Automatic Generation of Electronic Program Guides

Brian Bailey
CWI
P.O. Box 94079
NL-1090 GB Amsterdam
The Netherlands
+31 20 592 4127

1. INTRODUCTION

The purpose of this work is to explore architectures, methods, and techniques for automatically generating well-designed hypermedia presentations in the context of an electronic program guide. Specifically, this work enables the user to query a simulated multimedia database containing program guide information, and in response to the query, automatically generate a well-designed, user-tailored, hypermedia presentation. Although the work is not yet complete, an initial prototype of the EPG system has been implemented. The results from this effort can be further extended or the lessons learned can be applied prior to developing a large-scale follow-on system. The purpose of this paper is to report on the architecture of the prototype system in terms of the SRM, provide details of selected parts of the system, and contribute ideas for continued as well as future work.

2. RELATED WORK

The classic work in the area of intelligent multimedia presentation systems is by Andre[1], Feiner[2], Graf [3], and Maybury[4]. The common view of their research is that the automatic generation of multimedia presentations is best achieved through a knowledge base and planning effort. The novel part of their approach is not only in the application of AI planning systems, but also in the modelling of multimedia presentations in terms of discourse structure and theory. Given a presentation goal, user knowledge, and presentation environment, the system selects a top-level plan operator (goal), and then tries to achieve the goal by recursively achieving its subgoals. The result is that the leaf nodes of the plan (goal) tree represent the atomic communicative acts the system must perform. The biggest disadvantage of these systems is that they are non-interactive for the user, although the authors are currently attempting to address this issue. To further understand these systems, the best approach is to obtain the implementations and play with them internally. It is also unclear as to the actual complexity of the underlying layout components in these systems.

3. CONTROL LAYER

The control layer of the EPG system is responsible for accepting a presentation goal from either the user or an external system component. Conceptually, the presentation goal represents the overall message that the EPG system has been requested to convey to the user. As a result, the quality of the presented information is commensurate with the quality of the original goal specification. Once the presentation goal has been defined, the control layer decomposes the goal into smaller subgoals, encodes the subgoals into a machine-readable format, and then passes the encoded sub-goals to the content layer for further processing.

3.1 SPECIFYING THE GOAL IN THE EPG SYSTEM

For the automated presentation of EPGs, goals are specified using two distinct mechanisms. First, the goal formulation interface enables the user to specify a new search query at any time during the presentation. This interaction between the user and the goal formulation interface is analogous to how users currently interact with web-based search engines. If the presented information does not meet the user's current information needs, then the user may immediately specify a new query and receive a new presentation. In the current implementation of the EPG, SQL is used as the goal representation language. For example, a goal is given to the system with the command:

```
submit_goal(SQL_Query).
```

However, future implementations of the EPG system may provide a different goal formulation interface such as a pre-defined menu, an HTML form, or a web-style search language. When the user sends an SQL query to the Control layer of the EPG system, the requested attributes are annotated as representing the *locus of interest*. The locus of interest represents the user's primary information interest; and as a result, the EPG system should visually design the final presentation so that the user's attention is drawn to the information forming the locus of interest before other parts of the presentation.

The second mechanism for specifying a goal to the EPG system is through the use of *dynamic links*. A dynamic link represents a refinement to the current goal, whereas a static link allows the user to navigate the hyperweb resulting from the original generated presentation. Each presentation generated as the result of a query will contain zero or more dynamic links. Dynamic links are an important aspect of an interactive presentation system for at least two reasons:

- *Delay presentation generation until needed*. Inserting a dynamic link saves the presentation system from generating peripheral parts of the presentation until the user explicitly requests to view it. Strategies for locating the appropriate places to insert dynamic links are still an open issue. However, the current system can easily be augmented to use the relational database structure as a starting point. Using this method, dynamic links would be inserted wherever a structural relation between two tables is found; i.e., foreign keys.
- *Give context for further generation*. Initial goals specified by the user are assumed to be very broad. Providing the user with a method for interactively refining their initial goal should prove to be invaluable.

Selecting a dynamic link invokes the entire generation process again using the refined goal attached to the link; whereas selecting a static link simply moves the user to the link's destination. For example, suppose the user initially requests information about movies involving man-eating fish. As part of the presentation, information about the movie's director is represented as a dynamic link. Note that this part of the hypermedia presentation has yet to be explicitly generated. If the user then selects the director link attached to the movie, say Piranha, then a refined query is posed to the system regarding directors of movies involving man-eating fish; with the locus of interest being the Piranha movie.

