A Presentation Language for Controlling the Formatting Process in Multimedia Presentations

Frédéric Bes and Cécile Roisin
INRIA Rhône-Alpes
655 avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex, France
{Frederic.Bes,Cecile.Roisin}@inrialpes.fr

ABSTRACT

Multimedia information encapsulated inside documents is more and more specific because its content is specified using domain vocabularies. Their integration in space and time to form a document implies transformation steps to produce "presentation structures".

In this context presentation languages and formatters must be enhanced to cover new needs of rendering such as: multiple output of the same information or dynamic changing of the reader context. These new document models and processing architectures induce new editing and formatting services to be proposed to the author.

This paper describes new presentation properties that can be added to existing presentation languages and that allow the author to express: priorities, more abstract properties and fall-back positions. These properties are used by our formatter in order to provide more adaptive renderings. The architecture of this formatting service is open in order to be used for different presentation systems with different presentation languages. In this paper, we describe our experiment using priorities and optimization requests for temporal formatting.

Categories and Subject Descriptors

I.7.2 [**Document and Text Processing**]: Document Preparation—*Languages and systems,Hypertext/hypermedia,Markup languages*

General Terms

Documentation, Languages

Keywords

multimedia presentation, formatting control, constraints

DocEng'02, November 8–9, 2002, McLean, Virginia, USA.

1. INTRODUCTION

Formatting techniques have been addressed since the beginning of electronic documents because they constitute the heart of any document rendering component. We can find numerous good formatting systems that cover the needs of the tools in which they are included (Latex is the best-known example). Therefore we could consider that formatting problems are solved and no new activity is necessary in that domain. However we can still find situations where authors are not satisfied and more importantly new issues arise that come from two main needs:

- multimedia specific formatting needs: if displaying multimedia objects can be performed by existing formatters, temporal formatting has also to be considered. More complex is the efficient integration of several formatters that must cover all new needs of multimedia presentations such as dynamic effects, animations, interactions, etc..
- adaptation needs: document architectures provide more and more distance between the data sources and the rendered information. These architectures are a new step in separating content from presentation but they aim at providing the reader more a dynamically adapted result depending on his/her needs or the terminal his/she is currently using, etc. In this rendering situations formatters have to involve because the set of results they may provide can be very large and can't be totally controlled by the author.

In this paper, we want to address these formatting problems through the definition of some presentation operators that will allow the author to better control the formatters. The idea is to provide a higher level of presentation operators that can be added to existing languages.

To illustrate this, let's take a classical example used in [31]. The Figure 1 shows an abstract sequencing operator. An author wants to present in a multimedia document a sequence of three elements. With this abstract sequencing operator, the author specifies the sequencial placement of these elements, but the presentation system is free to choose the final presentation of this sequence in the horizontal or vertical spatial orientation, or using a temporal sequence or even an hyperlink sequence.The author may add to this abstract sequence the order of his/her preferences for these alternatives. We will see in section 4 that this example can be used on top of a language like SMIL which is capable of expressing the presentation of spatial or temporal sequences and

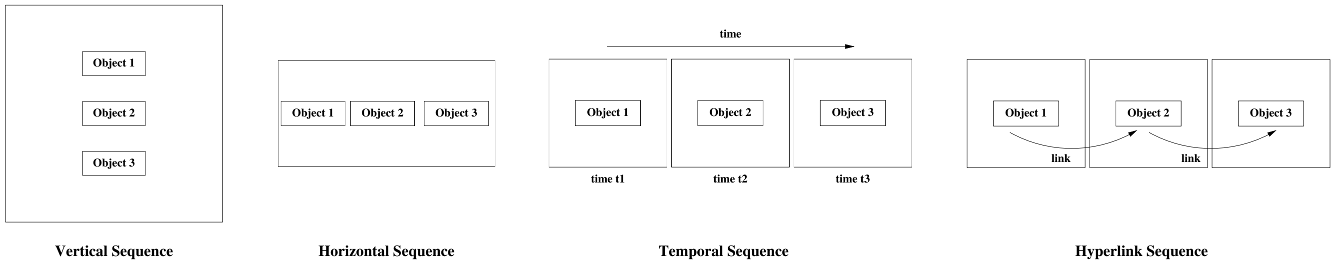

Figure 1: Abstract sequence presented using four different concrete sequences

hyperlink between elements. One can note that SMIL [36] does not allow to manage the spatial sequences relation.

