggestion, it is still possible for him/her to change the final generated presentation. 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 user decides whether or not to play them simultaneously;
- **mutual inclusive** (represented by an 0): two resources should be played concurrently.

The following schema summarizes these relations used between two types of resources:

<i>Stream_Relations</i>
$_X_ : R \times R$
$_0_ : R \times R$
$_?_ : R \times R$
$\forall r1, r2 : R \bullet$
$(r1.t = sound \wedge r2.t = sound \Rightarrow r1 \ X \ r2) \vee$
$(r1.t = sound \wedge r2.t = animation \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = sound \wedge r2.t = video \Rightarrow r1 \ ? \ r2 \vee r1 \ 0 \ r2) \vee$
$(r1.t = sound \wedge r2.t = picture \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = sound \wedge r2.t = text \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = sound \wedge r2.t = midi \Rightarrow r1 \ ? \ r2 \vee r1 \ X \ r2) \vee$
$(r1.t = animation \wedge r2.t = animation \Rightarrow r1 \ ? \ r2 \vee r1 \ X \ r2) \vee$
$(r1.t = animation \wedge r2.t = video \Rightarrow r1 \ ? \ r2 \vee r1 \ X \ r2) \vee$
$(r1.t = animation \wedge r2.t = picture \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = animation \wedge r2.t = text \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = animation \wedge r2.t = midi \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = video \wedge r2.t = video \Rightarrow r1 \ X \ r2) \vee$
$(r1.t = video \wedge r2.t = picture \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = video \wedge r2.t = text \Rightarrow r1 \ 0 \ r2) \vee$
$(r1.t = video \wedge r2.t = midi \Rightarrow r1 \ ? \ r2 \vee r1 \ 0 \ r2) \vee$

$$\begin{array}{l}
(r1.t = picture \wedge r2.t = picture \Rightarrow r1 \ 0 \ r2) \vee \\
(r1.t = picture \wedge r2.t = text \Rightarrow r1 \ 0 \ r2) \vee \\
(r1.t = picture \wedge r2.t = midi \Rightarrow r1 \ 0 \ r2) \vee \\
(r1.t = text \wedge r2.t = text \Rightarrow r1 \ 0 \ r2) \vee \\
(r1.t = text \wedge r2.t = midi \Rightarrow r1 \ 0 \ r2) \vee \\
(r1.t = midi \wedge r2.t = midi \Rightarrow r1 \ X \ r2)
\end{array}$$

Note that, in some situations, we suggest more than one relation (e.g., $r1 \ ? \ r2 \vee r1 \ 0 \ r2$). In this case, the user is asked to provide the decision first. If a decision is not made, the second relation (i.e., 0) is used.

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

$$\begin{array}{l}
pri : R \rightarrow \\
\hline
\forall r : R \bullet \\
(r.t = video \Leftrightarrow pri(r) = 2) \vee \\
(r.t = animation \Leftrightarrow pri(r) = 2) \vee \\
(r.t = sound \Leftrightarrow pri(r) = 1) \vee \\
(r.t = midi \Leftrightarrow pri(r) = 1) \vee \\
(r.t = picture \Leftrightarrow pri(r) = 0) \vee \\
(r.t = text \Leftrightarrow pri(r) = 0) \vee \\
(r.t = silent \Leftrightarrow pri(r) = 0)
\end{array}$$

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 $r1 \gg r2$) is decided based on the following:

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

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 or MIDI music³. Moreover, 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 does it determine the screen layout, but the resolution information also works with the temporal information in that no screen region can be occupied by two or more resources at the same time in our current system⁴.

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

Relations	Meanings
$always(r1, n)$	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 burden on 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:

³Even sound can be recorded at a different sampling frequency and with 8-bit or 16-bit options; we do not consider the resulting difference in the generated presentation, as long as the underlying hardware works properly.

⁴Overlapping regions will be allowed in our next version of software.

