4
Módulo XConnector

Como apresentado no Capítulo 2, um conector hipermídia (Muchaluat-Saade, 2001a; Muchaluat-Saade, 2001b) representa uma relação que pode ser usada para a criação de elos em hiperdocumentos. Um conector especifica uma relação de forma independente do relacionamento, isto é, não especifica quais são os nós que irão interagir através da relação. Elos que representam um mesmo tipo de relação, mas que interligam nós distintos, podem reusar um mesmo conector reaproveitando toda a especificação já feita. Um conector hipermídia especifica um conjunto de pontos de interface, chamados papéis (*roles*). Para a criação de um elo, faz-se referência a um conector e define-se um conjunto de *binds*, que associam cada extremidade do elo (ponto de interface de um nó) a um papel do conector utilizado.

A Figura 11 ilustra um conector *R* representando uma relação com três papéis distintos, que significam três tipos de participantes da relação. A figura também mostra dois elos diferentes, l_1 e l_2, reusando *R*. Enquanto o conector define o tipo de relação, o conjunto de *binds* de um elo define os participantes. O elo l_1 especifica três *binds*, interligando os nós *A*, *B* e *C* aos papéis de *R* e o elo l_2 também, só que interligando um conjunto diferente de nós (*B, C* e *D*). Os elos l_1 e l_2 definem relacionamentos diferentes, já que interligam conjuntos distintos de nós, mas representam o mesmo tipo de relação, pois usam o mesmo conector. Na especificação de um documento, um elo faz referência a uma instância de conector.

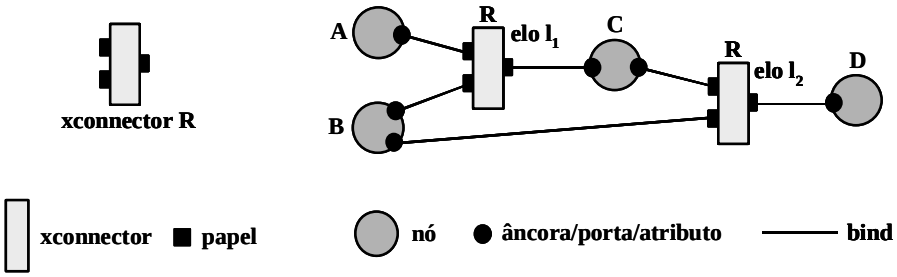

Figura 11. Exemplo de elos usando o mesmo conector R

Conceitualmente, conectores podem representar qualquer tipo de relação hipermídia, tal como relações de referência, relações de sincronização, relações semânticas, relações de derivação etc. Como as relações de sincronização espacial e temporal são, sem dúvida, as mais complexas de serem especificadas, este trabalho concentrou seus esforços na especificação desse tipo de relação e também das relações de referência, oferecendo o suporte necessário para a criação de documentos hipermídia.

XConnector (Muchaluat-Saade, 2002a) é uma linguagem XML que permite a definição de conectores que podem ser usados para especificar relações de referência e de sincronização espaço-temporal hipermídia, tratando relações de referência como casos particulares de relações de sincronização.

Apesar de XConnector ter sido incorporado como módulo da versão 2.0 da linguagem NCL, XConnector é totalmente independente dos outros módulos de NCL e pode ser utilizado em conjunto com outras linguagens. Mais adiante, ainda neste capítulo, será apresentada uma proposta de extensão da linguagem XLink (W3C, 2001c), incorporando as facilidades de XConnector.

A linguagem XConnector permite a definição de relações multiponto com semântica causal ou de restrição. Em uma relação causal, uma condição deve ser satisfeita para que uma ação seja executada. Um exemplo de relação causal é a tradicional relação de referência hipermídia, que causa a navegação para um nó de destino quando uma âncora de um nó de origem for selecionada pelo usuário. Um outro exemplo de relação causal pode iniciar a apresentação de um nó, quando a apresentação de outro terminar. Além de relações causais, relações de restrição, sem nenhuma causalidade envolvida, também podem ser especificadas por XConnector. Considere, por exemplo, uma restrição especificando que um nó deve terminar sua apresentação ao mesmo tempo que outro começa a dele. A ocorrência de uma apresentação sem a ocorrência da outra também satisfaz a restrição, que especifica que, se e somente se esses dois nós forem apresentados, seus tempos de fim e início devem coincidir.