4. CONTENT LAYER

The Content layer is responsible for selecting the appropriate media content that meets the information needs of the specified presentation goal. The selection process is non-trivial and must rely on extensive domain knowledge in order for the correct content to be selected. When the content is originally stored, semantic information must be attached that describes how the different content items relate to one another. Ultimately, the goal is to define semantic information germane to the current domain, but general enough so that the knowledgebase does not become a collection of pre-defined presentations. The degree of generality desired gives rise to several research questions that need to be answered for the EPG domain. First, what is the range of presentations that the system will generate for the user? What is the method and amount of interaction that the user will have with the system? A small task analysis of the domain would be a worthwhile endeavour. A task analysis can help answer the question of "what are the information needs of the user?" What is it that current systems do not give to them that an IMMPS can?

In the present system, the content layer relies on the simple concept of a slice (adopted from the RMM methodology CACM 1995). for semantic information. A slice represents a subset of the attributes of a single relation. For now, two slices are defined for each table; the primary and secondary slice. When the content layer receives the result of the goal query from the control layer, the set of slices represented by the

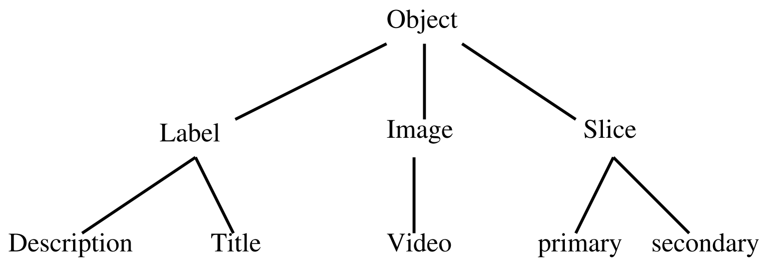

Figure 1. A domain-independent ontology for the EPG domain.

attributes are retrieved. The query result is then augmented with the additional attribute members of each slice. This result is then passed on to the Design Layer.

5. DESIGN LAYER

The design layer is responsible for media generation and overall layout of the media content. The design layer of the EPG system relies on a class-instance hierarchy to define a small ontology relevant to the EPG domain (Figure 1 above). Although the ontology itself is not a novel contribution, the real contribution is in the extensible infrastructure in which it was defined. Adding new concepts, attributes, and relations is trivial and the mechanism for searching and retrieving their values is already defined. For each class in the ontology, a set of class attributes (in the form of name-value pairs) can be attached. Example class attributes are width, height, border, and rhetorical relation to another type. By default, each type inherits the attributes defined for its parent, but the value of an inherited attribute can be overridden by re-defining the same attribute for itself.

Each relational attribute of the movie database is an instance of a class listed in Figure 1. Each instance may also have a set of instance attributes attached that override the class attributes. When an instance attribute value is requested, the value of the first matching attribute starting from the current instance and moving up its class hierarchy is retrieved.

To define spatial relations between attributes, a set of specific-to-general rules are defined. The rules are in the format

spatial_relation(AttrA, RelationA, AttrB, RelationB, Rhetoric, position (left, right, above, below)).

For example, a spatial relation stating that the image attribute of the movie table is to be placed underneath the title attribute of the movie table could be defined as follows:

spatial_relation(image, movie, title, movie, _, below).

However, the same relationship could also be defined on a more generic level by relating the classes and not the specific attribute instances:

spatial_relation(image, _, title, _, _, below).

This states that images are to be placed underneath titles regardless of which relation they belong to.

One of the instant lessons from the EPG application, is that achieving an aesthetically pleasing layout for an arbitrary set of data, even with a well-defined ontology, is an extremely difficult task. The difficulty arises due to the notion of context. Using an ontology only allows local decisions to be made; i.e., whether two labels should be placed to the left or above each other. It does not specify what happens when the two labels have extended the bounds of the neighbouring image and as a group should be placed elsewhere. Perhaps a better approach is to define layout templates for each relation and place the research emphasis on the ordering and selection of the content to be placed within the templates.