Temporal Specification Statements

Specifications	Meanings
$always(r1, n)^+$	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 generalized versions 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 as given below:

$$\begin{aligned}
always^+ &: R \times \\
sequential^+ &: R \times R \times \\
intersects^+ &: R \times R \times \\
synchronizes^+ &: R \times \quad \times R \times \\
always &: R \times \\
meets &: R \times R \\
before &: R \times R \times \\
starts &: R \times R \\
finishes &: R \times R \\
overlaps &: R \times R \times \\
embraces &: R \times R \times \\
equal &: R \times R \\
simultaneous &: R \times \quad \times R \times.
\end{aligned}$$

We use two abbreviation definitions to denote the domain of the temporal specification statements ($TSPEC$) and the domain of the revised temporal relations ($TREL$):

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

In order to generate temporal implementation from temporal specification, we define a number of inference rules in our system to carry out derivation. Two translation functions are defined. The Ψ function takes as input a temporal specification and returns a temporal relation. The Y function takes the temporal relation produced by Ψ , and generates a series of presentation values (in the domain of TV):

$$\left| \begin{array}{l}
\Psi : TSPEC \rightarrow TREL \\
r, r1, r2 : R \\
n, n1, n2 : \\
\hline
\forall_s : TSPEC : r : TREL \bullet \psi(s) = r \Leftrightarrow
\end{array} \right.$$

$$\begin{aligned}
 s &= \text{always}^+(r, n) \Leftrightarrow \text{always}(r, n) \\
 s &= \text{sequential}^+(r1, r2, n) \wedge n = 0 \Leftrightarrow r = \text{meets}(r1, r2) \\
 s &= \text{sequential}^+(r1, r2, n) \wedge n > 0 \Leftrightarrow r = \text{before}(r1, r2, n) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n = 0 \wedge r1.te = r2.te \Leftrightarrow r = \text{equal}(r1, r2) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n = 0 \wedge r1.te \neq r2.te \Leftrightarrow r = \text{starts}(r1, r2) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n > 0 \wedge r1.te = r2.te \Leftrightarrow r = \text{overlaps}(r1, r2, n) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n > 0 \wedge r1.te \neq r2.te \wedge r1.te < n + r2.te \Leftrightarrow r = \text{overlaps}(r1, r2, n) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n > 0 \wedge r1.te \neq r2.te \wedge r1.te = n + r2.te \Leftrightarrow r = \text{finishes}(r1, r2) \\
 s &= \text{intersects}^+(r1, r2, n) \wedge n > 0 \wedge r1.te \neq r2.te \wedge r1.te > n + r2.te \Leftrightarrow r = \text{embraces}(r1, r2, n) \\
 s &= \text{synchronizes}^+(r1, n1, r2, n2) \wedge n1 = 0 \wedge n2 = 0 \Leftrightarrow r = \text{starts}(r1, r2) \\
 s &= \text{synchronizes}^+(r1, n1, r2, n2) \wedge n1 = 0 \wedge n2 > 0 \Leftrightarrow r = \text{intersects}(r1, r2, n2) \\
 s &= \text{synchronizes}^+(r1, n1, r2, n2) \wedge n1 > 0 \wedge n2 = 0 \Leftrightarrow r = \text{synchronizes}(r2, n2, r1, n1) \\
 s &= \text{synchronizes}^+(r1, n1, r2, n2) \wedge n1 > 0 \wedge n2 > 0 \Leftrightarrow r = \text{simultaneous}(r1, n1, r2, n2)
 \end{aligned}$$

$$\frac{\Upsilon : TREL \rightarrow TV}{n:} \quad \frac{}{\forall r : TREL; tv : TV \bullet \Upsilon(r) = tv \Leftrightarrow r = TREL(n) \Rightarrow tv = TEXP(n)}$$

Note that, in the definition of the γ 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:

n	TREL(n)	TEXP(n)
1	always(r1, n)	if r1.te >= n then -(r1 n!) else -(r1 ~n-r1.te)
2	meets(r1, r2)	-(r1, r2)
3	before(r1, r2, n)	-(r1, *n, r2)
4	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)

5 finishes(r1, r2)    if r1 X r2
                    then
                      declare_error(mutual_exclusive)
                    else
                      if r1.te>r2.te
                      then $(r1, *r1.te-r2.te^r2)
                        || $(!r1.te-r2.te r1, r2)
                      else $(*r2.te-r1.te^r1, r2)
                        || $(r1, !r2.te-r1.te r2)

6 overlap(r1, r2, n)  if r1 X r2
                    then
                      if r1>>r2
                      then meets(r1, !r1.te-nr2)
                      else meets(r1n!, r2)
                    else
                      if r1>>r2
                      then
                        starts(r1, *n^r2)
                        || finishes(r1, !r1.te-n r2)
                      else
                        if r1 0 r2
                        then
                          finishes(r1~r2.te+n-r1.te,r2)
                        else
                          starts(!nr1, r2)
                          || finishes(r1^*r2.te+n-r1.te,r2)