The first interest for an author is to be able to express this kind of abstract sequence. The second interest is to express it with his/her favorites language even if it doesn't manage this operator. The last interest is to give him/her capabilities to control this operator to enable to chose which sequence will be choosen and in which circonstances.

After the study of related works we develop the different presentation properties we introduce in the presentation language and then we show how these properties can enhance the solutions provided by constraint-based formatters. We describe then the general architecture of our proposal and give some results.

2. RELATED WORKS

Since the begining of multimedia document activity, one important direction concerns the improvement of the expressiveness of the multimedia description presentation either in declarative languages or inside multimedia presentation systems. Both approaches have evolved from the formatting of structured document, hypermedia document or graphic objects towards full multimedia application. Related works on formatting control can be classified in two parts.

The first category of works focuses on specific relations, operators or features improvement. Is case, for instance, in the evolution of the CSS [37] or XSL [1] languages or works on spatial layout management [18] (this reference quotes number of works in this domain). This category also includes applications dedicated to specific needs in particular domain like medical [12, 13], tv news report [27], technical documentation [3] or music composition [29]. All these works enable to increase the expressive power of classical multimedia languages and use but they remain limited in providing a generic and global formatting process controls.

The second category group together works on document adaptation and document generation. To enable the authoring of adaptative documents or facilitate the multimedia document generation, some efforts have been done to add flexibility in the formatting process. Flexible definition domains for the presentation attributes and inter-media relations have been used to do this. But this flexibility has brought an important need of control in the formatting process. This control doesn't exist in the actual declarative multimedia languages. These systems [3, 12, 13, 19, 29, 32] can handle a certain level of control on the formatting. They use various technics: constraint logic programming, arithmetic constraint solvers, optimization, decision trees, complex templates or ad'hoc algorithm. But in all these solutions the control of the formatting process is added during the system implementation. For instance, the Cuypers hypermedia system [32] is able to manage abstract multidimensional operators. Even very powerful, the control on the formatting process is fixed by the designer of the system and authors (or readers) have no way to change that control or build their own formatting process.

The interest of our approach is the use of a classical declarative language on which we propose to add some control properties that extend the semantics of its relations.

3. PRESENTATION PROPERTIES FOR CONTROLLING THE FORMATTING PROCESS

Presentation languages allow the author to attach presentation properties to structured (or unstructured) content. Some of these languages are now standardized (CSS [37], XSL-FO [1]) and cover main needs for style definition, object placement in space and time. Another characteristic is the ability to specify dependencies of the same property applied to different objects (for instance inheritance of the character size) or even between different properties (for instance the color of an object can continuously change while that object moves). In this section we want to improve the expressive power of these languages because even with these powerful languages there are some situations where existing formatters cannot provide satisfying results.

3.1 Examples of needs

Let's take the situation of the presentation of lessons about gardening for different vegetable that can have different modalities of rendition: duration (long explanations with videos, text and images or short ones), display (on a large or a small screen), media or content preferences. The lessons have the same structure for every selected vegetable but as the content media are different the resulting renderings can vary. With existing languages and tools, it is very bothering to the author to specify and control all the possible resulting presentations. For instance there is no way to define priorities among the selected contents (for example, plantation steps are more important than damages description) if the total duration exceeds the 10 minutes requested by the reader. This situation may produce documents that do not fit user's intention such as an unbalanced repartition of media types or contents. The user needs to specify general properties that can involve several different properties: for instance priorities can be set to select content when a global criteria has to be maintained (maximum or minimum duration/size of the presentation, media balancing, maximum list size, etc.).

3.2 Principles of our Approach

These simple examples show the need for the definition of presentation operators that will allow the author to specify:

- priorities of content or presentation values
- global and abstract properties
- fall-back procedures when the current computed values do not fit a given criteria.

As these features are independent from any existing language and can be applied to any set of properties, our objective is to propose a set of presentation definitions that can be added to any existing presentation language. However if we want that the formatter computes the "best" solution that fits the intentions of the author (as expressed through these new properties), the language must allow some flexibility in the specification of the properties. So languages that allow relative properties will better benefit from our proposition.

