XConnector: Extending XLink to Provide Multimedia Synchronization

Débora C. Muchaluat-Saade[1,2], Rogério F. Rodrigues[1], Luiz Fernando G. Soares[1]
[1]TeleMídia Laboratory – Departamento de Informática – PUC-Rio
R. Marquês de São Vicente, 225 – Gávea – 22453-900, Rio de Janeiro, RJ, Brazil
[2]Departamento de Engenharia de Telecomunicações – Universidade Federal Fluminense (UFF)
R. Passo da Pátria, 156 – São Domingos – 24210-240, Niterói, RJ, Brazil
debora@telemidia.puc-rio.br, rogerio@telemidia.puc-rio.br, lfgs@inf.puc-rio.br

ABSTRACT

This paper proposes XConnector, a language for the creation of complex hypermedia relations with causal or constraint semantics. XConnector allows the definition of relations independently of which resources are related. Another feature is the specification of relation libraries, providing reuse in relationship definition. The main goal is to improve linking languages or the linking modules of hypermedia authoring languages in order to provide multimedia synchronization capabilities using links. Following this direction, an extension to W3C XLink is proposed, incorporating XConnector facilities.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications – *information browsers.*

General Terms

Languages, Standardization.

Keywords

Multimedia synchronization, links, hypermedia connector, XLink, XConnector.

1. INTRODUCTION

Hypermedia authoring languages must be rich in their semantic capabilities to express different kinds of relationships among document components. Besides the traditional referential relationship, they should provide context relationships, which capture the hierarchical structure of a document, such as a book and its chapters, chapters and their sections and so on; and synchronization relationships, which define both temporal and spatial ordering of document components. A discussion about other types of relations can be found in [18].

Regardless the possibility of including objects (images, applets, scripts, etc.) in web pages, the HTML language only provides simple uni-directional referential relations, represented by content-embedded links. Due to this limitation, recent efforts have been done by W3C in order to provide means for creating more sophisticated relationships among WWW resources. These efforts are present in SMIL [16] and XLink [22] recommendations. Both of them can represent referential and synchronization relations, although following different approaches. SMIL provides compositions with specifc semantics for creating synchronization relations, besides uni-directional single-headed links to capture simple referential and temporal relations. XLink allows the creation of more sophisticated referential relations and simple temporal relations using links.

Hypermedia relations can be expressed by either compositions or links. Defining synchronization relations using compositions makes the authoring task easier, since the author can specify with a single composition what would be alternatively specified using several links [14]. However, languages usually offer a limited set of compositions to capture synchronization relations. In SMIL, for example, there are only three composition types, which are parallel, sequential and exclusive. As a consequence, complex relationships must be built through a hierarchy of basic compositions. This authoring approach obliges the document structure to match its presentation structure, what may not be necessarily desirable. Moreover, this may impair the approach inherent advantage of facilitating the authoring process. The use of links for specifying synchronization relations allows reserving composite elements for context relationship specification [2], separating the document presentation structure from its logical one. Of course, compositions can still be used for synchronization specification when presentation and logical structure are the same.

W3C XLink has some limitations as a hypermedia linking language. This paper addresses these limitations proposing an XML language, called **XConnector**, which provides the creation of complex referential and multimedia synchronization relation types. XConnector uses the concept of hypermedia connector, already introduced in [11], which allows the relation specification independent of which resources[1] are related. Different links of the same type can be created reusing the same connector and defining different sets of participating resources. Besides presenting XConnector, this paper shows how XLink can be extended to use

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

[1] A resource is any addressable unit of information that is part of a hypermedia document. In this text, the word resource is used as a synonym for: a document component or node; or a node anchor (portion of the content of a node); or a node attribute.

XConnector facilities. Using the concept of connectors, the proposed XLink extension will provide more expressiveness but will also maintain XLink facility of use. Although this paper proposes an extension to XLink, XConnector could be used to extend the linking modules of other hypermedia languages, such as XHTML [21] or SMIL.

In short, the main contributions of this paper, comparing to previous published work [11] are the detailed definition of constraint hypermedia connectors, the definition of XConnector as an XML language, and the extension to XLink, which is a completely new proposal.

The paper is organized as follows. Section 2 briefly describes the W3C XLink features and discusses its limitations as a hypermedia linking language. Section 3 presents the XConnector proposal and shows how XLink can be extended to use it. Section 4 compares the proposal to related work and Section 5 is reserved to conclusions and future work.