7 embrace(r1, r2, n)  if r1 X r2
                    then
                      if r1>>r2
                      then
                        starts(!nr1, r2)
                        || finishes(r1 n+r2.te!, r2)
                      else
                        meets(r1n!, r2)
                        && meets(r2, !r1.te-n-r2.te r1)
                    else
                      if r1>>r2

```

	<pre> then if r10 r2 then finishes(r1, r2 ~ r1.te-n-r2.te) else starts(r1, *n^r2) finishes(r1, r2^ *r1.te-n-r2.te) else starts(!nr1, r2) finishes(r1 n+r2.te!, r2) </pre>
8	<pre> equal(r1, r2) if r1 X r2 then declare_error(mutual_exclusive) else \$(r1,r2) </pre>
9	<pre> simultaneous(r1,n1,r2,n2) if r1 X r2 then declare_error(mutual_exclusive) else if n1>n2 then starts(r1, *n1-n2^r2) else starts(*n2-n1^r1, r2) </pre>

A specification shown on the left side of the table is translated, according to the rules specified on the rihgt, into some intermediate representations. These intermediate representations are translated again into 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 form a single specification statement. In this case, we use the “&&” operator to represent this situation. Also, alternatives are possible, which situation is denoted by the “||” operator in the inference rules. The declare_error function raises exceptions, depending on the exception signals (i.e., mutual_exclusive, possible_concuurent, or mutual_inclusive) received. In the exception case, the user has to modify his/her presentation specifications to avoid an abnormal situation. Finally, we define a *schema* [16] showing the use of the Ψ and the Υ functions:

$\begin{array}{l} \text{---Translate---} \\ \text{temporal_specifications? : } TSPEC \\ \text{temporal_relations : } TREL \\ \text{temporal_values! : } TV \\ \hline \text{temporal_values!} = \{tv : TV \mid \exists tspec : TSPEC : trel : TREL \bullet \\ \quad tspec \in \text{temporal_specifications?} \wedge trel \in \text{temporal_relations} \wedge \\ \quad trel = \Psi(tspec) \wedge tv = \Upsilon(trel)\} \end{array}$

The final temporal implementation of a presentation is a sequence of temporal operations/functions. Our system needs to take as input the sequence and produce 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-assertions* and the *Y-assertions*, 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 an 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 appended R s 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 an 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 in the same way 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 which do not occur in any of the existing streams in the time 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 into the existing streams.

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

4. VISUALIZATION OF THE SYSTEM

A presentation specification is a set of temporal relations among multimedia resources with spatial information for those resources. Even presentation specifications have their own representations and intermediate forms, so it is quite difficult for a presenter who is not familiar with computer programming to design a presentation using presentation specification statements. For the convenience of the user, we have developed an ICON programming technique as well as a graphical user interface.

When a presentation designer is about to design his/her presentation, after designing the presentation topics and a rough schedule, the designer will collect resources, convert them to some digital forms, and store these resources in our multimedia database [13]. Fig. 1 shows a resource browser user interface which allows the presenter to insert/delete resources and to retrieve multimedia resources according to properties given in separate text or list boxes. When these properties are decided, the user is able to push the **Refresh** button which causes a number of resource ICONs to appear in the Selection Media subwindow. These resources can be previewed before the user pushes the **Use** button to add these resources to a presentation. These resources, when passed to our presentation generator, are represented in the resource specification format discussed in section 3.1.