Depending on the kind of formatter that compute that presentation language, our definitions will be either used by a separate formatter or more closely used to control the solving process (see section 4). Figure 2 represents the principles of this approach.

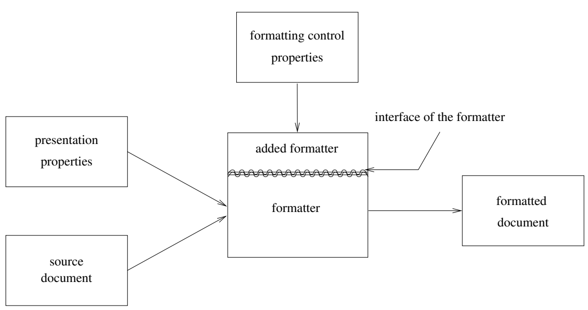

Figure 2: Principle of formatting

3.3 Description of the language

The three types of definitions we have previously introduced are now presented and illustrated through the gardening example.

Priorities : This is the way by which the author will specify preferences used by the formatter. This one will then be able to apply the correct choice among alternatives or to decide to compute again with new values when a formatting failure occurs. Several sets of priorities can be specified, for instance to be used independently for spatial and temporal formatting.

The example described below uses an XML syntax to set structured priorities on elements identified by their *ID*. The media elements refer to gardening steps without vegetable identification (it is supposed that a transformation phase will produce these elements for each selected vegetable). Priorities can be defined by discrete values (attribute *val* of *level* elements) or by position of referenced elements in the list (with increasing or decreasing order). Priorities have themselves *ID* to be used in optimization or fall-back definitions.

In this example the author expresses that information about planting and harvesting is of higher importance than soil preparation, maintenance and diseases. Moreover another priority is defined between media types: videos will be selected only if the formatter will have the possibility (see the fall-back definition).

```
<priorities>
<order id="p1" ordering="increasing"/>
 <order id="p3">
   <level val="1">
     <element ref="Planting-Txt"/>
     <element ref="Planting-Img"/>
     <element ref="Planting-Comment"/>
     <element ref="Harvesting-Txt"/>
     <element ref="Harvesting-Img"/>
   </level>
   <level val="2">
     <element ref="SoilPreparation-Txt"/>
     <element ref="SoilPreparation_Img"/>
     <element ref="Maintenance-Txt"/>
     <element ref="Maintenance-Img"/>
     <element ref="Maintenance-Coment"/>
   </level>
   <level val="3">
     <element ref="Diseases-Txt"/>
     <element ref="Diseases-Img"/>
   </level>
  </order>
  <order id="p2" ordering="increasing">
    <element ref="Planting-Video"/>
    <element ref="Maintenance-Video"/>
    <element ref="Diseases-Video"/>
  </order>
</order>
...
</priorities>
```

Abstract properties : This class of definitions allows to specify a global or partial policy over the formatting system, for instance: dispatching figures regularly over the pages, setting a maximum duration on a sequence, etc. If it is easy to define such high level properties, it is worth noting that they usually can't be solved because they produce NP-Complet problems. So we have prefered to provide some realistic features in the sense that we can provide a formatting solution with the solver we have experimented (see section 4.2): a *balancing* property can be added to a spatial or a temporal sequence structure (so only one axis can be optimized). The policy of a balancing can refer to a priority definition.

For instance the priority *p3* defined above is used to define a formatting policy where the importance of each component of a sequence is proportional to its level of priority:

```
<sequence id="gardening" balancing = "proportional(p3)">
...
</sequence>
```

Fall-back definition : Fall-back strategies are useful when first, the formatter may fall into failure and second, it can detect a failure situation (see next section). Failure may occur when there is no solution that satisfies the set of presentation properties (included global properties). Another case of failure to consider is when an unexpected result is computed. This case occurs when some flexibility is provided by the presentation language either at the level of media properties (size, position, duration specified by intervals of values) or at the composition level (flexible relations such as *on the left*, *before*, *during*). The author must be able to express a global goal that must be added to the set of properties to be computed.

To help the author to limit these two situations we provide a *fall-back* attribute with which she/he can indicate the formatter which changes in the properties set must be applied to reach the solution. The value of that attribute precises successive strategies. Priorities definitions are used for that purpose.

