Formal Specification of Multimedia Authoring

Timothy K. Shih, Ding-An Chiang, Huan-Chao Keh, and Chien-Chou Shis
Department of Computer Science and Information Engineering
Tamkang University
Tamsui, Taiwan
R.O.C.
email: TSHIH@CS.TKU.EDU.TW
fax: Intl. (02) 623-8212

Abstract

An interactive multimedia presentation system is introduced. We firstly define a formal specification for interactive presentations in the Z notation. The model looks at the presentation from two views: the navigation view and the representation view. The presentation navigation is based on message passing among presentation frames of a presentation, while common information is inherited and shared by frames. The system allows a presenter to plan the audience's reaction in advance. When the audience is watching a presentation, the underlying inference system is learning from his/her responses. This mechanism makes a presentation to be proceeded again act according to the audience's background and knowledge. Thus, the resulting presentation is more diversified.

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) [6, 7] 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. WIP [1] is able to generate knowledge-based presentations that explain to a user how to use an espresso machine. The work described in [3] integrates knowledge representation systems and a propositional logic theorem prover to create text and map based illustrations showing the situations and plans of a Navy's fleet. APT (A Presentation Tool) [9, 10] automatically generates graphical presentations of relational information. A Piano tutor described in [5] is able to use coordinated media, video, voice, and graphics display, to teach beginners how to play a piano.

Other articles [11, 4] address relations between multimedia and multiple modalities. A modal is the way information is presented, such as a natural language statement, one piece of picture, or a table. A medium is the device which presentation is delivered, such as sound, text, or video. The approach described in [4] suggests that the mapping should be constructed between characteristics of media and characteristics of information. The research discussed in [4] also proposes that presentation knowledge can be classify naturally into four major groups: characteristics of media, characteristics of information, goal of the presenter, and the background of addressees. Issues related to solving the synchronization problems of temporal multimedia resources can be found in [12]. A multimedia collaborative design environment for scientific and engineering applications can be found in [2]. The work described in [8] is a learning environment for the second year French students to learn about French culture. This multimedia application uses maps and visual icons as well as video files to show locations that can be visited in a city.

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 designs:

1. Authorware Professional by Macromedia, Inc.
2. Multimedia Viewer by Microsoft
3. Multimedia Toolbook by Asymetrix Corporation
4. Hypermedia System by ITRI
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 allowing the presenter to specify presentation objects and controls, which can be decomposed into several levels in a hierarchy 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 allowing 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 click and drawing. Multimedia Viewer also provides a set of medium editing tools. Presentation objects produced by these tools can be linked together by a script language supporting functions, data structures, and commands. None of the above systems, however, allows the interactive sequences provided by a user to be learned by a presentation.

As a result, presentations created by these tools were either communicating with its addressees in a single direction, or providing limited navigation controls for the audiences via push buttons or menus. These presentations can not incorporate addressees' responses. Thus, an audience watches the same demonstration over and over again even he/she has told the computer one understands the topic. As the communication efficiency became better and better between multimedia computers and human, an intelligent multimedia presentation design system will further make computers speak, show, and interact with human better.

Our research[1] is to investigate presentation design techniques and to develop a system that helps multimedia presentation designers to deliver intelligent multimedia applications as CD ROM titles. The system focuses on the following criteria:

- Intelligent presentation design environment and specification language
- Addressee characteristics specification and learning
- Canonical representation of knowledge
- Multimedia resources DBMS and reuse of presentations

Presentation intelligence is represented in a canonical rule-based format. These knowledge not only include the addressee's background (i.e., common sense of the person who watches the presentation), but allow human reactions to be learned by the presentation program. A database management system is also designed for the CD ROM title designers to organize and store multimedia resource information. This database can be integrated to the presentation reuse model thus the system performs as a configuration management system for the multimedia presentation development. An intelligent specification language is designed. The language provides facilities for hypermedia access and rule-based statements for knowledge representation. The system supports personalization. Not only the graphical user interface of the generated presentation can be fully customized, but the underlying knowledge of the addressee can be easily updated. The system also provides a learning subsystem to be included in the generated title which allows an addressee's interaction be asserted into the knowledge base. This learning environment, the presentation inference engine, and components marked with a "*" in figure 1 will be provided as the runtime environment of a CD ROM title. Figure 1 illustrates the overall architecture of our proposed system which includes an user interface subsystem, a database subsystem, and a runtime subsystem.

The preliminary results of our research can be found in [14, 15]. In this paper, we give a formal specification of a multimedia presentation.

[1]This project is supported by the NSC grant # NSC 85-2213-E-032-011 of the Republic of China.