One novel aspect of the design layer is that the media layout can be specified in terms of space, time, or links. Here is an attempt at formalizing how this works:

```
Sequence(X1, ..., Xn) :-
        Spatial_Sequence(X1, ..., Xn)
            ;
        Link_Sequence(Xfail*, ..., Xn)
            ;
        Temporal_Sequence(Xfail*, ..., Xn).

Spatial_Sequence(X1, ..., Xn) :-
        Horizontal(X1, ..., Xn)
            ;
        Vertical(Xfail*)
        Sequence(Xfail* + 1, ..., Xn).

Link_Sequence(X1, ..., Xn) :-
        NewPage()
        LinkPages()
        Sequence(X1, ..., Xn).

Temporal_Sequence(X1, ..., Xn) :-
        NewPage()
        SequencePages()
        Sequence(X1, ..., Xn).
```

* - Xfail represents the element at which the previous goal failed.

Sequence is a rhetorical structure relation (Mann and Thompson 1987) that takes a set of items as parameters. For example, a bulleted list represents a sequence relation. However, the abstract concept of a sequence relation does not explicitly specify how it must be realized in terms of presentation format. The above set of goals (defined in semi-Prolog syntax) defines how the abstract notion of a sequence of items (x1, ..., Xn) can be achieved in space, time, or links. Note that time and links would not be used together in a real implementation, but are shown simply to demonstrate the idea. Briefly, the set of above goals state that the items are first to be positioned horizontally across the page, and when this fails (due to the horizontal page boundary), a new row is allocated and the initial sequence goal is regressively called with the remaining items. If allocation of a new row fails (due to the vertical page boundary), then a new page is created and either temporally sequenced or linked from the previous page. The initial sequence goal is once again regressively called with the remaining items.

5.1 DESIGN RULES

When defining the temporal and spatial layout of media content, principles of good multimodal design should be employed. The following guidelines have been accumulated from numerous articles and office discussions:

5.1.1 Animation/Dynamic Media

- Use animation to attract user attention.
- Use animation to control user viewing order.
- Do not use animation when other important objects are on the screen. Animation will steal or compete for user attention causing one of the two objects to be missed. Either do not use the animation or properly sequence the objects and effects.

5.1.2 Still images

- Images that are larger, brighter, more detailed, or shown away from others will receive attention first.

- Images require at least 1 second for viewing time.
- Images are viewed before their caption. So if the caption needs to be read first, then the caption should be displayed before the image, and after the reading time has expired, then display the image.
- Captions placed on charts and or diagrams are best placed to the right of the diagram. Users eyes flow left to right more naturally than top to bottom.

5.1.3 Labels

- Gradually reveal labels and other object pointers; e.g., arrows.
- Do not have more than one highlighted object or reveal within the same time period. This will confuse the user as to which object they should focus their attention.
- Use sequencing when doing the reveals. That is, do not do them all at once.
- Use reveals sparingly; i.e., not all captions should be revealed. Only use them to shift attention or to emphasize important information.
- Ensure that an object and its label appear together to improve identification.
- Allow reading time after cueing a label. Once revealed when cued from the speech track, allow sufficient time for the user to read the text. Do not reveal other objects, use animations, or allow other attention stealing affects to be cued before ample reading time for the label is allowed.

5.1.4 Speech

- Allow about a 1-2 sec pause before a speech track starts for a new scene.
- Use the speech track to cue reveals, animations, and highlighting of objects when they are referenced in the speech track.

6. REALIZATION LAYER

The Realization layer is responsible for creating concrete media objects as well as solving their spatial, temporal, and linking constraints. The constraints specified to the Realization layer are known as *intra-dimensional* constraints. Each intra-dimensional constraint defines a relationship among media objects within a single dimension of space, time, or links without crossing over or including other dimensions. For example, a spatial constraint between two media objects may not also include a temporal constraint within the same specification; however, a single media object itself may separately be involved in both a spatial and temporal constraint. The main point is that each set of dimensional constraints are solved independently of the others; i.e., the solution to the set of spatial constraints has no impact on the solution to the set of temporal constraints, and vice versa. The Design layer communicates the media objects along with their intra-dimensional constraints to the Realization layer through a design plan known as the Realization DTD.