2. W3C XLINK LANGUAGE

XLink [22] provides two kinds of links, simple links (*simple*-type elements), which are similar to uni-directional HTML hyperlinks, and extended links (*extended*-type elements), which offer full XLink functionality. Extended links allow describing sophisticated relations with an arbitrary number of participating resources. An extended link specifies a set of participants and a set of traversal rules. An example is shown in Figure 1.

```
<courseload>
   <person xlink:href="students/peterkorb60.xml"
           xlink:label="student"/>
   <person xlink:href="students/patjones62.xml"
           xlink:label="student"/>
   <person xlink:href="profs/jaysmith7.xml"
           xlink:label="prof7"/>
   <course xlink:href="courses/cs101.xml"
           xlink:label="CS-101"/>
   <go xlink:from="student" xlink:to="CS-101"
       xlink:show="replace" xlink:actuate="onRequest"/>
   <go xlink:from="prof7" xlink:to="CS-101"
       xlink:show="replace" xlink:actuate="onRequest"/>
   <go xlink:from="CS-101" xlink:to="prof7"
       xlink:show="replace" xlink:actuate="onRequest"/>
</courseload>
```

Figure 1. XLink extended link example

Participants may be local resources (*resource*-type elements) or remote resources (*locator*-type elements), considering the resource where the link is defined. Each participant of an extended link has a *label* attribute that is used in the specification of traversal rules. The example shown in Figure 1 defines four remote participants, three of them are *person* elements and one is a *course* element.

An XLink traversal rule (*arc*-type element) relates a source label (*from* attribute) to a target label (*to* attribute), besides specifying when navigation must occur (*actuate* attribute) and how target node presentation must be done (*show* attribute). Since several participants may share the same label, it is possible to define multipoint relationships. Thus, traversal rules determine which participants are source (*starting* resources) or target (*ending* resources) in a link. Navigation may be triggered by user request (*actuate="onRequest"*) or may be done automatically when the starting resource is loaded (*actuate="onLoad"*). The ending resource presentation may be done in a new window (*show="new"*), may substitute the starting resource presentation (*show="replace"*) or may even be embedded in the presentation (*show="embed"*). Since XLink extended links may define several traversal rules, each extended link represents a set of relationships (hypermedia links) among participants, instead of just one. The example of Figure 1 shows three *arc*-type elements named *go*, where one of them represents a multipoint relationship, since label "student" is shared by two participants.

XLink also provides semantic attributes (*role*, *arcrole* and *title*) to describe the meaning of resources within the context of a link, but they are out of the scope of this paper.

XLink allows defining links that reside in a location separate from the linked resources. This is useful when resources are read-only or when they offer no way to embed linking constructs, as for example creating links anchoring on video content files. In addition, XLink provides the definition of link repositories, called *linkbases*, facilitating link management by gathering several linking elements.

Despite the fact that XLink allows describing more sophisticated relationships than simple HTML hyperlinks, it also has limitations:

- XLink only provides the specification of causal relations: if a condition is satisfied, an action must be fired. There is no support for defining constraint relationships among resources. Constraint relations have no causality involved. Consider, for example, a constraint specifying that one participant must finish its presentation at the same time another participant begins its exhibition. The occurrence of the presentation of one participant without the occurrence of the presentation of the other also satisfies the constraint, which specifies that, if and only if these two participants are presented, their end and beginning times have to coincide.
- XLink navigation may be triggered only by user request or automatically when the starting resource is loaded. There is no support for other conditions, such as "when a resource content presentation ends".
- Although one can create multipoint links using XLink, each traversal rule only relates a unique condition type for all source participants, determined by its *actuate* attribute value. Also, a unique action type is defined for all target participants, which is "start the presentation of the target label considering the *show* attribute value". There is no way of defining a multipoint relation that specifies composite conditions and actions. For example, one might need to specify: "if *A* is being presented and an anchor of *B* has been selected, pause the presentation of *C* and stop the presentation of *A*".
- XLink does not allow the specification of spatial relations, such as a constraint defining that "two participants must be vertically aligned" or a causality specifying "if *A* is moved to a specific position, then start the presentation of *B*".

Moreover, XLink does not allow the definition of traversal rule types independent of the definition of their participants. This prevents relation definition reuse in distinct traversal rules with the same traversal behavior and, as a consequence, also introduces another limitation:

- XLink allows the definition of link repositories but does not provide the definition of traversal-rule type repositories. This feature becomes important when complex types of traversal rules may be defined.

3. EXTENDING XLINK TO PROVIDE MULTIMEDIA SYNCHRONIZATION

In contrast to XLink, this proposal detach the definition of a relation from the definition of a relationship. First, it introduces an entity to define the relation type, called *xconnector,* following the concept of hypermedia connector proposed in [11]. Then, as an xconnector does not specify participating resources, another entity is needed to define the relationship, which is the *extension to XLink*. The extension is very similar to XLink extended links, although it has a richer semantics. The improvements come from using XConnector and also from the new way participant labels are bound to traversal rules, as follows.

An xconnector defines the role of each participant and how they interact, but it does not specify who they are. Its purpose is similar to an XLink *arc*-type element, but it has a richer semantics and is defined as an independent element and not as a child element of an extended link.

The extension to XLink proposes a new *arc*-type definition that can comprise the current XLink *arc*-type definition or, instead, refer to an xconnector and associate its roles to participant labels. This is indeed a significant difference from XLink, allowing the definition of the relation type separated from the relationship. This facility allows reusing the relation type definition and, as a consequence, providing traversal-rule type databases besides linkbases like XLink. Moreover, this feature also facilitates link maintenance, since modifying an xconnector automatically updates all links referring to it.