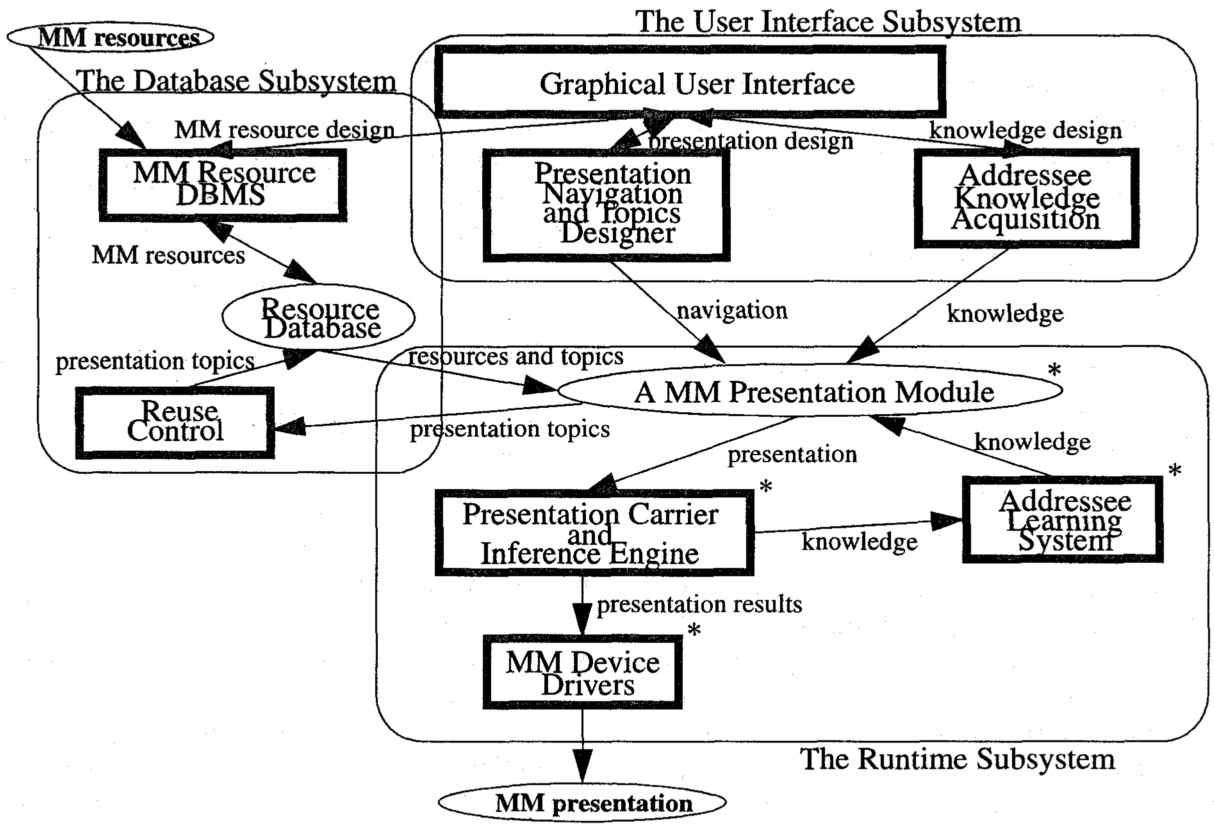

Figure 1: A system for intelligent multimedia presentation designs

Since we use the *Z* notation as a tool to discuss our formal specification of a presentation, we give a short summary of the *Z* formal specification language in section 2. In section 3, we propose a model for the representation, navigation, and inheritance of presentation frames (to be discussed). We then address the language designing issues in section 4. A short section (section 5) is given to highlight our contributions and point our our further directions.

2. A Brief Review of Z

Z notation [13] is a language for expressing formal specifications of systems. It is based on typed set theory, coupled with a structuring mechanism (i.e., the 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 given 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 definitions[2] add nothing to the power of the *Z* language but ease the definition of recursive objects. Free type definitions can be translated into the other *Z* constructs. An abbreviation definition using 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 follows, we will introduce concepts and some syntax of *Z* notation while necessary.

The reason we take a formal specification approach is that formal specifications use mathematical notation to precisely describe **what** proper-

[2]Free type definitions are discussed in [13] as a short mechanism to introduce new types in Z.

ties a system need to have, without concerning too much about **how** these properties are implemented. This approach avoids our discussion of the system from being too tedious. Z notation is a formal specification language widely used in Europe. The notation we use here follows the standard given in [13]. We find that, by using the mathematical tool-kit provided in [13], we can easily describe our formal specification of multimedia presentations.

3. The Formal Specification

Most presentations produced by multimedia authoring tools are designed as a hypermedia document. Unlike traditional presentations using slides, multimedia presentations are proceeded in a nonlinear manner in that push buttons are provided for the user to navigate among different related issues in a presentation. These presentation issues are not totally independent for two reasons. Firstly, two issues closely related to each other should be linked such that an addressee watching the presentation can refer to related issues easier by following the links. For instance, a presentation touring Paris may link a picture of the Paris Tower with a description, either by audio or text, of its history. Secondly, two issues presented in a presentation may share common information. There is no reason of storing duplicated data in one presentation which makes a consistent update tiresome. For instance, the addressee's name and background that can be used in an interactive presentation should be stored only once. While updated, the addressee's name will be changed consistently. We suggest that a presentation can be designed from two different point of views: the *navigation view* and the *representation view*. From the navigation view, a presentation is a graph with nodes as issues in the presentation and edges as relations between issues. From the representation view of a presentation, information that can be shared among issues are background of the addressee (e.g., the addressee's name or knowledge to the presentation topic), multimedia resources (e.g., a text file of description or a video file showing a mechanical operation), or other knowledge useful in the presentation. A property inheritance structure such as a tree or a DAG (directed acyclic graph) is suitable for our knowledge representation architecture. In this section, we propose a presentation model meets the above two criteria.