Para capturar relações causais e de restrição, conectores são especializados em conectores causais e conectores de restrição. Em ambos os tipos, a definição de um conector (elemento *xconnector*) é feita por um conjunto de papéis (*roles*), que determinam a função dos participantes da relação, e um atributo *glue*, que

descreve como os papéis interagem. A definição de papéis é baseada no conceito de evento. Cada papel descreve um evento associado a um participante da relação e o glue descreve a combinação entre os eventos de acordo com a semântica de causalidade ou de restrição.

Como afirmado em (Pérez-Luque, 1996), um evento é uma ocorrência no tempo, que pode ser instantânea ou pode ocorrer durante um período. O tipos básicos de eventos em XConnector são *presentation* (apresentação), *mouseClick* (clique do mouse), *mouseOver* (posicionamento do mouse), *focus* (foco na interface do usuário), *prefetch* (pré-busca) e *attribution* (atribuição). Esse conjunto pode ser estendido para incluir outros tipos relevantes de evento.

O comportamento de cada evento é controlado por sua máquina de estados, exibida na Figura 12. Estados e transições da máquina de estados de um evento podem ser usados na especificação de papéis de um conector. Para os tipos de evento *presentation*, *prefetch* e *attribution*, é bastante útil permitir a ação de pausa e recomeço (*resume*) em sua ocorrência. Por outro lado, pausar e recomeçar eventos do tipo *mouseClick*, *mouseOver* e *focus* não faz muito sentido. Nesse caso, uma máquina de estados mais simples é considerada, apenas com os estados *prepared* e *occurring* e as transições entre eles. A máquina de estados de eventos do tipo *prefetch* é diferente das outras, pois quando esse evento termina naturalmente, ele vai para o estado *finished*. Na verdade, o evento *prefetch* deve acontecer antes da ocorrência de qualquer outro evento sobre um mesmo participante. Assim sendo, quando um evento *prefetch* chega ao estado *finished*, ele causa a criação das outras máquinas de estado que controlam eventos referentes ao mesmo objeto, iniciando-as no estado *prepared*.

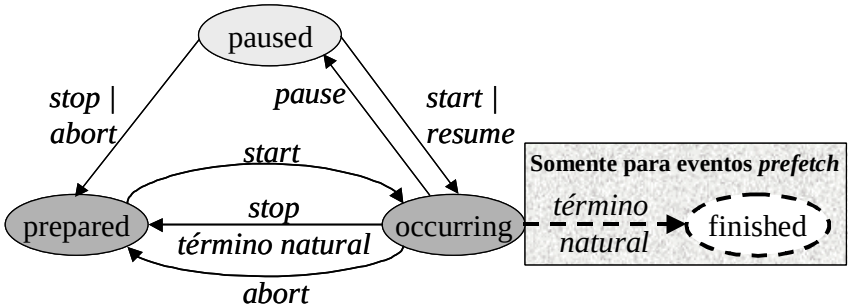

Figura 12. Máquina de estados de um evento

Eventos em XConnector, exceto os do tipo *prefetch*, têm um atributo associado chamado *occurrences*, que conta quantas vezes seu estado muda de

occurring para *prepared*. Eventos do tipo *presentation* e *attribution* também têm outro atributo chamado *repeat*, que indica quantas vezes eles devem ainda ocorrer.

A sintaxe de XConnector usa várias construções de outros padrões existentes, tais como XHTML e SMIL, e introduz algumas novas (Muchaluat-Saade, 2002a). Contudo, o conceito de evento usado na definição de papéis é o definido por (Pérez-Luque, 1996), como mencionado anteriormente, podendo ter uma duração. Com isso, diferentes tipos de evento definidos por SMIL e XHTML, que são sempre instantâneos, são tratados como transições de um mesmo evento por XConnector. Por exemplo, os eventos *beginEvent* e *endEvent* de SMIL correspondem às transições *starts* e *stops* na máquina de estados de um evento de apresentação em XConnector. Os nomes das transições de estados usadas na autoria de um conector estão listados na Tabela 11. A definição completa da linguagem XConnector usando XML Schema (W3C, 2001b) está disponível no Apêndice A.