Increasing the expressiveness of the linking language without losing its facility of use is another goal of the proposal. Thus, the idea is to have expert users defining xconnectors, storing them in libraries and making them available for others to create links.

Figure 2 shows an xconnector named *R*, representing a relation with three different roles, which means three different types of participants. The figure also illustrates two different links, l_1 and l_2, using *R*. A link is defined by an xconnector and by a set of binds relating xconnector roles to resources (node anchors or node attributes) [11]. While xconnector defines the relation type, the set of binds specifies the interacting resources. Link l_1 defines three binds connecting anchors/attributes of nodes *A*, *B* and *C* to the roles of xconnector *R*. Link l_2 also defines three binds, but it connects a different set of nodes (*B*, *C* and *D*). Links l_1 and l_2 define different relationships, as they relate different sets of nodes, but represent the same type of relation, as they use the same xconnector. One may consider links as instantiations of xconnectors.

One might ask why we need to extend XLink to provide additional synchronization facilities if we already have XHTML, SMIL and even the inclusion of SMIL features into XHTML through the XHTML+SMIL Profile [20]. Two main reasons can be highlighted:

- The first reason is one of the motivations that have raised the W3C XLink specification, which is to "express links that reside in a location separate from the linked resources" [22]. This cannot be done with XHTML, SMIL or XHTML+SMIL.
- The second one is being able to express complex n-ary synchronization relationships using links.

If we combine XHTML+SMIL and XLink, we may be able to specify some complex n-ary synchronization relationships using SMIL facilities, but we certainly will not be able to store them in a location separate from the linked resources.

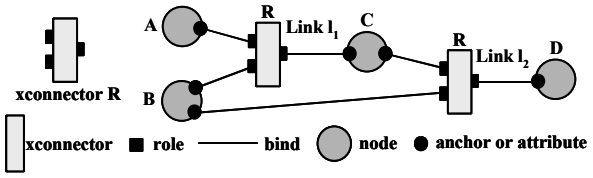

Figure 2. Example of links using the same xconnector *R*

The main feature of XConnector is that it can be used in conjunction with hypermedia linking languages regardless of how they define their document components. The usefulness of extending XLink with XConnector facilities is that it improves XLink expressiveness, maintains the possibility of defining relationships separately from the linked resources and adds the possibility of having relation databases besides linkbases.

3.1 XConnector

XConnector is an XML-based language providing the definition of xconnector elements. Each xconnector defines a set of *roles* (that will be played by participating resources in a relationship) and how they actually interact, through a child element called *glue* [11]. In order to capture causal and constraint relations, xconnectors are specialized in causal and constraint types. In both kinds of xconnectors, the glue describes how roles interact, specifying the causality or constraint semantics. The complete XML Schema [15] definition of XConnector is available at http://www.telemidia.puc-rio.br/specs/xml/XConnector.xsd.

3.1.1 Roles

The definition of roles is based on the concept of event. As stated in [13], an event is an occurrence in time that can be instantaneous or can occur over some time period. Reference [8] compares links and events as different approaches for defining activation and deactivation information of document components. This proposal combines both, since xconnectors are defined as relationships among events that happen over document components. XConnector basic types of events are *presentation*, *mouseClick*, *mouseOver*, *focus*, *prefetch* and *attribution*. This set may be extended to include other relevant event types.

States and transitions of an event state machine, as shown in Figure 3, can be used for authoring xconnector roles. For *presentation*, *prefetch* and *attribution* events, it is very useful to allow them to be paused or resumed. On the other hand, pausing or resuming *mouseClick*, *mouseOver* and *focus* events does not make much sense. In this case, a simpler event state machine is considered, with just the *prepared* and *occurring* states and transitions between them. The *prefetch* event state machine is different from the others, because when this event naturally ends, it does not return to the *prepared* state, but it goes to the *finished* state. In fact, the *prefetch* event must happen before the occurrence of any other event type of a same resource. Thus, when a *prefetch* event reaches the finished state, it actually causes the creation of other state machines controlling events of the same resource, starting from the *prepared* state.

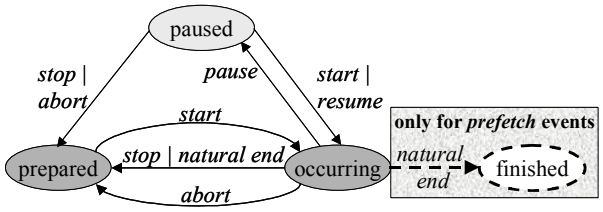

Figure 3. Event state machine for authoring xconnector roles

XConnector events, except *prefetch*, have an associated attribute called *occurrences*, which counts how many times its state changes from *occurring* to *prepared*. *Presentation* and *attribution* events also have another attribute named *repeat*, which indicates how many times they must still occur.

A *role* defines an id, an event type and its cardinality. A role *id* does not need to be a unique value inside the whole document, since a role definition cannot be reused in different xconnectors. The role *event type* refers to one of the event types previously defined. The role *cardinality* specifies the minimal and maximal number of resources that may play this role (number of binds) when this xconnector is used for creating a link, as will be defined later. If a role event type is *attribution*, the role must also define which is the corresponding resource attribute name.