Before we discuss the presentation model, some terminologies are addressed. A multimedia *resource* is a picture, a description, a video, or other materials that can be used in a multimedia computer. A *topic* is a resource carrier that presents the resource to the addressee. A *frame* is a composed object which represents related issues that a presenter wants to illustrate. A frame may contain push buttons, one or more topics to be presented, and a number of knowledge rules. A *message* with optional parameters can be passed between two frames (or back to the same frame).

We use a number of abbreviation definitions to represent the domains of objects (e.g., messages, presentations, or resources) in our system. In this paper, following most of the discussions in Z notation, a domain has its name all in capital, a schema name starts with a capital letter followed by lower case letters, and variables or functions are all in lower case. A presentation uses some *presentation resources* (in domain PR), contains a number of *frames* (in domain FRM), and uses some *messages* (in domain MSG). We leave the discussion of PR, FRM, and MSG in the following subsections. Note that, in Z, $\mathbb{P}\, X$ denotes a set of objects in domain X. And, $\mathbb{P}_1\, Y$ represents a non-empty set of objects in domain Y. Thus, the domain of presentations is a cartisian product shown below:

$$\begin{aligned}&PRESENTATION == \\ &\quad \mathbb{P}\, PR \times \mathbb{P}\, FRM \times \mathbb{P}\, MSG\end{aligned}$$

Next, we define our first schema. As shown in between two horizontal lines leading by a name, schema *Presentation* holds a *presentation* in domain *PRESENTATION* and a resource database *resourcedb* where the presentation extracts presentation resources. The schema also contains a number of frames used in the presentation. This schema will be *referenced* by other schemae. The concept of a schema reference is similar to (but not exactly the same) inheritance in that variables and functions of a referenced schema can be used in its child schema. Schema reference is achieved by declaring the name of a referenced schema inside a child schema[3].

$$\begin{array}{|l}
\textit{Presentation} \\
\hline
\textit{resourcedb} : \textit{RDB} \\
\textit{presentation} : \textit{PRESENTATION} \\
\textit{frames} : \mathbb{P}_1\ \textit{FRM} \\
\hline
\end{array}$$

We discuss these objects in more detail from two views in the following two subsections.

3.1. Presentation Navigation

From the navigation view, a frame is a multidigraph with possible loops. That is, a graph with multiple edges between two nodes and an edge starts and ends with the same node. A frame is a node while a message is an edge in the graph. Multiple edges mean that one or more messages are passing between two frames. A loop represents a frame passes a message back to itself (e.g., a `close` message closes the frame itself).

A frame has a unique name in the name domain N. Objects contained in a frame include buttons in domain B, topics in domain T, in coming messages and out going messages in domain MSG [4], and knowledge in domain K (to be discussed). A message has a name, a source frame (specified in the first FRM domain), a destination frame, and some parameters in PAR. A button is consist of a button definition BD (such as its resolution RES, color, or location), in coming messages which enable or disable the button, and an out going message which is sent out while the button is pushed. A topic, besides its topic definition TD, demonstrates a presentation resource PR while some messages ($\mathbb{P}\ MSG$) are received. Note that a topic may receive different messages that manipulate the resource. For instance, a topic playing a video may receive `play`, `fast_forward`, and `stop` messages. Some detail definitions of domains, such as BD and TD are omitted, while N and PAR are just ASCII codes. The following are these domain definitions:

$$\begin{array}{l}
FRM == N \times \mathbb{P}\ B \times \mathbb{P}\ T \times \mathbb{P}_1\ MSG \times \mathbb{P}_1\ MSG \times K \\
MSG == N \times FRM \times FRM \times PAR \\
B == BD \times \mathbb{P}\ MSG \times MSG \\
T == TD \times PR \times \mathbb{P}\ MSG \\
N == ASCII \\
PAR == ASCII \\
BD == RES \times \ldots \\
TD == RES \times \ldots
\end{array}$$