Tabela 11. Nomes das transições para a máquina de estados de um evento

Transição (causada pela ação)	**Nome da Transição**
prepared→occurring (start)	*starts*
occurring→prepared (stop ou término natural)	*stops*
occurring→prepared (abort)	*aborts*
occurring→finished (término natural)	*ends*[9]
occurring→paused (pause)	*pauses*
paused→occurring (resume ou start)	*resumes*
paused→prepared (stop ou abort)	*abortsFromPaused*

Cada papel (elemento *role*) de um conector define um identificador único (*id*) no conjunto de papéis do conector, um tipo de evento (*eventType*) e sua cardinalidade. A cardinalidade de um papel indica o número mínimo (*min*) e máximo (*max*) de participantes que podem desempenhar o papel (número de *binds*), quando esse conector for usado na criação de um elo, como será definido posteriormente. Se o tipo de evento do papel for *attribution*, ele deve ainda definir o nome do atributo correspondente.

Papéis são especializados em condição, ação e propriedade. Diferentes tipos de papéis são usados de acordo com o tipo de conector. Em conectores de

[9] Válida somente para eventos do tipo *prefetch*.

restrição, somente papéis do tipo propriedade devem ser usados. Em conectores causais, qualquer tipo de papel pode ser usado.

Papéis do tipo ação (*actionRole*) capturam ações que devem ser executadas em uma relação causal. Os tipos de ação estão ilustrados na Figura 12 por arcos rotulados, que causam transições na máquina de estados (exceto "término natural"). Além do tipo da ação, um papel do tipo ação pode definir um retardo (*delay*) a ser introduzido antes da ação ser executada; valores a serem atribuídos no caso de eventos de atribuição; e um valor a ser atribuído ao atributo *repeat* do evento, no caso de apresentação ou atribuição. Todos esses valores podem ser parametrizados, permitindo que um mesmo conector possa ser reusado passando diferentes valores para seus parâmetros. Um exemplo de papel do tipo ação é "pause a apresentação depois de um retardo de *n* segundos", onde *n* é definido como parâmetro do conector. Papéis do tipo ação ainda podem definir o atributo *show*, que tem o mesmo significado do atributo homônimo em XLink (W3C, 2001c), especificando o comportamento dos nós de origem quando um elo for seguido. Os possíveis valores desse atributo são *new*, *embed* e *replace*, especificando respectivamente que o participante de destino da relação deve ser apresentado sem afetar a execução do participante de origem, ou então deve ter sua apresentação embutida na apresentação do participante de origem ou ainda deve substituir a apresentação desse participante. Em relações causais multiponto, o comportamento de participantes de origem da relação quase sempre pode ser modelado por outra ação da relação, exceto no caso do valor *embed*, o que justifica o uso do atributo *show* em XConnector.

Em conectores causais, condições devem ser satisfeitas para disparar as ações. As condições são capturadas por papéis do tipo condição (*conditionRole*), que definem expressões lógicas avaliando estados e transições de eventos ou valores de atributos. Quando uma condição é avaliada, ela retorna um valor booleano.

As condições podem ser simples ou compostas. Uma condição simples (*simpleCondition*) pode ser uma comparação usando os operadores *eq* (=), *dif* (≠), *lt* (<), *lte* (≤), *gt* (>) ou *gte* (≥), podendo testar uma transição de estados do evento, um valor de estado do evento, um valor de atributo do evento tal como *repeat* e *occurrences*, como explicado anteriormente, ou ainda um valor de atributo de um

participante. Quando a expressão é avaliada sobre uma transição de estados do evento, a condição é considerada verdadeira somente no momento em que a transição ocorre.