Lets take the gardening examples: the user wants to present gardening advices whose total duration must not exceed 10 mn. Even if the intrinsic presentation properties of media elements cannot be known statically, the author can specify an expected presentation structure with fall-back attributes to indicate the formatters how to manage failure situations: if the sequence lasts more than 10 minutes, the formatter must first reduce the least priority media (p2), then if necessary the main media (p3) and finally if it is not enough it will suppress video media in the order of less importancy. When an author express "*reduction(p2)*", it means that all the media defined in the priority *p2* will be reduced according to the given bounds of their duration. If a media has no flexibility, there is no reduction.

```
<Temporal-sequence id="s1" duration="10min"
          fall-back="reduction(p2);
                     reduction(p3);
                     suppression(p2)">
    ... temporal structure involving the media
    of the different gardening steps
</Temporalsequence>
```

High level presentation with these three definitions : These new properties can help the author to specify high level presentation properties such as the abstract sequence introduced in section 1. Indeed the definition of an abstract sequence, for instance the different steps for gardening, can be specified with a hierarchy of fall-back definitions.

```
<abstract-sequence id="GardeningSteps"
          fall-back="s0;     * spatial sequence
                     s1;     * temporal sequence
                     s2;">   * hyperlink sequence
     <element ref="SoilPreparation"/>
     <element ref="Planting"/>
     <element ref="Maintenance"/>
     <element ref="Harvesting"/>
     <element ref="Diseases"/>
</abstract-sequence>
```

4. A FORMATTING PROCESS UNDER CONTROL

4.1 Existing formatters

In the world of electronic documents, we can find two main families of formatters. The first one is descended from a long experience. It regroups all the main formatters of the current opened presentation languages among them the *tex* formatter [22], the *amaya* [30] editor or *fop* [4] are good representatives. They are powerful and quite efficient. Their formatting process is mainly characterized by the use of the hierarchical box model for formatting text in pages. Each element is associated to one box and can consist of several other boxes. All these boxes are formatted from the top to the bottom of the pages and the position of one box is relative to the previous ones. The formatting of each box is performed by a formatter and additional global formatters are used to control these box formatters in order to realize complex formatting functions. This build a hierarchy of formatters where properties can be inherited but where finally all general dependencies are difficult to be managed. The consequence of this kind of approach is that it is very difficult to develop at the same time as the language without a tedious work. Indeed, each formatting function is an ad'hoc solution developed independently from the others. This makes global approach difficult to integrate too.

The second family of formatters tries to answer to this problem of global process while being interested in the particular requirements of multimedia. It is characterized by the use of constraints techniques. The recent formatters or experiments in this domain have began with previous works in temporal reasonning [24, 2, 14] or direct graphic interaction [34, 28, 9]. One can quote the work of CCSS [5] and CSVG [7] for spatial and visual formatting and Cuypers [32], Isis [21], PPP [3] and Madeus [23] for temporal and multimedia formatting. These techniques, still young, do not still completely satisfy the needs of formatting electronic documents, particularly as regards the scalability or the control of solution. But it manages to demonstrate its interests in global resolution, relative relations handling and multidimension solving.

Several techniques are used in this kind of formatters. First are the local propagation solvers : one-way or multiway constraints techniques. They enable to solve in an effective way simple relations but they are limited by potential cycles between variables and in term of expressiveness [33]. The second one are the techniques based on global linear constraints resolution. The best known solver family is based on the simplex algorithm and has given the two solvers cassowary and qoca [11, 6]. These solvers enable to manage global relations and transform constraint hierarchies [10] into optimization problems by assigning appropriate weights to strenghts. They even have been recently used by Marriott et al. of the Monash University to successfully handle some non linear problems with strong disjunction relations [25]. It also exists a lot of other techniques that have been used to handle particular problems. For example optimization problem according to a given criteria [29], search with numerous backtrack techniques [15] for finite domains [8] ... These last ones can come to complete existing solvers but do not cover enough formatting needs.