Roles are specialized in action roles, condition roles and property roles. Different types of roles are used depending on the xconnector type. In constraint xconnectors, only property roles are allowed. In causal xconnectors, any type of role may be used.

Action roles capture actions that can be executed over resources. Types of actions are illustrated in Figure 3 by labeled arcs causing transitions in the event state machine (except *natural end*). Besides the action type, an action role may define a delay to be waited before the action is executed; values to be assigned to participant attributes if the role event type is *attribution*; and a value to be assigned to the event *repeat* attribute, if the event type is *presentation or attribution*[2]. An action role example is "pause the presentation of a resource after a three-second delay". Action roles may also define a *show* attribute, having the same meaning of the XLink *show* attribute, whose possible values are "new", "embed" and "replace". The behavior of the starting resources when a link is followed can almost always be modeled by another action of a multipoint link. One exception is when the *show* attribute has the "embed" value, justifying why it was included in XConnector.

In causal xconnectors, conditions must be satisfied in order to trigger actions. Conditions are captured by the *condition role type*, which defines a logical expression evaluating event states, event state transitions or attribute values. When evaluated, a condition returns a boolean value. Conditions may be simple or compound.

A *simple condition* can be a comparison using comparators "eq" (=), "dif" (≠), "lt" (<), "lte" (≤), "gt" (>) or "gte" (≥), testing an event state transition, an event state value, an event attribute value such as *repeat* and *occurrences*, as explained previously, or even a resource attribute value. When the expression is evaluated over an event state transition, the condition is considered to be true only at the moment the transition occurs.

Any condition may be negated and a *compound condition* consists of a binary logical expression involving two other conditions over the same event and based on the operators "and" or "or". A compound condition role example is "the presentation of a participant finished for the second time", which would be specified as "[(*event-type* = "presentation"), ((*transition* = "stops") AND (*occurrences* = "2"))]".

The third and last type of role is the *property role type*. A property can be an event state value, the time instant an event state transition occurs, an event attribute value or a resource attribute value. While a condition always returns a boolean value when evaluated, a property returns any type of value, depending on the property type. Event transition and attribute value properties may specify an offset that can be added to the property result. For example, a property may specify "five seconds after the time instant a presentation event stops" or "the screen vertical position plus 50 pixels".

3.1.2 Glue

As previously mentioned, an xconnector is defined by a set of roles and a glue, where the glue specifies how its roles interact. Every xconnector role must be used in the glue. A *constraint xconnector* has a *constraint glue,* which defines a property expression relating property roles. A *causal xconnector* has a *causal glue*, which defines both a *trigger expression*, relating condition and property roles, and an *action expression*, relating action roles. When the trigger expression is satisfied, the action expression must be executed.

A property expression can be simple or compound. A *simple property expression* can compare either property roles of the same type or a property role to a value of the same type of the property result. The comparison can use comparators "*eq*" (=), "*dif*" (≠), "*lt*" (<), "*lte*" (≤), "*gt*" (>) or "*gte*" (≥). For example, suppose one property role *P* specifying the *starts* transition of a presentation event and a property role *Q* specifying the *stops* transition of a presentation event. If a simple property expression S_1 defines that "*P = Q"*, S_1 will be evaluated as true if the first presentation event starts at the same time the other presentation event stops. As another example, suppose a property role *H* specifying the screen horizontal position attribute value. If a simple property expression S_2 specifies that "$H \geq 100$", S_2 will be evaluated as true if the horizontal position of a participant playing the *H* role is greater than 100. Any property expression may be negated and a *compound property expression* consists of a binary logical expression, based on the operators "*and*" or "*or*", involving two property expressions. Although property expressions can be used in causal xconnectors, their main utility is in the specification of constraint xconnectors, as follows.

Table 1 illustrates a constraint xconnector expressing a spatial synchronization relation specifying "two nodes must be horizontally aligned by their tops".

A *trigger expression* can be simple or compound. A *simple trigger expression* refers to the *id* of a condition role. A *compound trigger expression* consists of a binary logical expression, based on the operators "*and*" or "*or*", involving either two trigger expressions, or one property and one trigger expression. Any trigger expression may be negated and may specify minimal and maximal delays to its evaluation. For example, given that a trigger expression *C* is true at

[2] As a future work, those delay and attribute values should be parameterized, allowing the same xconnnector to be used defining different parameter values.

instant *t*, *C'* defined as *C* with *min-delay*="t_1" and *max-delay*="t_2" is true at the interval [$t+t_1$, $t+t_2$].

Table 1. Example of constraint xconnector

Role type and id	Event type (cardinality)	attribute name	offset
Property P_1	*attribution (1,1)*	*top*	*0*
Property P_2	*attribution (1,1)*	*top*	*0*

Glue type	Property expression
Constraint	$P_1 = P_2$

Compound trigger expressions can relate any number of condition and property roles. However, one restriction is necessary to guarantee causal link consistency. Every trigger expression may only be satisfied at an infinitesimal time instant, requiring that at least one condition role of each causal xconnector define an event state transition condition.