6.1 REALIZATION DTD

The design plan is an instance of an XML Document Type Definition, known as the Realization DTD. The goal of the Realization DTD is to be comprehensive enough to capture all of the necessary constraints within the EPG domain, yet be flexible enough to accommodate future requirements. The Realization DTD provides elements, attributes, and containment relationships describing the spatial, temporal, and linking aspects of the desired multimedia presentation independent of a specific document format such as SMIL or HTML. Maintaining this document format independence is important for three reasons. First, it provides the Design layer with a single and well-known medium with which to communicate the design plan to the Realization layer. Second, by not committing to a specific document format, an intelligent choice can be made at runtime as to which output presentation format the Realization layer should choose

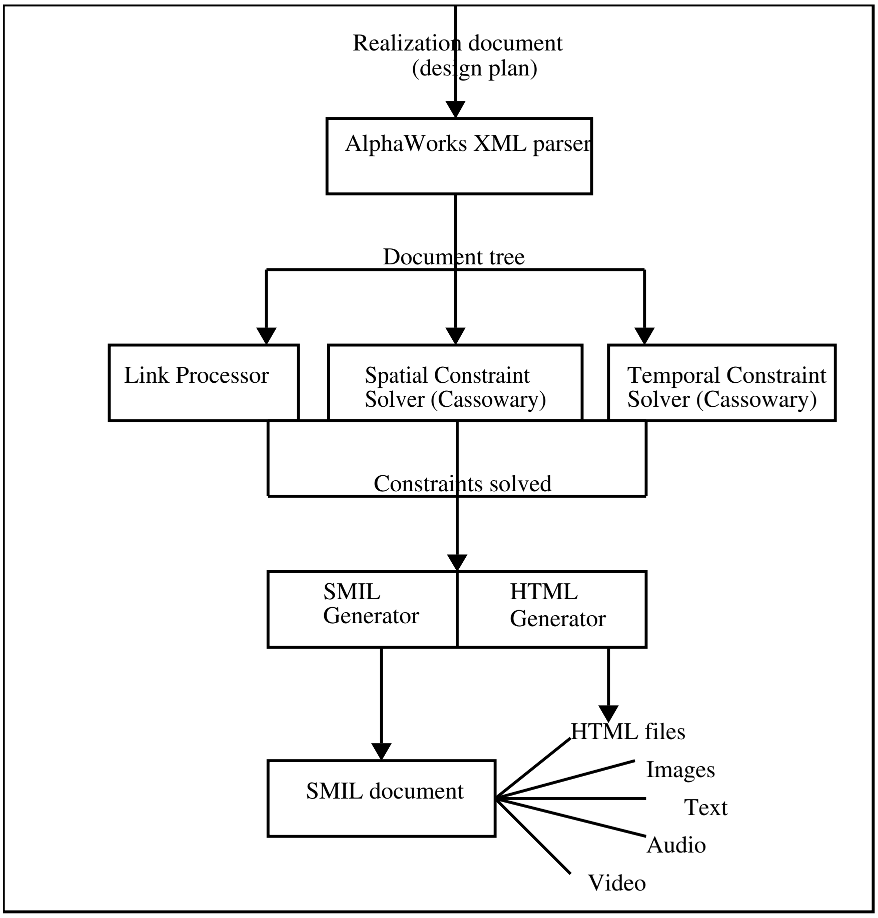

Figure 2. Processing model of the Realization layer.

based on both the type of media and constraints specified within the design plan. For example, a design plan that specifies only text and graphics without any temporal constraints may be realized adequately using only HTML. Third, a complex design plan may require multiple output presentation formats to be generated and then integrated at presentation time. For example, a design plan specifying different text font properties such as family, size, color, and face, can easily be realized by first generating HTML markup for the text, and then integrating this HTML markup into the final presentation using SMIL.

6.2 PROCESSING THE REALIZATION DOCUMENT

To realize the design plan, the Realization layer first parses the XML document instance and creates an in-memory representation of the document tree. The document tree is then traversed to check the semantic consistency of the specifications as well as to separate the different types of relationships; e.g., spatial, temporal, and link relationships. Both the XML parser and DOM implementation were provided by IBM's AlphaWorks XML package implemented in Java. Once separated, both the temporal and spatial constraints are transformed into numerical constraints and subsequently solved using the Cassowary constraint solver. Links are further processed for quick retrieval during output generation. The solutions from the solver are then pass to the SMIL generator which maps the elements, attributes, and relationships from the Realization DTD to elements, attributes, and relationships of SMIL. This entire process is depicted in Figure 2 above.