Finally, we can evocate two different levels used to check the consistency of a multimedia system : the qualitative and the quantitative checking. These two notions have been introduced with the Allen's relations [2] to manage two different kinds of documents. Qualitative constraints enable to express relative relations between elements whereas quantitative constraints are expressed on the attributes of elements (durations, positions...). Qualitative constraints can be very powerful with disjunction, particularly in document generation [17] where properties are not known statically, but for authoring it has not enough expressive power. The qualitative consistency checking can be used to filter consistency before applying quantitative consistency checking. This is interesting with basic relations, like Allen relations with a strong formal model behind. But two reasons let these kind of relations and consistency checking less and less used. The first reason is that all needs in multimedia presentation can't be described with qualitative relation. For instance, it is impossible to express that less than six videos can be presented in the same page. The second reason is that numerical solvers used to check quantitatives consistency become more efficient [35] and can manage qualitative relations. For instance, with cassowary solver it is possible to use in the same resolution phase the qualitative relation *"x is below y"* and the quantitative relation *"y is 6 pixels below z"*. The first filtering phase becomes useless.

4.2 Choices of formatting techniques

The elaboration of a formatter becomes more and more difficult. The large diversity of solutions that all presentation languages incorporate makes the dependences between dimensions, relations or objects very complex to integrate into the formatters. Beside this increase of complexity, the needs in term of time-performance are always very significant. In addition to these two general observations, our language brings the design of formatters to be concerned with their capacities to manage fallback solutions, priorities and resolution through global criteria. We have seen in the previous section that some of existing formatters are able to handle some features of that language but the difficulty lies in its global nature and in the integration of various techniques. The objective of this paper is to analyse how to deal with existing solutions and provide better formatting results. Therefore we examine now, from a technical point of view, the implications, in terms of formatting needs, of the three concepts presented in section 3.3.

- Priorities
- Global and abstract properties
- Fallback solutions

Priorities can be used in decision making. The first way to use them is in the case of a discrete choice. That is to say, to choose one element among a set of elements (it can be a relation, a media, a property value...) when there is no other decision making information to do it. In a backtrack process, it can give the order to choose the values of the variables in their definition domains. In fallback solutions, it can be used to give the order of alternative objects. The second way is more complex. It consists in the use of a priority order in a more continuous way. For instance to format the duration of a sequence respecting a priority order on objects. The adapted techniques are based on the use of evaluation functions to guide the solving process. This can be done by developing a particular algorithm to solve a given operator. In this case the evaluation function is hard-coded inside the formatter. Some constraint techniques enable to build evaluation functions more dynamically than this solution. This is the case using hierarchical constraints and weighted constraints solvers. Priority of objects or relations can be translated into weights or hierarchical strongs. With these solvers, a formatter can manage more easily a priority order over any existing operator.

Global and abstract properties can be expressed in many ways in our language. The main characteristic for this concept is the global approach it needs. Hence, the implication on techniques to use is to have a global solver at the same level of the relation or expressed criteria. This means two potential architectures for these solvers. The first consists in the use of one global monolithic solver. This solution can be useful for particular formatters, but seems to be less and less realistic in the domain of multimedia formatting. So the second architecture consists in a hierarchy of formatters. In this kind of architecure, some formatters are specialized in the control of global properties over a set of other formatters. A good current exemple is the XSLFO model, but with it, the global consistency is limited to one level. This limitation comes from its basic formatter. They are most cases too specialized and not generic enought to be able to be interestingly parametrized by the upper levels. Indeed, the second main characteristic for formatting global properties is the need of a more abstract approach. A good hierarchical formatter would be one that has basic formatters that could accept external parameters. For instance priorities between the objects it must format or a particular property to preserve during the formatting of the relation this formatter is specialised in. In other words, basic formatters need to be specialized but enought generic !

Fallback solutions are described by the author to cover failure situations during the formatting process. It covers as well presentation structure modifications (when changing or deleting objects or operators) as resolution choices (relaxing some constraints, changing evaluation goals associated to the formatting process of the relations).

What we call fallback is different than backtrack [15, 26]. Both build a way toward the solution but the difference come from the method to build it. With backtrack the built of this way is systematic, applying a fixed heuristic. If it uses constraints relations, a failure can make the system go back and to choose another set of constraints. It is very close with what we have with fallback process. The main difference come from the point that a failure with our fallback is driven directly by what authors have expressed. If they want a classical backtrack apply, they do not express anything different than previous solution (i.e. the new sets of constraints to use in failure cases). This is the default fallback. But if they want particular backtracks to occur in failure cases, they can construct them. So fallbacks are sets of backtracks constructed by authors.