$$\begin{array}{|l}
\textit{Navigation} \\
\hline
\Xi \textit{Presentation} \\
\textit{presources} : \mathbb{P}\ PR \\
\textit{messages} : \mathbb{P}\ MSG \\
\hline
\textit{presentation} = (\textit{presources}, \textit{frames}, \textit{messages}) \\
\\
\forall\, m_1, m_2 : MSG \bullet \exists\, sf_1, sf_2, df_1, df_2 : FRM \bullet \\
m_1, m_2 \in \textit{messages} \land \\
m_1 = (\textit{mname}_1, sf_1, df_1, par_1) \land \\
m_2 = (\textit{mname}_2, sf_2, df_2, par_2) \bullet \\
((sf_1 = sf_2) \land (df_1 = df_2) \Rightarrow \\
\textit{mname}_1 \neq \textit{mname}_2) \land \\
sf_1 \in \textit{frames} \land sf_2 \in \textit{frames} \land \\
df_1 \in \textit{frames} \land df_2 \in \textit{frames} \\
\\
\forall\, f : FRM \bullet f \in \textit{frames} \land \\
f = (\textit{fname}, \textit{buttons}, \textit{topics}, \\
\textit{f\_in\_msgs}, \textit{f\_out\_msgs}, \textit{knowledge}) \bullet \\
(\forall\, b : B \bullet \exists\, \textit{in\_msg} : MSG \bullet \\
b \in \textit{buttons} \land b = (\textit{def}, \textit{in\_msgs}, \textit{out\_msg}) \land \\
\textit{in\_msg} \in \textit{in\_msgs} \land \\
\textit{in\_msg} = (\textit{in\_name}, sf, df, par) \Rightarrow \\
\textit{in\_name} \in \{\ \texttt{enable}, \texttt{disable}\ \} \land \\
sf \in \textit{frames} \land df \in \textit{frames} \land \\
\textit{in\_msgs} \subset \textit{f\_in\_msgs} \land \\
\textit{out\_msg} \in \textit{f\_out\_msgs}) \land \\
(\forall\, m : MSG \bullet m \in \textit{f\_in\_msgs} \land \\
m = (\textit{mname}, sf, df, par) \Rightarrow df = f) \\
\\
\forall\, f : FRM \bullet \exists\, rsf : RSF \bullet f \in \textit{frames} \land \\
f = (\textit{name}, \textit{buttons}, \textit{topics}, \textit{in\_msgs}, \\
\textit{out\_msgs}, \textit{knowledge}) \bullet \\
\forall\, t : T \bullet t \in \textit{topics} \land \\
t = (\textit{def}, \textit{presource}, \textit{msgs}) \Rightarrow \\
\textit{msgs} \subset \textit{messages} \land \\
\textit{presource} \in \textit{presources} \land \\
\exists\, sc : SC \bullet \textit{presource} \in rsf(\textit{resourcedb}, sc) \\
\hline
\end{array}$$

Navigation references *Presentation*. Besides inheriting the *resourcedb*, the *presentation*, and the *frames* variables from *Presentation*, *Navigation*

[3] Inside the *Navigation* schema to be discussed, we will reference the *Presentation* schema.

[4] $\mathbb{P}_1\ MSG$ is for the in coming messages and the second $\mathbb{P}_1\ MSG$ is for the out going messages.

further defines some variables for presentation resources (*presources*) and *messages* used in the presentation. The Ξ sign annotating the *Presentation* reference indicates that the state of *Presentation* is not changed by *Navigation*. Expressions below the center horizontal line of the *Navigation* schema are restrictions, known as axioms, applied to variables. The first axiom indicates that the presentation is the composition of the three objects. The second axiom says that a message passed in between two frames has a unique name, and all frames used in the presentation are defined in *frames*. A first order logic expression such as $\exists\, n : N \bullet n = \texttt{message\_name}$ introduces a variable n in the domain N such that the variable is restricted to its value `message_name`. We use this notation in most of the axioms of our model. The third axiom ensures that a button only accepts a "`enable`" or "`disable`" message, and only the destination frame can receive the specific message sent to it. The last axiom indicates that there exist some selection criteria for a resource to be a presentation resource used in a frame. The domain of SC and RSF are for the selection criteria and resource selection functions, which are not discussed in this paper due to space limitation.

3.2. Knowledge Inheritance and Inference

$$
\begin{aligned}
&K == \mathbb{P}\, KR \cup \mathbb{P}\, KF \cup \mathbb{P}\, KQ\\
&KR == P \times PS\\
&KF == P\\
&KQ == P\\
&P == ASCII\\
&PS == \mathbb{P}\, P\\
&LEXP == P \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \mid P \Leftrightarrow P\\
&BND == P \rightarrow P
\end{aligned}
$$