Fig. 1. The multimedia resource browser.

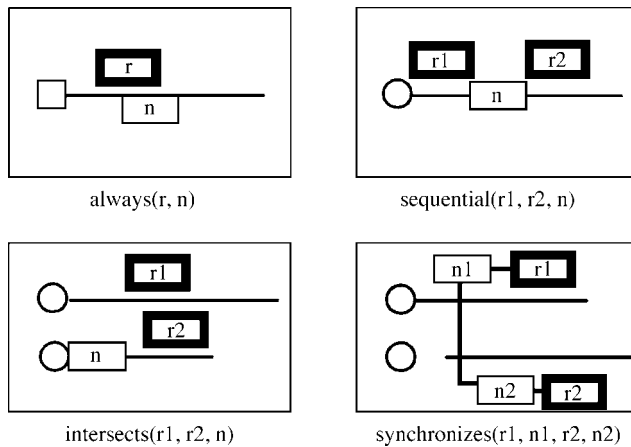


Fig. 2. Temporal specification ICONs.

Next, the presentation designer has to use our ICON programming technique to design the schedule of the presentation. We have four temporal ICONs for the four temporal specification statements discussed in section 3.2. In Fig. 2, we show four ICONs. A box with four surrounding thick lines in an ICON represents a place holder for a multimedia resource

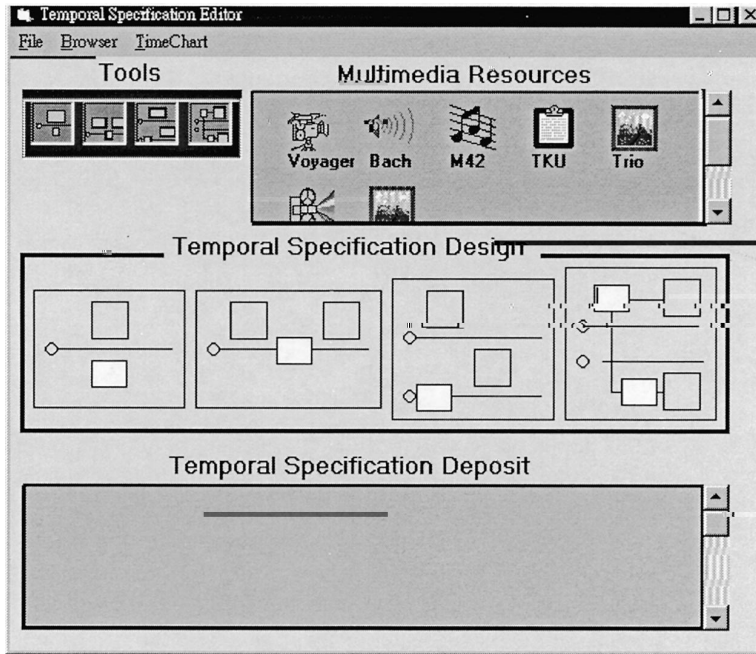


Fig. 3. The temporal specification editor.

(indicated by its name “r”, “r1”, or “r2”). A box with thin lines is used by the user to assign a timing parameter (shown as “n”, “n1”, or “n2”). After the user selects the resources that he/she plans to use from our multimedia resource browser, shown in Fig.3, when the user



Fig. 4. The spatial specification editor.

resources. Our system does not provide templates or wizards which ease presentation design or guide a presenter through the designing steps. Our system does not come with a set of resource editing tools, such as a bitmapped graphic editor, video editor, or sound editor. The major contribution of the system is in its use of inference rules based on temporal relations for automatic scheduling, which is a new approach to multimedia presentation design.

Comparison of Authoring Systems

Authoring Systems	Layout Design	Automatic Scheduling	Template or Wizard	Database Support	Resource Editing
Authorware	interactive	no	yes	yes	no
Director	interactive	no	no	yes	yes
Viewer	prog. ctrl.	no	no	no	yes
Toolbook	interactive	no	yes	no	no
Hypermedia	interactive	no	no	yes	no
Action!	interactive	no	yes	no	yes
Our System	interactive	yes	no	yes	no

6. CONCLUSIONS

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

The development of our prototype system took about one and a half years. Two Master students implemented the system and contributed ideas in their dissertations. The interface was constructed using *Visual Basic* running under Windows 95. About 5000 lines of programs were used in the interface part. On the other hand, the multimedia presentation generator is implemented in *VisualC++* (about 6000 lines). We used DLL as the linkage interface between the two parts of our prototype system. In the process of development, we found that formal specification could precisely describe our temporal rules. We used this advantage to avoid tedious discussion and mistakes. Our system proves that it is possible to include realistic considerations in a formal specification.

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

Another important task related to presentation generation is to allow the user to change the generated presentation. The system can then implement the change designated by the user and synthesize a new specification. Moreover, multimedia resources depend on each other. For instance, a picture showing the skyline of Paris and a text file describing the history of the city would generally be used together. This interdependency 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. However this topic is beyond the scope of this paper.

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

ACKNOWLEDGEMENT

I would like to thank Mr. Steven K. C. Lo and Mr. F. Y. Jeng for their implementation of the system. The prototype system could not have been successfully constructed without their hard work.

APPENDIX A. THE Z NOTATIONS