The first requirement here is to be able to detect the failure cases. The same result of the formatting phase can be considered sometimes as a failure case and sometimes not. Let's take a spatial composite object containing three pictures. If one or more variables (presentation properties) take their value out of their definition domain, a failure must be detected. But respecting all the domains of the variables of all the objects is still not sufficient. If the author had expressed a global criteria to define his/her resulting page: for example that he/she does not want pictures with a ratio smaller than half of the original size, or he/she does not want more than 30% of the pages to be filled by pictures, the failure cases are more complex to detect. This meets the optimization parts of the formatter. If we assume that the failure cases can be detected, the formatter must be able to start a fallback solution.

The second requirement in this part is to be able to manage these changes. Changes in terms of resolution in the formatting process and changes in terms of dependencies in the resolution. It meets the well known problem of the incremental authoring [16]. In a global resolution context or multi-way propagation, it can ask to solve again all the system because of the dependences between elements or dimensions. The use of constraint hierarchies can reduce the problem of cycles or of numerous possible ways [20, 33]. In a local resolution context, it consists in the new formatting of some attributes of certain elements without any other implication if there is no cycle. It means to solve a new local plan in the one-way constraint solvers or execute a new local formatting object in the approaches that use a box model like XSLFO. Cycles are problematic and are even more complex than in classical resolution. The problem comes from the dependencies that appear between all the elements and all the operators after having suppressed and/or added some elements and/or some operators. Indeed, removing an object because of the failure of the temporal formatting of an operator can produce an inconsistency in the spatial dimension. In this case the dependence is that when the duration of an element change to zero, its spatial dimension too. But because of the spatial organisation of the document, more than one object can change its spatial attributs to zero and thus produce a new temporal resolution. Constraint-based formatters, dependencies detection are enable, but currently no automatic technique can help efficiently the fallback process to take a decision of what to release and what to keep. In other terms, it would be possible to investigate what can be the optimal fallback position, but it will certainly spend too much time. We prefer to use the basic choices given by the author.

The third requirement is the use of disjunctions. Fallback solutions naturally conduce to use intrinsic notion of disjunctions in the operators themselves. It is easier for an author to express an operator of non-overlapping between some elements than to construct his own relation upon presentation language capabilities to do the same. Except in particular contexts and in simple problems, solving systems that use disjunctive constraints is NP-hard.

4.3 Use of our language

We have described in the previous sections a language allowing a better control of the formatting process. Our goal is not to provide another new presentation language but to provide formatting services. Like the CSS approach for presentation properties, our language tries to bring formatting services in existing presentation languages.

This principle can be illustrated with the example of the abstract sequence seen in introduction. From an existing document written with SMIL, we add the notion of abstract sequence while staying in the SMIL language. The organization of the presentation in SMIL consists in a sequence of three texts, *obj1*, *obj2* and *obj3*, presented during the same time in the temporal parallel operator *trel1*, and in three different regions *r1*, *r2* and *r3*. This organisation describes an horizontal sequence of five secondes.

```
<region id="r1" left="0"  top="10" width="10" height="5"/>
<region id="r2" left="15" top="10" width="10" height="5"/>
<region id="r3" left="30" top="10" width="10" height="5"/>

<par id="trel1">
    <text id="obj1" region="r1" dur="5s"/>
    <text id="obj2" region="r2" dur="5s"/>
    <text id="obj3" region="r3" dur="5s"/>
</par>
```

Attached to this document, in our language, the author has defined his/her abstract sequence. Related to the elements present in the SMIL document, it containts the same sequence *sequence_spat_h* except for the *alt* attribute. During the formatting, if a failure occurs, two sequences are expressed. These alternatives change some parameters of the reference elements. For the spatial vertical sequence, it changes the *region* attribute. One can note that two new regions have been given to express the new spatial organisation. It would have been possible to change the region attributes. For the temporal sequence, the *begin* and the *region* have changed to express that the three texts will be presented at the same spatial position, but one by one.

```
<region id="r4" left="0" top="20" width="10" height="5"/>
<region id="r5" left="0" top="30" width="10" height="5"/>

<sequence_spat_h ref="trel1" alt="trel2,trel3"/>

<sequence_spat_v ref="trel1" id="trel2" alt="true">
  <object1 ref="obj1"/>
  <object2 ref="obj2" region="r4"/>
  <object3 ref="obj3" region="r5"/>
</sequence_spat_v>

<sequence_temp ref="trel1" id="trel3" alt="true">
  <object1 ref="obj1"/>
  <object2 ref="obj2" region="r1"  begin="id(obj1).end"/>
  <object3 ref="obj3" region="r1"  begin="id(obj2).end"/>
</sequence_temp>
```