An *action expression* can also be simple or compound. A *simple action expression* refers to the *id* of an action role. A *compound action expression* consists of a binary expression, based on the operators "*par*", "*seq*" and "*excl*", involving two action expressions. Parallel (*par*) and sequential (*seq*) compound actions respectively specify if the execution of actions must be done in any order or in a specific order. Exclusive (*excl*) compound action specifies that only one of the actions must be fired[3].

When the maximal cardinality value of a role is greater than one, several participants may play the same role. In this case, a *qualifier* must be specified each time this role is used in the glue expressions. Table 2 presents possible qualifier values.

Table 2. Role qualifier values

Role type	Qualifier	Semantics
condition	*all*	all conditions must be true
condition	*any*	at least one condition must be true
property	*all*	all properties must be considered
property	*any*	at least one property must be considered
action	*par*	all actions must be executed in parallel
action	*excl*	only one of the actions must be executed

Table 3 illustrates a causal xconnector example expressing a temporal synchronization relation. The xconnector specification is interpreted as "if a group of participants is being presented (C_1) and another participant is selected (C_2), stop the presentation of a group of participants (A_1) and start the presentation of another participant". In order to stop the presentation of the same group of participants that played role C_1, a link using this xconnector must create two binds for each participant in the group, one to role C_1 and another to A_1.

Table 3. Example of causal xconnector

Role type and id	Event type (cardinality)	Condition	Action
Condition C_1	*presentation (1, unbounded)*	*state=occurring*	
Condition C_2	*mouseClick (1, 1)*	*transition=stops*	
Action A_1	*presentation (1, unbounded)*		*stop*
Action A_2	*presentation (1, 1)*		*start*

Glue type	Trigger expression	Action expression
Causal	*all(C_1) AND C_2*	*seq(par(A_1), A_2)*

[3] When only one of the actions must be executed, the document formatter (browser) should decide which one to be performed or it may ask the user to do it.

3.2 Extending XLink to use XConnector

In order to incorporate XConnector facilities into XLink, *arc*-type elements must be modified. Attributes *from*, *to*, *show* and *actuate* become deprecated and a new attribute, named *xconnector*, used for identifying a valid xconnector URI, is introduced. Moreover, *arc*-type elements must define a set of *bind*-type child elements. Each *bind*-type element specifies which participating resource plays a specific role of the *xconnector* used by the *arc*-type parent element. A *bind*-type element must have a *label* attribute to identify an XLink participant label and a *role* attribute to identify the id of one of the *xconnector* roles. Note that the semantics of this *role* attribute is different from the XLink homonym attribute. This new *arc*-type element specification gives more flexibility to link definition since the number of labels associated to the same traversal rule is not limited to two, as it is in XLink.

Although the extension to XLink maintains XLink *arc*-type attributes specifying traversal behavior, let us show an example of how current XLink semantics would be represented using the new proposal. In order to represent the example given in Figure 1 (Section 2) using XConnector, each traversal rule defining different *show* and *actuate* attribute values has to be converted to an xconnector. In that example, all *go* elements have the same values for these attributes, so they are represented by the same xconnector. Figure 4 illustrates the definition of this causal xconnector identified by "xlink-replace-onRequest".

```
<xconnector id="xlink-replace-onRequest"
   xsi:type="CausalHypermediaConnector" >
   <condition-role id="from" event-type="mouseClick"
     max="unbounded">
      <condition xsi:type="EventTransitionCondition"
        transition="stops"/>
   </condition-role>
   <action-role id="to" event-type="presentation"
     action-type="start" show="replace" max="unbounded"/>
   <glue>
      <trigger-expression
        xsi:type="SimpleConditionExpression"
        condition-role="from" qualifier="any"/>
      <action-expression
        xsi:type="SimpleActionExpression"
        action-role="to" qualifier="excl"/>
   </glue>
</xconnector>
```

Figure 4. xconnector "xlink-replace-onRequest"

Note that, as XLink labels can be used by several resources, the *max* cardinality attribute of each role in Figure 4 is set to "unbounded". Although several resources may share the same label in XLink, actually a traversal rule specification generates several point-to-point arcs. Since the recommendation does not constraint the application behavior in this case and xconnector authors may choose the desired behavior in this situation, the qualifier for action role "to" was chosen to be "*excl*" in the example.