The Z notation [16] is a language for expressing formal specifications of systems. It is based on typed set theory, coupled with a structuring mechanism (i.e., schema calculus is one of its key features). A schema introduces a named collection of variables and relationships among variables that are specified by axiom definitions. Schemas are used to describe both the static aspects of a system (e.g., the structure of a program) and the dynamic aspects (e.g., the execution). Schemas can be generic, thus, polymorphic functions can be defined. Every variable introduced in a Z specification is assigned a type. These types can be given set names or can be constructed by type constructors (e.g., tuple, schema product, or the power set constructors). Free type definitions⁵ as a short mechanism to introduce new types in Z.} add nothing to the power of the Z language but ease the definition of recursive objects. Free type definitions can be translated into other Z constructs. An abbreviation definition using the symbol “==” introduces a new global constant. The identifier on the left becomes a global constant, and its value is given by the expression on the right. An axiomatic description introduces one or more global variables and optionally specifies a constraint on their values. In the discussion that follows, we will introduce concepts and some syntax of the Z notation where necessary.

The reason we take a formal specification approach is that formal specifications use mathematical notation to precisely describe **what** properties a system needs to have, without concerning very much **how** these properties are implemented. This approach

⁵Free type definitions are discussed in [16] as a short mechanism used to introduce new types in Z.

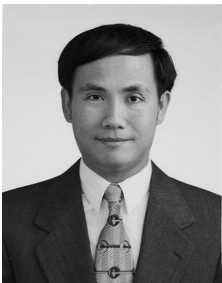
enables us to avoid tedious our discussion of the system. The Z notation is a formal specification language widely used in Europe. The notation we use here follows the standard given in [16]. We find that, by using the mathematical tool-kit provided in [16], we can easily describe the formal specification of our multimedia database. Some notations of Z are described below:

$\neg P$	[Not P]
$P \wedge Q$	[P and Q]
$P \vee Q$	[P or Q]
$P \Rightarrow Q$	[P implies Q]
$P \Leftrightarrow Q$	[P if and only if Q]
$\forall x : T \mid P \bullet Q$	[All x of type T satisfying P also satisfy Q]
$\forall x : T \bullet Q$	[All x of type T satisfy Q]
$\exists x : T \mid P \bullet Q$	[Some x of type T satisfies both P and Q]
$\exists_1 x : T \mid P \bullet Q$	[Exactly one x of type T satisfies both P and Q]
$x \in S$	[x is a member of S]
$S \subseteq T$	[S is a subset of T]
	[The empty set]
$\{x_1, \dots, x_n\}$	[The set containing exactly x_1, \dots, x_n]
$\{x : T \mid P\}$	[The set containing those x of type T which satisfy P]
$\{x : T \mid P \bullet t\}$	[The set of values of t for those x of type T satisfy P]
$\mu x : T \mid P$	[The unique x of type T which satisfies P]
$\mu x : T \mid P \bullet t$	[The value of t for that unique x of type T satisfying P]
(x_1, \dots, x_n)	[Ordered n -tuple]
$S_1 \times \dots \times S_n$	[Cartesian product]
S	[The set of all subsets of S]
\mathcal{S}	[The set of all finite subsets of S]
${}_1\mathcal{S}$	[The set of all non-empty finite subset of S]
$S \cap T$	[Intersection of S and T]
$S \cup T$	[Union of S and T]
$S \setminus T$	[Set difference]
$S = T$	[Set equivalence]
$\#S$	[Size of finite set S]
$disjoint\langle S_1, \dots, S_n \rangle$	[Sets S_1, \dots, S_n are disjoint]
$\langle S_1, \dots, S_n \rangle_{partitionS}$	[Sets S_1, \dots, S_n partitions set S]
	[Integers]
	[The natural numbers, $\{0, 1, 2, \dots\}$]
	[The positive integers, $\{1, 2, \dots\}$]
${}_1$	[Selection]
$S..x$	[The range from m to n]
$m..n$	[Binary relation between X and Y]
$X \leftrightarrow Y$	[Maplet from x to y]
$x \mapsto y$	[Domain of R]
$dom R$	[Range of R]
$ran R$	[Composition of relations]
$R_1 R_2$	[Backward composition of relations]
$R_1 \circ R_2$	[Inverse of R]
R^{-1}	

$\text{id } S$	[Identity relation of S]
$R \ S$	[Relational image]
$S \ R$	[Domain restriction]
$R \ T$	[Range restriction]
$S \ R$	[Domain anti-restriction]
$R \ T$	[Range anti-restriction]
$X \ Y$	[Partial functions from X to Y]
$X \rightarrow Y$	[Total functions from X to Y]