As mentioned at the beginning of this paper, common knowledge and information regarding the addressee's background can be shared through a presentation. The information is kept as the knowledge of a presentation. We are considering three types of knowledge in our knowledge domain, K. *Knowledge rules* (KR) are logical implications[5]. *Knowledge facts* (KF) are conclusions hold in a frame. In order to start the knowledge inference, the third type of knowledge is a *query* (KQ) associated with each frame that, when the frame receives an **open** message, starts the inference. The domain P represents predicates, similar to predicates in Prolog, with their detail omitted (specified as $ASCII$). Also, PS is a set of predicates. The domain $LEXP$ specifies logic expressions obtained from predicates by using logic operators $\wedge, \vee, \Rightarrow$, *and* $\Leftrightarrow$. The last domain we defined here is the binding (BND), which is used in the logic inference procedure. The concept of a binding is similar to the concept of an environment of a procedural language program. Our process to accumulate variable binding is rather similar to Prolog. We are not specifying the detail here. A binding is also a postfix operator. While applied to a predicate, the binding substitutes values for free variables occur in the predicate.

```
— Representation ————————————————
  ΞPresentation
  rules : ℙ KR
  facts : ℙ KF
  queries : ℙ KQ
  knowledge : K
————————————————
  knowledge = rules ∪ facts ∪ queries

  knowledge = ⋃{ k : K |
  ∀ frame : FRM • frame ∈ frames •
  frame = (fn, btns, tps, in_msgs, out_msgs, k) }

  ∀ f : FRM •
  f = (fn, btns, tps, in_msgs, out_msgs, k) ∧
  f ∈ frames • ∃ kq : KQ • kq ⊂ k ∧ #kq = 1
————————————————————————————————
```

From the knowledge (or representation) point of view, a presentation is a directed acyclic graph (i.e., DAG) with frames as nodes and inheritance relations as edges. We define the representation of knowledge inheritance and inference in three schemae: *Representation*, *Inheritance*, and *Inference*. In *Representation*, *rules*, *facts*, and *queries* are the set of all rules, facts, and queries used in the *presentation*. *knowledge* is a set union of these rules, facts, and queries. *knowledge* is also a union of the knowledge of all frames in the *presentation*. The last axiom specifies that each frame of a presentation has only one query.

[5]We have also defined language constructs that allow rules to be expressed easier in addition to using implications.

Inheritance

$$\begin{array}{|l}
\Xi \mathit{Representation} \\
\gamma : \mathit{FRM} \leftrightarrow \mathit{FRM} \\
\delta : \mathit{FRM} \rightarrow K \\
\Delta : \mathbb{P}\, \mathit{FRM} \rightarrow K \\
\hline
\forall f_1, f_2, f_3 : \mathit{FRM} \bullet f_1 \gamma f_2 \wedge f_2 \gamma f_3 \Rightarrow f_1 \gamma f_3 \\
\exists f_2 : \mathit{FRM} \bullet \forall f_1 : \mathit{FRM} \bullet f_1 \gamma f_2 \Rightarrow f_1 \neq f_2 \\
\exists f_1, f_2 : \mathit{FRM} \bullet \exists f_3 : \mathit{FRM} \bullet f_1 \gamma f_3 \wedge f_2 \gamma f_3 \\
\forall \mathit{frame} : \mathit{FRM} \bullet \\
\mathit{frame} = (\mathit{fn}, \mathit{btns}, \mathit{tps}, \mathit{in\_msgs}, \mathit{out\_msgs}, k) \Rightarrow \\
k \subseteq \delta \mathit{frame} \\
\forall f_1, f_2 : \mathit{FRM} \bullet f_1 \gamma f_2 \Rightarrow \delta f_1 \subseteq \delta f_2 \\
\forall f_1, f_2, f_3 : \mathit{FRM} \bullet \\
f_1 \gamma f_2 \wedge f_1 \gamma f_3 \Rightarrow \delta f_2 \cap \delta f_3 = \delta f_1 \\
\forall f_1, f_2 : \mathit{FRM} \bullet f_1 \gamma f_2 \Rightarrow \delta f_2 = \delta f_1 \bigoplus \delta f_2 \\
\forall \mathit{fs} : \mathbb{P}\, \mathit{FRM} \bullet \exists f : \mathit{FRM} \bullet \\
\mathit{fs} = \{\ f\ \} \Rightarrow \Delta \mathit{fs} = \delta f \vee \\
\forall \mathit{fs} : \mathbb{P}\, \mathit{FRM} \bullet \exists f : \mathit{FRM} \bullet \\
\#\mathit{fs} > 1 \wedge f \in \mathit{fs} \Rightarrow \\
\Delta \mathit{fs} = \Delta(\mathit{fs} \setminus \{\ f\ \}) \cup \delta f \\
\forall \mathit{fs} : \mathbb{P}\, \mathit{FRM} \bullet \exists f_2 : \mathit{FRM} \bullet \\
\exists k : K \bullet \forall f_1 : \mathit{FRM} \bullet \\
f_1 \gamma f_2 \wedge f_1 \in \mathit{fs} \Rightarrow \Delta \mathit{fs} \cup k = \delta f_2 \\
\hline
\end{array}$$