Qualquer condição pode ser negada e uma condição composta (*compoundCondition*) consiste de uma expressão lógica binária, baseada nos operadores *and* ou *or*, envolvendo duas outras condições sobre o mesmo evento. Um exemplo de papel com condição composta é "a apresentação de um participante terminou pela segunda vez", que seria especificada como "[(*eventType* = "presentation"), ((*transition* = "stops") AND (*occurrences* = "2"))]".

O terceiro e último tipo de papel é a propriedade (*propertyRole*). Uma propriedade pode ser um valor de estado do evento, o instante de tempo em que uma transição de estados do evento ocorre, um valor de atributo do evento ou um valor de atributo de um participante. Enquanto uma condição sempre retorna um valor booleano, uma propriedade retorna qualquer tipo de valor, dependendo do tipo da propriedade. Propriedades referentes a transições de eventos e valores de atributos podem especificar um valor de deslocamento (*offset*) que pode ser adicionado ao resultado da propriedade. Por exemplo, uma propriedade pode especificar "5 segundos após o instante de tempo em que um evento de apresentação termina" ou "a posição vertical na tela mais 50 pixels". O valor do atributo *offset* também pode ser parametrizado pelo conector.

Como mencionado anteriormente, um conector é definido por um conjunto de papéis e um *glue*, que especifica como os papéis interagem. Todo papel de um conector deve ser usado em seu *glue*. Um conector de restrição tem um *glue* de restrição, que define uma expressão relacionando papéis do tipo propriedade. Um conector causal tem um *glue* causal, que define tanto uma expressão de disparo (*triggerExpression*), relacionando papéis do tipo condição ou propriedade, quanto uma expressão de ações (*actionExpression*), relacionando papéis do tipo ação. Quando a expressão de disparo for satisfeita, a expressão de ações deve ser executada.

A expressão de propriedades do *glue* de restrição pode ser simples ou composta. Uma expressão de propriedades simples (*simplePropertyExpression*) pode comparar papéis de propriedades do mesmo tipo, ou um papel de

propriedade com um valor do mesmo tipo do resultado da propriedade, onde esse valor pode ser parametrizado pelo conector. A comparação pode usar os operadores *eq* (=), *dif* (≠), *lt* (<), *lte* (≤), *gt* (>) ou *gte* (≥). Por exemplo, suponha um papel de propriedade identificado por *P*, especificando a transição que inicia um evento de apresentação, e um outro papel de propriedade identificado por *Q*, especificando a transição que termina um evento de apresentação. Se uma expressão de propriedades simples S_1 definir que "$P = Q$", S_1 será avaliada como verdadeira se o primeiro evento de apresentação iniciar ao mesmo tempo em que o outro evento de apresentação terminar. Como um outro exemplo, suponha que um papel de propriedade *H* especifica o valor do atributo posição horizontal na tela. Se uma expressão de propriedades simples S_2 definir que "$H \geq 100$", S_2 será avaliada como verdadeira se a posição horizontal de um participante desempenhando o papel *H* for maior que "100". Qualquer expressão de propriedades pode ser negada e uma expressão de propriedades composta (*compoundPropertyExpression*) consiste de uma expressão lógica binária, baseada nos operadores *and* ou *or*, envolvendo duas outras expressões de propriedades. Apesar de expressões de propriedades poderem ser utilizadas em conectores causais, sua principal utilidade está na especificação de conectores de restrição, como no exemplo a seguir.

A Tabela 12 ilustra um exemplo de conector de restrição expressando uma relação de sincronização espacial especificando que "dois nós devem ser alinhados pelo topo (seus atributos *top* devem ser idênticos)".