(Ensure that ImageA remains a rectangle)	**(Ensure that ImageB remains a rectangle)**
ImageA.UL.X = ImageA.LL.X (required)	ImageB.UL.X = ImageB.LL.X (required)
ImageA.UR.X = InageA.LR.X (required)	ImageB.UR.X = InageB.LR.X (required)
ImageA.UL.Y = ImageA.UR.Y(required)	ImageB.UL.Y = ImageB.UR.Y(required)
ImageA.LL.Y = ImageA.LR.Y (required)	ImageB.LL.Y = ImageB.LR.Y (required)
(Ensure that ImageA maintains its width/height)	**(Ensure that ImageB maintains its width/height)**
ImageA.UR.X - ImageA.UL.X = ImageA.Width (strong)	ImageB.UR.X - ImageB.UL.X = ImageB.Width (requirec
ImageA.LR.Y - ImageA.UR.Y = ImageA.Height (strong)	ImageB.LR.Y - ImageB.UR.Y = ImageB.Height (re-

(Ensure that ImageA is left of ImageB)

ImageA.UR.X < ImageB.UL.X (required)
ImageA.UR.Y = ImageB.UR.Y(required)

6.2.1 AlphaWorks XML Parser

IBM's Java AlphaWorks XML package was used to parse the XML document and manipulate the document tree through its DOM implementation. The parser is surprisingly efficient, the documentation is excellent, and no bugs were encountered. As a result, this should be the XML parser used for future projects. The parser can be obtained from http://www.alphaworks.ibm.com.

6.2.2 Cassowary

The Cassowary hierarchical constraint solver was developed by Alan Borning at the University of Washington as an improvement to the DeltaBlue system mainly in terms of performance. Each constraint in Cassowary is defined as a linear equality or inequality among a set of distinct variables. Along with each constraint, the user can also specify the *strength* of the constraint; required, strong, medium, and weak. The set of specified constraints are then solved using a modified Simplex algorithm. The Realization layer transforms each spatial and temporal constraint into one or more linear equalities and/or inequalities and then passes them to Cassowary. For example, the constraint:

Left(ImageA, ImageB, top-aligned).

which states that ImageA is to be placed to the left of ImageB with their top edges aligned is translated into the following constraints. Other constraints are transformed in a similar fashion.

6.3 SMIL GENERATION

The SMIL generator accepts the solved spatial and temporal constraints as well the set of processed links as input to the generation process. The SMIL generator firsts creates the SMIL output file and writes the SMIL document header into the output file. The SMIL generator transforms each media reference (defined with the Reference element) defined in the original realization document into a single SMIL region element. The position of the region is determined by the corresponding solutions to the spatial constraints involving this media reference. Each scene of the realization document is transformed into a single SMIL par element. The duration attribute of the par element as well as the contained media elements is set according to the solutions of the temporal constraint solver involving these elements. However in some cases, the temporal constraints can be directly realized by creating a SMIL seq element and then placing the appropriate media elements as children of this element. Temporal constraints such as before and after are processed in this fashion. A link is transformed into a SMIL a element whose child is the source of the link.

6.4 OVERVIEW OF THE REALIZATION DTD

This section gives a brief overview of the elements used within the Realization DTD. For a more detailed description of the elements along with their associated attributes, see the XML source listed in the appendix.

6.4.1 SRM_Realization
This element represents the root document element and is required to be the first element within any Realization document instance. The element accepts width and height attributes; however, if either is not specified, then the smallest screen space required to display the requested information will be dynamically computed at run-time. This is true of all elements' width and height attributes.

6.4.2 Scene
A scene represents a logically connected set of media that together communicate a single concept to the user. Typically, a scene will represent a single screen of information. This information may be statically displayed to the user (all at the same time), or temporally sequenced in order to create a more effective presentation.

6.4.3 Link
This element represents a hyperlink between two entities. The link may be between two scenes, two media elements, or a media element and a scene. Although this definition of a link provides very basic functionality, it is comprehensive enough to capture the current requirements of the EPG domain.

6.4.4 Element
This element represents a logical grouping of media references along with their spatial, temporal and link relations. An Element element is essentially a Group plus temporal and link information.

6.4.5 Group
This element defines an atomic group of spatially related media elements that cannot be broken apart temporally or with links. Temporal and link information should be added at the Element level.