In schema *Inheritance*, we define a super frame relation, γ. $\mathit{frame}_{super}\ \gamma\ \mathit{frame}_{sub}$ indicates that frame_{super} is the super frame of frame_{sub}. Note that this relation is transitive, as indicated in the first axiom belows the center horizontal line. The second axiom ensures that the inheritance architecture is acyclic. That is, a frame can not be a super frame of its own. However, multiple inheritance is allowed (indicated by the third axiom). We further define the functions, δ, and Δ, extract the knowledge from a frame and the knowledge from a set of frames, respectively. The definition of δ function is given in axiom four. The fifth axiom says that a subframe (f_2) inherits all knowledge from its super frame (f_1). And the sixth axiom says that knowledge common to two subframes should belong to their super frame. The seventh axiom says that knowledge of a subframe overrides the one in its super frame. The notation $f \bigoplus g$ indicates a functional overriding. Function Δ is defined recursively in the eighth axiom. Note that, $\#\mathit{fs}$ represents the number of frames in the frame set fs. And $\mathit{fs} \setminus \{\ f\ \}$ is a new set of frames taking all frames from fs, except frame f. The last axiom says that a frame inherits knowledge from all of its super frames.

Inference

$$\begin{array}{|l}
\Xi \mathit{Representation} \\
\aleph : P \times PS \rightarrow \mathit{LEXP} \\
\Re : P \rightarrow \mathit{LEXP} \\
\Im : \mathbb{P}\, \mathit{KR} \rightarrow \mathit{LEXP} \\
\mathfrak{U} : \mathbb{P}\, \mathit{KF} \rightarrow \mathit{LEXP} \\
\alpha : P \times P \rightarrow \mathit{BND} \\
\Theta : \mathit{KQ} \times \mathit{BND} \rightarrow \{\ \mathit{true}, \mathit{false}\ \} \\
\mathit{act\_frame} : \mathit{FRM} \\
\mathit{act\_query} : \mathit{KQ} \\
\mathit{act\_facts} : \mathbb{P}\, \mathit{KF} \\
\mathit{act\_rules} : \mathbb{P}\, \mathit{KR} \\
\mathit{f\_facts} : \mathbb{P}\, \mathit{KF} \\
\mathit{f\_rules} : \mathbb{P}\, \mathit{KR} \\
\hline
\forall r : \mathit{KR} \bullet r \in \mathit{rules} \bullet \exists q, p_1, p_2, ..., p_n : P \bullet \\
r = (q, \{\ p_1, p_2, ..., p_n\ \}) \Rightarrow \\
\aleph r = (q \Rightarrow p_1 \wedge p_2 \wedge ... \wedge p_n) \\
\forall f : \mathit{KF} \bullet f \in \mathit{facts} \bullet \exists p : P \bullet f = p \Rightarrow \Re f = p \\
(\exists r : \mathit{KR} \bullet \mathit{rules} = \{\ r\ \} \Rightarrow \Im \mathit{rules} = \aleph r) \vee \\
(\exists r : \mathit{KR} \bullet r \in \mathit{rules} \wedge \\
\#\mathit{rules} > 1 \Rightarrow \Im \mathit{rules} = \aleph r \vee \Im(\mathit{rules} \setminus \{\ r\ \})) \\
(\exists f : \mathit{KF} \bullet \mathit{facts} = \{\ f\ \} \Rightarrow \mathfrak{U} \mathit{facts} = \Re f) \vee \\
(\exists f : \mathit{KF} \bullet f \in \mathit{facts} \wedge \#\mathit{facts} > 1 \Rightarrow \\
\mathfrak{U} \mathit{facts} = \Re f \vee \mathfrak{U}(\mathit{facts} \setminus \{\ f\ \})) \\
\forall p_1, p_2 : P \bullet \alpha(p_1, p_2) = \emptyset[\mathit{BND}] \vee \\
(\exists \rho : \mathit{BND} \bullet \alpha(p_1, p_2) = \rho \Rightarrow p_1 \rho = p_2 \rho) \\
\mathit{act\_frame} = (\mathit{fn}, \mathit{btns}, \mathit{tps}, \mathit{in\_msgs}, \\
\mathit{out\_msgs}, \{\ \mathit{act\_query}\ \} \cup \mathit{f\_facts} \cup \mathit{f\_rules}) \\
\mathit{act\_facts} \cup \mathit{act\_rules} = \bigcup \{\ k : K \mid \forall f : \mathit{FRM} \bullet \\
f \gamma \mathit{act\_frame} \bullet k = \delta f\ \} \cup \mathit{f\_facts} \cup \mathit{f\_rules} \\
(\exists f : \mathit{KF} \bullet f \in \mathit{act\_facts} \wedge \\
\alpha(f, \mathit{act\_query}) \neq \emptyset[\mathit{BND}] \Rightarrow \\
\Theta(\mathit{act\_query}, \emptyset[\mathit{BND}]) = \mathit{true}) \vee \\
(\forall f : \mathit{KF} \bullet f \in \mathit{act\_facts} \wedge \\
\alpha(f, \mathit{act\_query}) = \emptyset[\mathit{BND}] \Rightarrow \\
(\exists \rho : \mathit{BND} \bullet \exists q : P \bullet \exists r : \mathit{KR} \bullet \\
r \in \mathit{act\_rules} \wedge r = (q, \{\ p_1, p_2, ..., p_n\ \}) \wedge \\
\alpha(q, \mathit{act\_query}) = \rho \wedge \rho \neq \emptyset[\mathit{BND}] \Rightarrow \\
\Theta(\mathit{act\_query}, \emptyset[\mathit{BND}]) = \\
\Theta(p_1, \rho) \wedge \Theta(p_2, \rho) \wedge ...\Theta(p_n, \rho))) \vee \\
((\forall f : \mathit{KF} \bullet f \in \mathit{act\_facts} \wedge \\
\alpha(f, \mathit{act\_query}) = \emptyset[\mathit{BND}]) \wedge \\
(\forall \rho : \mathit{BND} \bullet \forall r : \mathit{KR} \bullet \exists q : P \bullet \\
r \in \mathit{act\_rules} \wedge \\
r = (q, \{\ p_1, p_2, ..., p_n\ \}) \wedge \\
\alpha(q, \mathit{act\_query}) = \emptyset[\mathit{BND}]) \Rightarrow \\
\Theta(\mathit{act\_query}, \emptyset[\mathit{BND}]) = \mathit{false}) \\
\hline
\end{array}$$