Tabela 12. Exemplo de conector de restrição

Tipo do papel e id	**Tipo do evento**	**Cardinalidade (min,max)**	**Nome do atributo**	***offset***
Propriedade P_1	*attribution*	*(1,1)*	*top*	*0*
Propriedade P_2	*attribution*	*(1,1)*	*top*	*0*

Tipo do glue	**Expressão de propriedades**
Restrição	$P_1 = P_2$

Uma expressão de disparo de um *glue* causal também pode ser simples ou composta. Uma expressão de disparo simples (*simpleTriggerExpression*) se refere ao *id* de um papel do tipo condição. Uma expressão de disparo composta (*compoundTriggerExpression*) consiste de uma expressão lógica binária, baseada nos operadores *and* ou *or*, envolvendo duas outras expressões de disparo, ou uma expressão de propriedade e uma expressão de disparo. Qualquer expressão de

disparo pode ser negada e pode especificar retardos mínimo (*minDelay*) e máximo (*maxDelay*) para sua avaliação. Por exemplo, dado que uma expressão de disparo *C* é verdadeira no instante *t*, *C'* definida com *minDelay*="t_1" e *maxDelay*="t_2" é verdadeira no intervalo [$t+t_1$, $t+t_2$]. Os valores de retardo mínimo e máximo podem ser parametrizados pelo conector.

Expressões de disparo compostas podem relacionar qualquer número de papéis de condição e de propriedade. Entretanto, uma restrição é necessária para garantir a consistência de relações causais. Toda expressão de disparo deve ser satisfeita somente em um instante infinitesimal, exigindo que pelo menos um papel de condição de cada conector causal defina uma condição sobre uma transição de estados de um evento.

Uma expressão de ações também pode ser simples ou composta. Uma expressão de ações simples (*simpleActionExpression*) se refere ao *id* de um papel do tipo ação. Uma expressão de ações composta (*compoundActionExpression*) consiste de uma expressão lógica binária, baseada nos operadores *par*, *seq* ou *excl*, envolvendo duas outras expressões de ações. Expressões compostas de ações parelelas (*par*) ou seqüenciais (*seq*) especificam que a execução das ações deve ser feita em qualquer ordem ou em uma ordem específica. Expressões compostas de ações exclusivas (*excl*) especificam que somente uma das ações deve ser executada[10].

Quando a cardinalidade máxima de um papel for maior que um (*max > 1*), vários participantes podem desempenhar o mesmo papel. Nesse caso, um qualificador (*qualifier*) deve ser especificado toda vez que esse papel for usado em expressões do *glue*. A Tabela 13 apresenta os possíveis valores para qualificadores.

Tabela 13. Valores para qualificadores de papéis

Tipo do papel	Qualificador	Semântica
Condição	*all*	Todas as condições devem ser verdadeiras
Condição	*any*	Pelo menos uma condição deve ser verdadeira
propriedade	*all*	Todas as propriedades devem ser consideradas
propriedade	*any*	Pelo menos uma propriedade deve ser considerada
ação	*par*	Todas as ações devem ser executadas em paralelo
ação	*excl*	Somente uma das ações deve ser executada

[10] Quando somente uma das ações deve ser executada, o formatador do documento deve decidir qual delas será, ou pode deixar a decisão a cargo do usuário.

A Tabela 14 ilustra um exemplo de conector causal expressando uma relação de sincronização temporal. A especificação do conector pode ser interpretada como "se um grupo de participantes estiver sendo apresentado (C_1) e outro participante for selecionado (C_2), pare a apresentação do grupo de participantes (A_1) e inicie a apresentação de outro participante (A_2)". Para parar a apresentação do mesmo grupo de participantes que desempenhou o papel C_1, um elo usando esse conector deve criar dois *binds* para cada participante do grupo, um para o papel C_1 e outro para o papel A_1.

Tabela 14. Exemplo de conector causal

Tipo do papel e id	**Tipo do evento**	**Cardinalidade (min,max)**	**Condição**	**Ação**
Condição C_1	*presentation*	*(1, unbounded)*	*state=occurring*	
Condição C_2	*mouseClick*	*(1, 1)*	*transition=stops*	
Ação A_1	*presentation*	*(1, unbounded)*		*stop*
Ação A_2	*presentation*	*(1, 1)*		*start*

Tipo do glue	**Expressão de disparo**	**Expressão de ações**
Causal	*all(C_1) AND C_2*	*seq(par(A_1), A_2)*