6.4.6 MediaElement
The MediaElement is an entity representing atomic media elements; such as audio, video, text, and image elements. The attributes of each of these media elements are almost identical to the corresponding SMIL elements. The major difference is that the video element requires a still attribute representing a keyframe to display when the video is not currently playing.

6.4.7 Reference
This element references any MediaElement. The mechanism was designed to allow each media element along with default attributes to be defined only once in the document, but referenced more than once within the document. For each media Reference, the default attributes of the MediaElement can be overridden. The attributes of the Reference element are simply a union of the attributes of each MediaElement.

6.4.8 TextStyle
This element represents a specific style of text to be used. By associating different styles with different text references, the same text can take on different looks within the final presentation.

6.4.9 Spatial Relation
This element defines a spatial relationship between two Elements, Groups, MediaElements, or a Group and a MediaElement. Hierarchical spatial constraints can be defined by specifying constraints between different levels of elements. The result is that complex layouts can easily be achieved by specifying a small set of constraints at each level. Figure 2 demonstrates the ability to specify hierarchical spatial constraints. The spatial constraints are defined using abstract terms such as left, above, right, and below, between the elements. Alignment may also be specified when the related elements are of different sizes. For example, the XML markup:

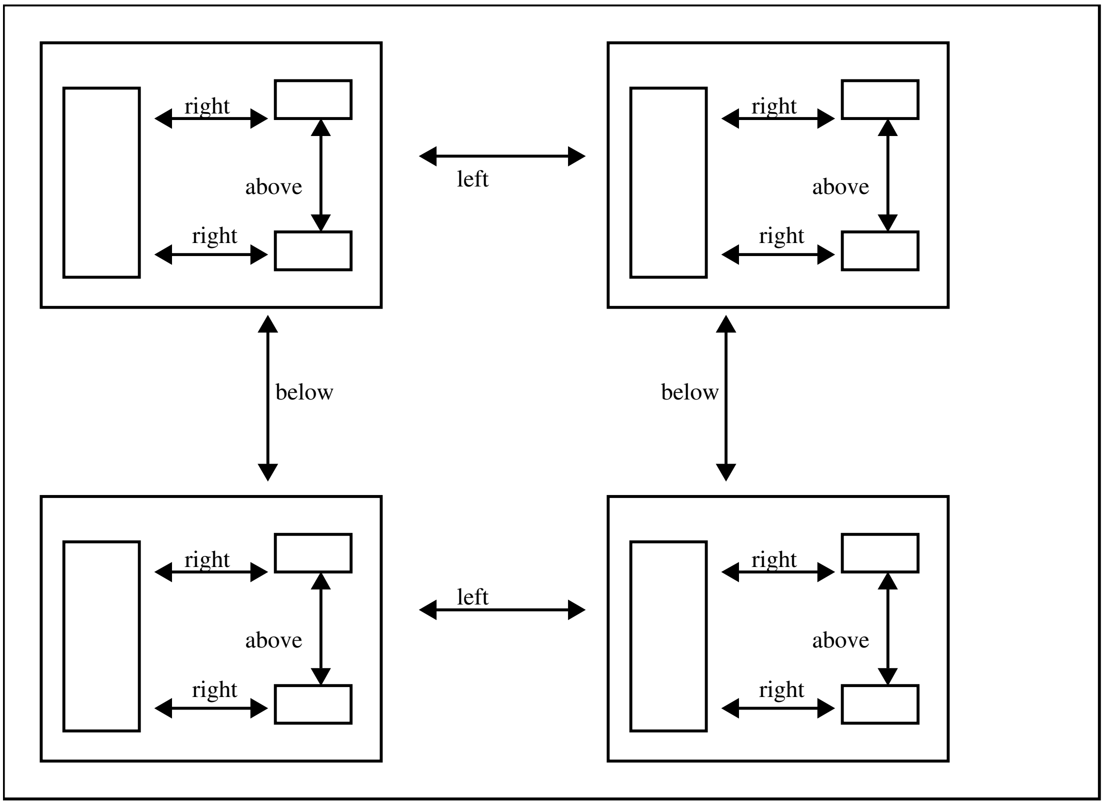

Figure 3. Defining hierarchical spatial relationships.

<Reference id="t1" media-element="text1" />
<Reference id="t2" media-element="text2" />
<SpatialRelation type="above" lhs="t1" rhs="t2" halign="left" />

states that reference t1 is to be placed above reference t2 and their left edges should be aligned.