The simplest way to solve this example is to let the SMIL formatter work. In a failure case, that can be defined by the spatial presentation properties of the document itself, our solver examines the next alternative in the given order, changes the values of the corresponding properties and once again can let the SMIL formatter work.

5. RESULTS AND PERSPECTIVES

To test the interest and the feasibility of our language, we have implemented an example using priorities and alternatives to control the formatting of the temporal sequence of the gardening case described in the section 3.3. We want this sequence to be equilibrated according to the priorities of the elements that it contains. The interest of this example is to be able to manage a global criteria, the duration of the sequence, through three fallback solutions: first the reduction of time presentation of the least priority media, then the reduction of the main media and if necessary the suppression in the order of the priority of the least priority media.

The implementation has been done in a java applet with the Cassowary [6] solver. Following the priority of each media in the sequence, it associates a weight on constraints that bind the media to the values of its duration domain. For its temporal relations, this is a classical temporal sequence. Contrary to the fallback solutions we just have seen in the previous sections, here alternatives are more continuous. Indeed, the elements are the same until the suppression of the first media. The weight conduce the goal evaluation of the Cassowary solver first to maintain the relation between each media with its preferred value of its domain for the duration. When the duration of the sequence is not sufficient, some constraints begin to be relaxed. This example was very interesting to be developed in a graphical and interactive way. The result for this is that the formatting phase answer enables us to change the global duration of the sequence in a mouse displacements. It enables us to easily explore all the possible solutions by visualizing the borderline cases where fallback solutions happened.

These results encourage us to pursue more in detail this experiment towards other elements of our language and more complex formatting situations.

6. REFERENCES

[1] S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman, and S. Zilles. *Extensible Stylesheet Language (XSL) Version 1.0*. W3 Consortium, October 2001. http://www.w3.org/TR/2001/REC-xsl-20011015/.

[2] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[3] E. Andr and T. Rist. Coping with temporal constraint in multimedia presentation planning. In W. Wahlster, editor, *ECAI 96. 12th European Conference on Artificial Intelligence*. John Wiley & Sons, 1996.

[4] Apache. FOP. http://xml.apache.org/fop/.

[5] G. J. Badros, A. Borning, K. Marriott, and P. Stuckey. Constraint cascading style sheets for the web. In *ACM Conference on User Interface Software and Technology*, November 1999.

[6] G. J. Badros, A. Borning, and P. J. Stuckey. The cassowary linear arithmetic constraint solving algorithm. *ACM Transactions on Computer Human Interaction*, 8(4):267–306, December 2001.

[7] G. J. Badros, J. J. Tirtowidjojo, K. Marriott, and B. Meyer. A constraint extension to scalable vector graphics. In *WWW10*, pages 489–498, Hong-Kong, May 2001. ACM.

[8] F. Benhamou and W. Older. Applying interval arithmetic to real, integer and boolean constraints. *Journal of Logic Programming*, 32(1):1–24, 1997.

[9] A. Borning. The programming aspects of thinglab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages ans Systems*, 3(4):353–295, October 1981.

[10] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, September 1992.

[11] A. Borning, K. Marriott, P. Stuckey, and Y. Xiao. Solving linear arithmetic constraints for user interface applications. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 87–96, October 1997.

[12] M. Dalal, S. Feiner, K. McKeown, S. Pan, M. Zhou, T. Hllerer, J. Shaw, Y. Feng, and J. Fromer. Negociation for automated generation of temporal multimedia presentations. In *ACM Multimedia'96*, pages 55–64, 1996.

[13] B. De Carolis and F. de Rosis. The dynamic generation of hypertext presentation of medical guidelines. *the new review of Hypermedia and Multimedia*, 4, 1998.