The *Inference* schema is to express the semantics of rules and facts and to discuss how a query is computed by the knowledge inference engine. Besides inheriting variables and functions from the *Representation* schema, A number of functions are also used in *Inference*. The functions $\aleph$, and $\Re$ are for the semantics of a rule and a fact, respectively. They take as input the rule and the fact, and produce the logic expressions as values. The semantics of a set of rules and a set of facts are the disjunctions of the individual facts and rules, as indicated in functions $\Im$ and $\mho$. The α function is the unification function. We only specify that either two predicates, p_1 and p_2, do not match each other; thus, an empty binding $\emptyset[BND]$ is produced. Or, if the most general unifier of p_1 and p_2 is ρ, p_1 and p_2 are equal after the substitution. The last function, Θ, is the logic inference function. It takes a query and an empty binding as inputs, and uses *facts* and *rules* for the inference. In this function, if the query matches one of the facts, the computation succeeds. Otherwise, the Θ function is called recursively by using the conclusion part of a rule and a new binding obtained from the hypothesis of the rule and the query. If none of the above case occurs, the inference fails.

4. The Proposed Language

Our original goal was to design a specification language that allows a presenter to design a presentation as a collection of frames, with messages and inheritance relations defined in these frames. However, since the system will be used by a non-programmer eventually, a language is not superior than a friendly user interface that guides a presenter to accomplish his/her presentation design. Thus, we refined the language and designed a graphical user interface. The system will collect pieces of statements specified by the designer in different windows of our interface and compose an internal specification program for each presentation designed. To create an intelligent presentation, a presenter needs to provide the following information:

- Presentation Resources
- Presentation Knowledge
- Navigation Rules
- Frame layouts

Presentation resources are standard multimedia files, such as Windows AVI files, created by the designer using commercial tools. After storing the resources in the database, a resource file can be used in different presentations. Presentation knowledge is a set of rule-based representation of the addressee's background, such as the addressee's familiarity with a special keyword, the presentation designer must consider in the presentation. Also, as the presentation proceeds, the user's knowledge may be incorporated into the presentation knowledge base. The user interface allows the designer to specify a query, facts, and rules for each frame. The query and knowledge facts have the following syntax:

```
query Predicate.
known Predicate.
```

where `Predicate` is a Prolog term. A knowledge rule can be specified in one of the following formats:

```
  if Predicate then Predicates.
or
  if Predicate then Predicates
                    else Predicates.
or
  case Predicate -> Predicates $
       Predicate -> Predicates $
       ...
       Predicate -> Predicates $
       true -> Predicates.
```

where `Predicates` is a list of one or more `Predicates` separated by commas. The first rule above indicates that if the predicate in the test part of the `if-then` rule satisfies, the conclusion predicates follow. The second rule specifies that if the testing predicate succeeds, the predicates after the `then` keyword hold; otherwise, the predicates after the `else` keyword hold. The `case` rule is an extension of the `if-then` rule. If any of the testing predicate holds, the corresponding predicates after the -> keyword also hold. The last branch true -> Predicates is optional, which allows a default case to be declared. Knowledge inheritance relations are declared by specifying the super frame of each frame. A special frame named root is a place holder. When a frame declares root as its super frame, the frame must be the first to be invoked in a presentation. When the root frame starts, its sends open messages to all of its child frames.