$X \ \overset{\circ}{Y}$	[Finite partial functions from X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Partial injections from X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Total injections from X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Finite partial injections X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Bijections from X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Surjections from X to Y]
$X \ \overset{\circ}{\rightarrow} Y$	[Partial surjections from X to Y]
$f x, f(X)$	[Function f applied to argument x]
$\lambda x : T P \bullet t$	[Lambda-notation]
$f \oplus g$	[Functional overriding]
$\text{seq } X$	[Finite sequences over X]
$\text{seq}_1 X$	[Non-empty finite sequences over X]
$\#s$	[Length of sequence s]
$\langle \rangle$	[Empty sequence]
$\langle x_1, \dots, x_n \rangle$	[The sequence containing x_1, \dots, x_n]
$s \ t$	[Concatenation of sequences s and t]
$\text{head}(s)$	[Head of sequence s]
$\text{last}(s)$	[The last element of sequence s]
$\text{tail}(s)$	[Tail of sequence s]
$\text{front}(s)$	[Front of sequence s]
$\text{rev}(s)$	[Reverse of sequence s]

REFERENCES

1. Yahya Y. Al-Salqan, et. al., "MediaWare: On multimedia synchronization" in *Proceedings of the International Conference on Multimedia Computing and Systems*, 1995, pp. 150-157.
2. James F. Allen "Maintaining knowledge about temporal intervals," *Communications of the ACM*, Vol. 26, No. 11, 1983, pp. 832-843.
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 IEEE International Conference on Multimedia Computing and Systems*, 1995, pp. 278-281.
4. Young Francis Day, et. al., "Spatio-temporal modeling of video data for on-line object-oriented query processing," in *Proceedings of the International Conference on Multimedia Computing and Systems*, 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/*



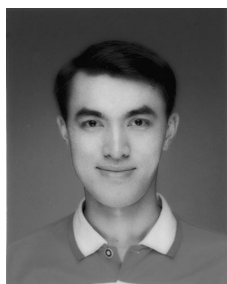
(施國琛)

Clara University in 1993. Dr. Shih has published over 130 refereed papers and served as session chairs and organization committees of many conferences. He has been invited frequently to give talks at national and international conferences and research organizations. The contact address of Dr. Shih is: Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 251, R.O.C.



and Information Engineering, Tamkang University, Tamsui, Taipei Hsien, Taiwan 251, R.O.C.

Huan-Chao Keh (葛煥昭) Dr. Keh has been an associate professor in Information Engineering and Computer Science at Tamkang University since 1991. His research interests include Object-Oriented Methodology, Parallel Architectures, Database Systems, and Multimedia Computing. Dr. Keh received his BS in engineering and MS in Computer Science from National Chung-Hsing University. He also received his Ph.D. in Computer Science from Oregon State University in 1991. The contact address for Dr. Keh is Department of Computer Science



His current research interests are Software Engineering, and Multimedia Database System.

Ying-Hong Wang (王英宏) Dr. Wang received the B.S. degree in Information Engineering from Chung-Yuan University, and the M.S. and Ph.D. degrees in Information Engineering from the TamKang University, in 1992 and 1996, respectively.

From 1988 to 1990, he worked in the Product Development Division of Institute of Information Industry (III). From 1992 to 1996, he was a lecturer in the department of information engineering of TamKang University. Beginning Fall 1996, he is an

Ying-Feng Kuo (郭穎鋒) is a Lecturer in Computer Science at the China Junior College of Technology and Commerce. He has been on the faculty of the Department of Electrical Engineering at the college since 1990. Ying-Feng Kuo has many research interests, including architecture designs, computer graphics, software engineering, and formal languages. He has published many technical papers and participated in many international conferences. Ying-Feng Kuo is currently a Ph.D. candidate in the Department of Information Engineering and

Computer Science, Tamkang University. He also received his B.S. and M.S. degrees from the same department in 1988 and 1990, respectively. The contact address for Ying-Feng Kuo is Department of Electrical Engineering, China Junior College of Technology and Commerce, Taipei, Taiwan, R.O.C.