[14] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[15] E. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29:24–32, 1982.

[16] R. Furuta, V. Quint, and J. Andr. Interactively editing structured documents. *Electronic Publishing – Origination, Dissemination and Design*, 1(1):19–44, April 1988.

[17] J. Geurts, J. van Ossenbruggen, and L. Hardman. Application-specific constraints for multimedia presentation generation. In *International Conference on Multimedia Modeling 2001 (MMM01)*, pages 247–266, Amsterdam, November 2001.

[18] W. H. Graf. Intelligent multimedia layout: A reference architecture for the constraint-based spatial layout of multimedia presentations. *Computer Standards and Interfaces*, 18:515–524, December 1997.

[19] V. Hakkoymaz, J. Kraft, and G. Ozsoyoglu. Constraint-based automation of multimedia presentation assembly. In *ACM Multimedia'99*, pages 500–518, Orlando, October 1999.

[20] H. Hosobe. *Theoretical Properties and Efficient Satisfaction of Hierarchical Constraint Systems*. PhD thesis, University of Tokyo, December 1997.

[21] M. Y. Kim and J. Song. Multimedia documents with elastic time. In *Third ACM International Conference on Multimedia*, pages 143–154, San Fransisco, November 1995. ACM Press.

[22] L. Lamport. *LATEX: a document preparation system*. Addison-Wesley Publishing Company, New-York, 1986.

[23] N. Layada. *Madeus : Systme d'dition et de prsentation de documents structurs multimedia*. PhD thesis, Joseph Fourier, Grenoble, June 1996.

[24] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[25] K. Marriott, P. Moulder, P. J. Stuckey, and A. Borning. Solving disjunctive constraints for interactive graphical applications. In *Principles and Practice of Constraint Programming*, 2001.

[26] O. R. Martinez. Design dependencies within the automatic generation of hypermedia presentations. Technical Report INS-R0205, CWI, June 2002.

[27] B. Merialdo, K. T. Lee, D. Luparello, and J. Roudaire. Automatic construction of personalized tv news programs. In *ACM Multimedia'99*, pages 323–331, Orlando, October 1999.

[28] G. Nelson. Juno, a constraint-based graphics system. In ACM, editor, *GIG-GRAPH'85 Conference Proceedings*, pages 235–243, San Francisco, July 1985.

[29] F. Pachet, P. Roy, and D. Cazaly. A combinatorial approach to content-based music selection. IEEE Multimedia, March 2000.

[30] V. Quint and I. Vatton. An introduction to amaya. *World Wide Web Journal*, 2(2):39–46, Spring 1997. http://www.w3.org/amaya/.

[31] L. Rutledge, B. Bailey, J. van Ossenbruggen, L. Hardman, and J. Geurts. Generating presentation constraints from rhetorical structure. In *11th ACM conference on Hypertext and Hypermedia*, pages 19–28, San Antonio, May 2000.

[32] L. Rutledge, J. Davis, J. van Ossenbruggen, and L. Hardman. Inter-dimensional hypermedia communicative devices for rhetorical structure. In *International Conference on Multimedia Modeling*

2000 (MMM00), pages 89–105, Nagano, November 2000.

[33] M. Sannella, J. Maloney, B. N. Freeman-Benson, and A. Borning. Multi-way versus one-way constraints in user interfaces: Experience with the deltablue algorithm. *Software - Practice and Experience*, 23(5):529–566, 1993.

[34] I. Sutherland. Sketchpad: A man-machine graphical communication system. In *Spring Joint Computer Conference, IFIPS*, 1963.

[35] L. Tardif, F. Bes, and C. Roisin. Constraint for multimedia documents. In *Practical Applications of Constraint Techniques and Logic Programming (PACLP'00)*, Manchester, April 2000.

[36] W3C SYMM Working Group. *Synchronized Multimedia Integration Language (SMIL 2.0)*. W3 Consortium, August 2001. http://www.w3.org/TR/2001/REC-smil20-20010807/.

[37] W3C Working Group on Cascading Style Sheets and Formatting Properties. *Cascading Style Sheets, level 2*. W3 Consortium, May 1998. http://www.w3.org/TR/1998/REC-CSS2-19980512/.