Figure 4 presents the specification of the traversal-rule type using XConnector. In order to complete the link definition, Figure 5 presents the specification of the new XLink extended element, reusing the same xconnector "xlink-replace-onRequest" in all

traversal rules. Note that, since we are now able to define different labels as sources of the same traversal rule, in the new XLink example, we just needed two *go* elements, instead of three, preserving the semantics of the example illustrated in Figure 1.

```
<courseload>
   <person xlink:href="students/peterkorb60.xml"
           xlink:label="student"/>
   <person xlink:href="students/patjones62.xml"
           xlink:label="student"/>
   <person xlink:href="profs/jaysmith7.xml"
           xlink:label="prof7"/>
   <course xlink:href="courses/cs101.xml"
           xlink:label="CS-101"/>
   <go xconnector:xconnector="xlink-replace-onRequest">
      <bind xconnector:label="student"
       xconnector:role="from"/>
      <bind xconnector:label="prof7"
       xconnector:role="from"/>
      <bind xconnector:label="CS-101"
       xconnector:role="to"/>
   </go>
   <go xconnector:xconnector="xlink-replace-onRequest">
      <bind xconnector:label="CS-101"
       xconnector:role="from"/>
      <bind xconnector:label="prof7"
       xconnector:role="to"/>
   </go>
</courseload>
```

Figure 5. Extended XLink/XConnector example

Comparing the XLink example shown in Figure 1 to its corresponding extended XLink/XConnector definition shown in Figures 4 and 5, one might think that the complexity to use XConnector is much higher than XLink. If you consider that most users would have to define xconnectors, this conclusion would make sense. However, the idea is to have expert users to write xconnector libraries representing several types of hypermedia relations, such as the one shown in Figure 4. Common users will just need to use them to create links, writing XML code like the one shown in Figure 5, which is very reasonable.

The previous example showed a simple referential relationship among participants in order to illustrate how *arc*-type element syntax was modified in the XLink extension proposal. The following examples show how extended XLink can provide complex multimedia synchronization among link participants.

Suppose that an XLink defines *A*, *B*, *C* and *D* as its participating resource labels (label *C* refers to *anchor1* of participant *C*). If we would like to create a synchronization relation using the causal xconnector defined in Table 3, the same XLink must define an *arc*-type element identifying this xconnector URI and the set of binds shown in Figure 6. This traversal rule represents the following temporal synchronization relation: "if participants *A* and *B* are being presented and *anchor1* of participant *C* is selected, stop the presentation of *A* and *B* and start the presentation of participant *D*". In the figure, participants *A* and *B* are drawn twice to represent that they play both condition and action roles.

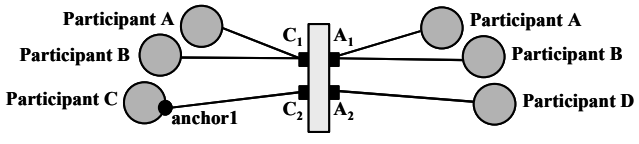

Figure 6. Set of binds of an XLink temporal relationship among participants *A*, *B*, *C* and *D*

As another example, we could create another *arc*-type element relating participants *A* and *B*, using the xconnector defined in Table 1, where the set of binds would be the ones shown in Figure 7. This link represents the following spatial synchronization relation: "participants *A* and *B* must be horizontally aligned by their top attributes".

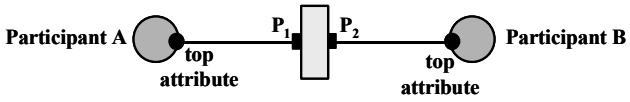

Figure 7. Set of binds of an XLink spatial relationship between participants *A* and *B*

Note that we may have consistency problems when xconnectors are used to create links. One example is when the number of XLink participants sharing the same label is lower than the minimal or greater than the maximal cardinality value of a role bound to this label. Another example is when someone misused an xconnector to create links and did not define binds correctly for all its roles. Problems may also occur when a link refers to an xconnector URI that became invalid for any reason. In all theses cases, document formatters must identify inconsistencies.

Some advantages of extending XLink in order to incorporate the concepts of XConnector can now be highlighted more clearly:

- Enhanced expressiveness for the linking language – XConnector allows the definition of new types of XLink traversal rules representing multimedia spatio-temporal synchronization relations with causal or constraint semantics:
  - traversal rules can define truly multipoint relations specifying composite conditions and actions, since glue expressions may relate any number of roles;
  - besides usual *presentation* and *mouseClick* events, other kinds of events may be used to define conditions and actions, such as content *prefetch*;
  - the whole life cycle of an event, as defined by its state machine, may be used in the definition of condition and actions, giving authors more flexibility and expressiveness to specify relations.
- Increased language reuse – An already defined type of traversal rule, represented by an *xconnector*, can be reused by several *arc*-type elements in the same XLink extended link or even inside different extended links.
- Facility of link maintenance – when an xconnector is modified, all links referring to it are automatically updated.
- Maintenance of facility of use despite the enhanced expressiveness – Applying the same idea of linkbases to xconnectors, *connector-bases* may be defined to create libraries of xconnectors that can be reused to create links.

As an example of xconnector library, consider the thirteen well-known synchronization relations defined by Allen [1]. Although Allen specified a complete set of all possible relationships that can exist between two intervals, they do not precisely express the causal or constraint semantics that should exist between the time intervals [4]. For instance, the *meets* relation only specifies that the end of interval *x* should coincide with the beginning of *y*, but several interpretations could be given. The *meets* relation could be considered either a simple constraint, or a starting causality (the end of *x* must cause the start of *y*) or even a stopping causality (the beginning of *y* must cause the stop of *x*). Therefore, a hypermedia

authoring language should provide means for specifying both causality and constraint relations if it wants to provide all possible interpretations of Allen's relations. A connector-base for Allen's relations unambiguously specifying the relation semantics desired is available at http://www.telemidia.puc-rio.br/specs/xml/allen_base.xml. Similarly, a connector-base providing spatial synchronization relations can also be defined.