We allows the designer to take a hypermedia traversal approach in the design of their presentation navigation. Three types of objects need to be defined for each frame. They are button definitions, topic definitions, and message definitions. The appearance of a button is drawn in the design area of a frame. This button appearance will be converted to a list of property-pair items as the property list of the button. Moreover, each button is associated with one or more actions separated by commas. These actions will be applied while the button is pushed. The kinds of actions used in our system could be:

```
send Msg(SrcFrame, DstFrame, [ Parms ])
set_btn(Frame, Button, "enable")
set_btn(Frame, Button, "disable")
assert(Predicate)
retract(Predicate)
proc_call("C_procedure_call")
```

The send action sends a message Msg from SrcFrame to DstFrame with optional parameters Parms. It is necessary to keep the source frame of a message in the internal representation of a presentation since, in some cases, frames are designed with a "back" button that allows the addressee to backtrack to a previously worked frame. The set_btn action will enable or disable a button in a particular frame. The assert and the retract actions apply only to the current frame. We only allow the knowledge set of a frame to be changed while the addressee is visiting the frame. This makes the knowledge assertion or retraction local to a particular user interaction and simplifies the knowledge collection process. The proc_call action calls an external C procedure which in turn makes an MCI (i.e., Media Control Interface by Microsoft) call to the underlying MS Windows multimedia drivers.

A topic definition is similar to a button definition, excepts that a topic does not send out a message. However, a topic can receive a message (e.g., play, stop, or forward) and respond with an appropriate action.

A message definition is a collection of message-action pairs. Each pair has the following format:

```
on_message: Message_name( Parms )
  do [ Actions ]
```

Message_name is the name of the message received by the current frame. Parms and Actions are lists of terms with possible sharing of variables. For instance, a message-action pair can be defined as:

```
on_message: play(AVI_file_name)
  do [proc_call(
    "play_resource(AVI_file_name)")]
```

where AVI_file_name is the name of a video file. When AVI_file_name is instantiated to its value, the file name is passed to a C function for playing a video.

The definition of frame layouts is omitted in our discussion. A presentation can be designed using our internal language, or using a graphical user interface of the system. After the presentation program is generated, it is run under the presentation carrier and inference engine subsystem.

5. Conclusions

In this paper, we introduced an intelligent multimedia presentation system allowing a presenter to design intelligent presentations. The presentations designed allow addressees' response to be learned via knowledge assertions as knowledge inference results of some pre-planned knowledge rules. Our new proposed model, by using an object-oriented approach, allows a presenter to design his/her presentation as a hypermedia document with navigation specified as messages among frames. This model, by allowing knowledge inheritance, also facilitates data sharing and ensures a consistent updating of knowledge. A presentation design is entered via our graphical user interface. Different components of a frame are given in different windows. A program generator takes these components and produces a presentation which is run by our presentation carrier and inference engine subsystem. We have a prototype environment designed under Microsoft Windows. The implementation language is C and Prolog. Some sample applications are also designed to show the usage of our system.

However, we are still seeking for solutions to improve our system. For instance, the TMS (Truth Maintenance System) technique could be used to check the consistency of knowledge rules in a presenter's design. And, we are designing the second version of our graphical user interface which is more friendly and powerful. Also, we are analysing presentation rules used commonly by people. Some system-wide rules address mappings between multimodal and multimedia need to be defined in our system in order to help the presenters to make better presentations.

References

[1] E. Andre, et. al., "WIP: The Automatic Synthesis of Multimodal Presentations," In Intelligent Multimedia Interfaces, edited by Mark T. Maybury, American Association for Artificial Intelligence, pp 75–93, 1993.

[2] V. Anupam and C. L. Bajaj, "Shastra: Multimedia Collaborative Design Environment," IEEE Multimedia, summer, 1994.

[3] Y. Arens, et. al., "Presentation Design Using an Integrated Knowledge Base," In Intelligent User Interfaces, edited by J. W. Sullivan and S. W. Tyler, ACM Press, pp 241–258, 1991.

[4] Y. Arens, et. al., "On the Knowledge Underlying Multimedia Presentations," In Intelligent Multimedia Interfaces, edited by Mark T. Maybury, American Association for Artificial Intelligence, pp 280–306, 1993.

[5] R. B. Dannenberg and R. L. Joseph, "Human Computer Interaction in the Piano Tutor," In Multimedia Interface Design, edited by M. M. Blattner and R. B. Dannenberg, ACM Press, pp 65–78, 1992.

[6] 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.

[7] 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.

[8] E. Schlusselberg, "Dans le Quartier St. Gervais: An Exploratory Learning Environment," In Multimedia Computing: Case Studies from MIT Project Athena, eited by M. E. Hodges, and R. M. Sasnett, Addison-Wesley, pp 103– 115, 1993.

[9] J. D. Mackinlay, "Automatic Design of Graphical Presentations," Ph. D. thesis, Stanford University, 1987.

[10] J. D. Mackinlay, "Search Architectures for the Automatic Design of Graphical Presentations," In Intelligent User Interfaces, edited by J. W. Sullivan and S. W. Tyler, ACM Press, pp 281–292, 1991.

[11] J. G. Neal and S. C. Shapiro, "Knowledge Based Multimedia Systems," In Multimedia Systems, edited by J. F. K. Buford, ACM Press, pp 403–438, 1994.

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

[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, "An Artificial Intelligent Approach to Multimedia Authoring," in Proceedings of the Second IASTED/ISMM International Conference on Distributed Multimedia Systems and Applications, Stanford, California, U.S.A., 1995.