6.4.10 Temporal Relation

This element defines a temporal relationship between any two Scenes or MediaElements. Although the current relationships only allow before and after, the goal is to support all 13 of the Allen relations. For example, the XML markup:

<Reference id="t1" media-element="text1" />
<Reference id="t2" media-element="text2" />
<TemporalRelation type="before" lhs="t1" rhs="t2" halign="left" />

specifies that the referenced media element t1 is to be displayed before referenced media element t2. This example will be translated into a SMIL seq tag with t1 as the first child and t2 as the second child.

6.5 FUTURE WORK

As the implementation of EPGs in terms of the SRM becomes more complex, the type and amount of information that needs to be communicated through the Realization DTD will also increase. A few possibilities for needed extensions are:

- *Extended temporal relations.* The minimum set of allowable temporal relations should be the infamous 13 Allen relations (before, after, during, meets, contains, etc...).
- *Improved performance*. Given the large number of constraints that can be generated by the Design layer, currently solving them in real-time is not possible. Incremental display of solved constraints

might be one method for giving quick feedback to the user.

- *Improved border support*. Borders are currently specified in a symmetrical fashion; i.e., a border of size 4 means that 4 pixels should be reserved along all four edges of the corresponding rectangle. The ability to independently specify borders along each edge (top, left, bottom, and right edge) of the rectangle would be a nice enhancement.
- *Extensions for visible borders*. Adding support for border display would make the final presentation more aesthetically pleasing.ws

7. REFERENCES

[1] Andre, E., W. Funkier, W. Graf, T.List, A. Scatter, and W. Waster. WIP: The automatic synthesis of multimodal presentations. In M. Mayburry, editor, *Intelligent Multimedia Interfaces*, pages 75-93. AAAI Press / The MIT Press, Menlo Park, CA, 1993.

[2] Duala, S. Feiner, K. McGowan, S. Pan, M. Zhu, T. Hurler, J. Shaw, Y. Fang, and J. Former. Negotiation for automated generation of temporal multimedia presentations. *Proceedings ACM Multimedia*, 1996.

[3] Graf, W.H. Constraint-Based Graphical Layout of Multimodal Presentations. Readings in Intelligent User Interfaces, eds. M.T. Mayburry and W. Waster. Los Altos, CA: Morgan Kaufmann, 1997.

[4] Mayburry, M.T (ed). Readings in Intelligent User Interfaces, Los Altos, CA: Morgan Kaufmann, 1997.