4. RELATED WORK

Since hypermedia links may represent several types of relation, a link needs to characterize the relation type and specify the set of related components. Although the definition is conceptually divided in two parts, hypermedia languages/models usually have a unique entity to capture both parts, the link itself.

The relation type captured by a link depends on the expressiveness of the authoring language used. In HTML[4] and Microcosm [10], links are uni-directional and single-headed representing the traditional hyperlink navigation behavior. SMIL [16] links are also uni-directional and single-headed, but they can be either triggered by user interaction or other triggering conditions, such as temporal events. In Dexter [7] and AHM [6], links may be bi-directional and multi-headed, but are used only to capture hypermedia relations that depend on user interaction. Dexter has no support for temporal relationships and AHM provides point-to-point synchronization arcs to capture temporal constraints between two parts of a presentation. Labyrinth [3] and NCM [17, 18] provide the definition of complex multipoint relationships among components based on different concepts of event. In NCM, a link is defined as a causal or constraint relationship among events that happen over document nodes, analogous to the ideas used in XConnector. In fact, many of XConnector features came from NCM links. In Labyrinth, an event is treated as a separate model entity that can be attached to a link in order to describe causal relationships. A Labyrinth event defines a condition and a list of actions, similar to a causal xconnector behavior.

Hypermedia languages/models also differ in the way the link entity is treated. In HTML, for example, links are embedded in node content, preventing reuse of the same resource without inheriting previously defined links. Other models allow the definition of a link as an independent entity, such as Dexter, AHM, Microcosm, XLink, Labyrinth and NCM. In Dexter and AHM, links are considered a type of document component and, different from most models, links can even be used to link other links [6]. In NCM, Microcosm and XLink, links can only connect nodes. In NCM, links are contained in composite nodes and must relate nodes recursively contained in the composite node. NCM composite nodes can be reused, but links alone cannot. Microcosm and XLink provide independent link repositories that can be reused in several documents.

Although most hypermedia languages/models provide a unique entity to capture relationship definitions, some of them already divide the definition in two parts, the first specifying the relation and the second, the related participants. In XLink, for example, the definition of participating resources and the definition of traversal rules are done with different child elements of an extended link. In NCM, a link is defined by sets of source and target endpoints and another attribute called meeting point, which specifies the composite traversal behavior. However, in both cases, traversal behavior is embedded in the link definition and cannot be handled independently.

Other hypermedia models provide predefined sets of relation types that can be used for creating relationships in a document. These models actually break the definition of a relationship in two parts and provide relation types separately from relation instances. Madeus [9] high-level constraints and the typed transitions of Petri-net based models like I-HTSPN [19] and caT [12] are examples. However, these models do not provide the definition of user-defined relation types.

Addressing relation type definition, reference [11] proposed another first-class entity, called hypermedia connector, whose main purpose is describing the relation independent of which participants are related. XConnector uses and extends this idea and proposes an authoring language for defining connectors to represent referential and synchronization relations. XConnector alone does not provide the definition of links, as it only allows specifying the relation type. Another language must be used to complete the link definition, specifying its set of participants. That language can be XLink, following the proposal presented in this paper, or any other hypermedia authoring language willing to take profit of XConnector facilities.

Note that features like automatically computing link sources and destinations, offered by Microcosm generic links and the virtual links of NCM and Labyrinth, are concerns of the linking language and not of XConnector. Another issue is the definition of the link context [5]. XConnector does not have the duty to define link specifiers, identifying component anchors, and consequently link specifier contexts, as proposed by AHM [6]. This is responsibility of the linking language using XConnector.

5. CONCLUSIONS

This paper presented XConnector, a language that can be used to extend existing linking languages in order to provide multimedia synchronization facilities for WWW resources using links. The paper defined how XLink can be extended to incorporate XConnector facilities.

The use of connectors also provides the possibility of defining relations as composite elements, augmenting even more the flexibility for expressing and reusing relation specifications. Composite connectors represent groups of several connectors and components, modeling more elaborated relationships among components of a document [11]. Although composite connectors could be provided, they were not discussed in this paper, since their definition requires the definition of document components. The main goal of this paper was to present a hypermedia connector authoring language that can be used together with a hypermedia linking language, regardless of how the latter defines its document components.

Connectors could be used in a broader sense to represent any kind of relation, besides the synchronization relation types focused in this work. For example, we could have a version connector to represent the ancestor/descendant relation of versioned resources, or we could have a channel connector to represent the publish-subscribe paradigm used in notification mechanisms. A future work is to extend XConnector and create new connector-bases to represent other kinds of relations found in the hypermedia domain.

[4] HTML script programming was not considered for comparison.