Como mencionado anteriormente, alguns valores de atributos de papéis e do *glue* de um conector podem ser parametrizados. A definição de parâmetros em um conector pode ser feita declarando elementos-filho chamados *param*, que possuem os atributos nome (*name*) e tipo (*type*) do parâmetro. Parâmetros podem ser definidos tanto para conectores causais como para conectores de restrição. Para especificar quais atributos recebem os valores dos parâmetros definidos pelo conector, ao realizar a definição desses atributos, deve-se especificar seus valores como sendo o nome do parâmetro precedido do símbolo *$*. Por exemplo, para parametrizar o atributo retardo (*delay*) de um papel do tipo ação, define-se um parâmetro chamado *actionDelay* (*<param name="actionDelay" type="xsd:unsignedLong"/>*) e usa-se o valor *"$actionDelay"* no atributo correspondente (*delay*="$actionDelay").

Como a definição de conectores não é simples de ser feita por um usuário leigo, pois ele precisaria conhecer os conceitos de estados e transições de eventos, a idéia é fazer com que usuários experientes definam conectores, os armazenem em bibliotecas, chamadas de bases de conectores (*connectorBases*), e as tornem disponíveis para a criação de elos. Como exemplo de base de conectores, considere um conjunto contendo as treze famosas relações de sincronização

temporal propostas por Allen (Allen, 1983). Apesar de Allen ter especificado um conjunto completo de todas as relações possíveis entre dois intervalos temporais, elas não expressam, precisamente, a semântica causal ou de restrição que deveria existir entre os intervalos (Duda, 1995). Por exemplo, a relação chamada *meet* especifica que o fim de um intervalo *x* deve coincidir com o início do intervalo *y*, o que dá margem a várias interpretações diferentes. A relação *meet* poderia ser considerada como uma simples restrição, ou então uma causalidade onde o término de *x* provoca o início de *y*, ou ainda uma causalidade onde o início de *y* provoca o término de *x*. Ao especificar relações com XConnector, o autor pode escolher e definir a semântica exata que deseja expressar através da relação. O Apêndice B apresenta uma base de conectores especificada com XConnector definindo as relações temporais de Allen sem ambigüidades.

Uma das principais características de XConnector é poder ser usado para a definição de elos em linguagens de autoria hipermídia, independente de como elas definem os nós componentes do documento. O capítulo anterior já comentou o uso de XConnector como módulo da linguagem NCL 2.0. A próxima seção ilustra como o padrão XLink do W3C pode ser estendido para incorporar as facilidades de XConnector.

4.1 Extensões ao padrão XLink

XLink (W3C, 2001c) é um padrão do W3C para a criação de elos entre recursos XML. Ao contrário de outras linguagens, XLink não especifica nomes de elementos XML para a criação de elos, mas sim atributos e valores para esses atributos, permitindo que uma linguagem defina elementos, com o nome que desejar, e utilize os atributos XLink para obter a semântica definida no padrão.

XLink fornece dois tipos de elos, elos simples (elementos do tipo *simple*), que são similares aos hiper-elos unidirecionais do XHTML, e elos estendidos (elementos do tipo *extended*), que oferecem todas as funcionalidades XLink. Elos estendidos permitem descrever relações sofisticadas com um número arbitrário de recursos participantes. Um elo estendido especifica um conjunto de participantes e um conjunto de regras de navegação, como é mostrado na Figura 13. O elemento

courseload ilustrado na figura é um elo estendido XLink (atributo *type=”extended”*).

```
<courseload xlink:type="extended”>

  <person xlink:type="locator” xlink:href="students/patjones62.xml"
          xlink:label="student"/>
  <person xlink:type="locator” xlink:href="students/peterkorb60.xml"
          xlink:label="student"/>
  <person xlink:type="locator” xlink:href="professors/jaysmith7.xml"
          xlink:label="professor"/>

  <course xlink:type="locator” xlink:href="courses/cs101.xml"
          xlink:label="course"/>

  <go xlink:type="arc” xlink:from="student" xlink:to="course"
      xlink:show="replace" xlink:actuate="onRequest"/>
  <go xlink:type="arc” xlink:from="professor” xlink:to="course"
      xlink:show="replace" xlink:actuate="onRequest"/>
  <go xlink:type="arc” xlink:from="course" xlink:to="professor"
      xlink:show="replace" xlink:actuate="onRequest”/>

</courseload>
```

Figura 13. Exemplo de elo estendido XLink

Participantes podem ser recursos locais (elementos do tipo *resource*) ou recursos remotos (elementos do tipo *locator*), considerando o recurso onde o elo é definido. Cada participante de um elo estendido tem um rótulo (*label*) que é usado para especificar as regras de navegação. O exemplo da Figura 13 define quatro participantes remotos, três deles são elementos *person* e um é o elemento *course*.