8. APPENDIX - Realization Document Type Definition

```
<?xml version="1.0" encoding="US-ASCII"?>
<!-- Revision: 1.0 DTD for Realization component of the SRM -->
<!-- Date: July 29, 1999 -->

<!-- A shorthand for defining a media element without creating a new tag -->
<!ENTITY % MediaElement "(Text | Audio | Video | Image)">


<!-- Tag: SRM_Realization
Purpose: The main document tag.
-->
<!ELEMENT SRM_Realization (TextStyle*, (%MediaElement;)+, Scene, (Scene | Link | TemporalRelation)* )>
<!ATTLIST SRM_Realization
width CDATA #IMPLIED
height CDATA #IMPLIED
title CDATA #IMPLIED>

<!-- Tag: Scene
Purpose: A single presentation scene.
-->
<!ELEMENT Scene (Element+, (SpatialRelation | TemporalRelation | Link)* )>
<!ATTLIST Scene id ID #REQUIRED
background-music CDATA #IMPLIED
border CDATA "0"
dur CDATA #IMPLIED
width CDATA #IMPLIED
height CDATA #IMPLIED>
```

```
<!-- Tag: Link
Purpose: To hyperlink a source with a destination.
-->
<!ELEMENT Link EMPTY>
<!ATTLIST Link id ID #REQUIRED
src IDREF #REQUIRED
dest IDREF #REQUIRED
text CDATA #IMPLIED
position (TL | TC | TR | BL | BC | BR) #IMPLIED
show (replace | new | pause) #IMPLIED
title CDATA "Link">

<!-- Tag: Element
Purpose: An atomic presentation element. An element must contain a single group of relations among media ob-
jects, subgroups, or both. The size of an Element will be exactly the same as its lone Group node.
-->
<!ELEMENT Element (Group)>
<!ATTLIST Element id ID #REQUIRED
border CDATA "0">

<!-- Tag: SpatialRelation
Purpose: Define a spatial relation between two elements.
-->
<!ELEMENT SpatialRelation EMPTY>
<!ATTLIST SpatialRelation id ID #REQUIRED
type (above | below | left | right) #REQUIRED
halign (left | center | right) "center"
valign (top | center | bottom) "center"
lhs IDREF #REQUIRED
rhs IDREF #REQUIRED>

<!-- Tag: TemporalRelation
Purpose: Define a temporal relation between two media elements/scenes.
-->
<!ELEMENT TemporalRelation EMPTY>
<!ATTLIST TemporalRelation id ID #REQUIRED
type (after | before) #REQUIRED
lhs IDREF #REQUIRED
rhs IDREF #REQUIRED>

<!-- Tag: Group
Purpose: Defines an atomic group of spatially related media elements that cannot be broken apart temporally or with
links. The Width and Height attributes can be hard coded numbers describing a (0,0) anchored rectangle in which
the elements are to be placed (centered). The attributes can also be percentages of the scene width and height. If
the Width and Height attributes are not defined, then the media elements will be positioned as compactly as possi-
ble.
-->
<!ELEMENT Group ( (Group | Reference)+, (Link | SpatialRelation)*)>
<!ATTLIST Group id ID #REQUIRED
border CDATA "0"
height CDATA "0"
width CDATA "0">

<!-- Tag: Reference
```

```
Purpose: To refer to a MediaElement.
-->
<!ELEMENT Reference EMPTY>
<!ATTLIST Reference id ID #REQUIRED
begin CDATA #IMPLIED
border CDATA #IMPLIED
dur CDATA #IMPLIED
end CDATA #IMPLIED
height CDATA #IMPLIED
media-element IDREF #REQUIRED
title CDATA #IMPLIED
width CDATA #IMPLIED
type CDATA #IMPLIED>

<!-- Tag: Style
Purpose: Define a presentation look for text.
-->
<!ELEMENT TextStyle EMPTY>
<!ATTLIST TextStyle id ID #REQUIRED
family (times | courier) "times"
size (10 | 12 | 14) "12"
face (normal | bold | italics) "normal"
border CDATA "0">

<!-- Tag: Text
Purpose: A block of text. The Width and Height atributes suggest a size for the rectangle in which the text must fit.
If not defined, then the text area will be sized based on the surrounding objects.
-->
<!ELEMENT Text (#PCDATA)>
<!ATTLIST Text id ID #REQUIRED
begin CDATA #IMPLIED
border CDATA "0"
dur CDATA #IMPLIED
end CDATA #IMPLIED
height CDATA #REQUIRED
src CDATA #REQUIRED
style IDREF #REQUIRED
title CDATA "Text"
width CDATA #REQUIRED
type CDATA #IMPLIED>

<!-- Tag: Image
Purpose: Define an image component. The Width and Height attributes can be percentages of the original image
size or hard-coded numbers. The image will be scaled accordingly.
-->
<!ELEMENT Image (#PCDATA)>
<!ATTLIST Image id ID #REQUIRED
begin CDATA #IMPLIED
border CDATA "0"
dur CDATA #IMPLIED
end CDATA #IMPLIED
height CDATA #REQUIRED
src CDATA #REQUIRED
title CDATA "Image"
```

```
width CDATA #REQUIRED>

<!-- Tag: Video
Purpose: Define a video segment. The Width and Height attributes are equivalent to the Image sizing attributes.
-->
<!ELEMENT Video (#PCDATA)>
<!ATTLIST Video id ID #REQUIRED
begin CDATA #IMPLIED
border CDATA "0"
clip-begin CDATA #IMPLIED
clip-end CDATA #IMPLIED
dur CDATA #IMPLIED
end CDATA #IMPLIED
height CDATA #REQUIRED
src CDATA #REQUIRED
still CDATA #REQUIRED
title CDATA "Video"
width CDATA #REQUIRED>

<!-- Tag: Audio
Purpose: Defines an audio segment.
-->
<!ELEMENT Audio (#PCDATA)>
<!ATTLIST Audio id ID #REQUIRED
begin CDATA #IMPLIED
clip-begin CDATA #IMPLIED
clip-end CDATA #IMPLIED
dur CDATA #IMPLIED
end CDATA #IMPLIED
src CDATA #REQUIRED
title CDATA "Audio">
```