The specification of synchronization relations separate from participating resources is indeed a form of defining link templates, which must be understood by document formatters. As a future work, this proposal will be generalized for document structures, making the definition of composition templates possible. Composition templates will define relationships among components of a composition. For example, the SMIL *par* and *seq* compositions could be seen as templates that would be interpreted by SMIL players, as well as other user-defined kinds of compositions. Link and composition templates could then be treated as high-level language elements by document authors.

As a validation for the proposed ideas, XConnector was incorporated into the HyperProp hypermedia system [18] implementation. Using JAXP (Java API for XML Processing), an XML parser for XConnector was implemented, allowing users to import connector-bases to the system and use hypermedia connectors to create causal and constraint NCM links. The HyperProp formatter was also adapted to understand and play documents with links using causal and constraint hypermedia connectors.

6. REFERENCES

[1] Allen J.F. "Maintaining Knowledge about Temporal Intervals", Communications of the ACM, 26(11), Nov. 1983.

[2] Antonacci M.J., Muchaluat-Saade D.C., Rodrigues R.F., Soares L.F.G. "Improving the expressiveness of XML-based Hypermedia Authoring Languages", *Multimedia Modeling Conference*, Nagano, Japan, November 2000.

[3] Díaz P., Aedo I., Panetsos F. "Modeling the Dynamic Behavior of Hypermedia Applications", *IEEE Transactions on Software Engineering*, 27(6), June 2001.

[4] Duda A., Keramane C. "Structured Temporal Composition of Multimedia Data", *IEEE International Workshop on Multimedia Database Management Systems*, NY, Aug. 1995.

[5] Hardman L., Bulterman D.C.A., van Rossum G. "Links in hypermedia: the requirement for context", *ACM Conference on Hypertext*, Seattle, Washington, November 1993.

[6] Hardman L. "Modelling and Authoring Hypermedia Documents", *PhD Thesis*, University of Amsterdam, 1998. Available at http://www.cwi.nl/~lynda/thesis

[7] Halasz F., Schwartz M. "The Dexter Hypertext Reference Model", *Communications of the ACM*, 37(2), February 1994.

[8] Hardman L., Schmitz P., Ossenbruggen J., Kate W., Rutledge L. "The Link vs. the Event: Activating and Deactivating Elements in Time-Based Hypermedia", *New Review of Multimedia and Hypermedia*, Taylor Graham, Vol. 6, 2000.

[9] Jourdan M., Layaïda N., Roisin C., Sabry-Ismail L., Tardif L. "Madeus, an Authoring Environment for Interactive Multimedia Documents", *ACM Multimedia Conference 98*, England, September 1998.

[10] Lowe D., Hall W. "Hypermedia & The Web: an Engineering Approach", John Willey & Sons, 1999.

[11] Muchaluat-Saade D.C., Soares L.F.G. "Hypermedia Spatio-Temporal Synchronization Relations also Deserve First-Class Status", *Multimedia Modeling Conference*, Amsterdam, November 2001.

[12] Na J., Furuta R. "Dynamic Documents: Authoring, Browsing and Analysis Using a High-Level Petri Net-Based Hypermedia System", *ACM Symposium on Document Engineering - DocEng'01*, Atlanta, USA, November 2001.

[13] Pérez-Luque M.J., Little T.D.C. "A Temporal Reference Framework for Multimedia Synchronization", *IEEE Journal on Selected Areas in Communications*, 14(1), January 1996.

[14] Rodrigues L.M., Antonacci M.J., Rodrigues R.F., Muchaluat-Saade D.C., Soares L.F.G. "Improving SMIL with NCM Facilities", *Journal of Multimedia Tools and Applications*, 16(1), Kluwer Academics Publisher, January 2002.

[15] "XML Schema Part 0: Primer", *W3C Recommendation*, May 2001. Available at http://www.w3.org/TR/xmlschema-0/

[16] "Synchronized Multimedia Integration Language (SMIL 2.0)", *W3C Recommendation*, August 2001. Available at http://www.w3.org/TR/smil20

[17] Soares L.F.G., Casanova M., Rodriguez N. "Nested Composite Nodes and Version Control in an Open Hypermedia System", *International Journal on Information Systems*, 20(6), Elsevier Science, England, September 1995.

[18] Soares L.F.G., Rodrigues R.F., Muchaluat-Saade D.C. "Modeling, Authoring and Formatting Hypermedia Documents in the HyperProp System", *ACM Multimedia Systems Journal*, 8(2), March 2000.

[19] Willrich R., Saqui-Sannes P., Senac P., Diaz M. "Hypermedia Document Design Using the HTSPN Model", *Multimedia Modeling Conference*, Toulouse, France, November 1996.

[20] "XHTML+SMIL Profile", *W3C Note*, January 2002. Available at http://www.w3.org/TR/XHTMLplusSMIL

[21] "XHTML 1.1 – Module-based XHTML", *W3C Recommendation*, May 2001. Available at http://www.w3.org/TR/xhtml11

[22] "XML Linking Language (XLink) Version 1.0", *W3C Recommendation*, June 2001. Available at http://www.w3.org/TR/xlink