Uma regra de navegação XLink (elemento do tipo *arc*) relaciona um rótulo de origem (atributo *from*) a um rótulo de destino (atributo *to*), além de especificar quando a navegação deve ocorrer (atributo *actuate*) e como a apresentação do nó de destino deve ser realizada (atributo *show*). Como vários participantes podem ter o mesmo rótulo, é possível definir relacionamentos multiponto. Assim sendo, as regras de navegação determinam que participantes são origem ou destino de um elo. A navegação pode ser ativada a pedido do usuário (*actuate* = “onRequest”) ou pode ser feita automaticamente quando um recurso de origem é carregado (*actuate*=“onLoad”). A apresentação do recurso de destino pode ser feita em uma nova janela (*show*=“new”), pode substituir o recurso de origem (*show*=“replace”) ou pode até ser embutida em sua apresentação (*show*=“embed”). Na verdade, um elo estendido XLink não representa um relacionamento único, pois cada regra de navegação é um relacionamento independente. Desta forma, um elo estendido pode ser visto como uma composição hipermídia que contém componentes (participantes XLink) e um

conjunto de elos interligando-os (regras de navegação). O elo estendido XLink ilustrado na Figura 13 define três regras de navegação representadas por elementos *go*.

XLink também fornece atributos semânticos (*role*, *arcrole* e *title*) para descrever o significado dos recursos no contexto de um elo, entretanto, eles estão fora do escopo deste trabalho.

XLink permite a definição de elos que residem em um local separado dos recursos relacionados. Isto é útil quando recursos são leitura-somente ou quando eles não oferecem meios para embutir construções de elos, como por exemplo arquivos cujo conteúdo é vídeo. Além disso, XLink fornece a definição de repositórios de elos, chamados *linkbases*, facilitando a gerência dos elos.

Apesar do fato de XLink oferecer elos mais sofisticados do que os hiper-elos simples do XHTML, a linguagem também apresenta limitações:

- XLink só fornece a especificação de relações causais, ou seja, uma condição deve ser satisfeita para que uma ação seja executada. Não existe suporte para a definição de relações de restrição entre recursos, como discutido no início deste capítulo.
- A navegação no XLink pode ser ativada a pedido do usuário ou automaticamente quando o recurso de origem for carregado. Não existe suporte para outras condições, tais como “quando a apresentação do conteúdo de um recurso terminar”.
- Apesar de podermos criar elos multiponto usando XLink, cada regra de navegação só relaciona um único tipo de condição para todos os participantes, determinado pelo valor do atributo *actuate*, e um único tipo de ação para todos os participantes, que é “inicie a apresentação do recurso de destino considerando o valor do atributo *show*”. Não há como definir uma relação multiponto que especifique condições e ações compostas. Como exemplo, considere uma relação que expressa “se um filme está sendo apresentado e um ícone de pedido de pizza for selecionado, pause a apresentação do filme e inicie a apresentação de um formulário de compra de pizza”.
- XLink não permite a especificação de relações espaciais, tal como uma restrição que define que “dois objetos devem estar alinhados

verticalmente” ou uma causalidade espaço-temporal que pode ser usada para especificar, em um jogo educativo, que “se um objeto for movido para uma posição específica, uma mensagem de áudio é apresentada” parabenizando a criança por ter completado a operação corretamente. Apesar da inclusão de relações espaciais dentro de um documento parecer estar ferindo os conceitos básicos de separação entre conteúdo e apresentação, ela pode ser necessária para especificar o posicionamento relativo entre objetos. Para garantir que um relacionamento espacial entre componentes de uma apresentação multimídia seja válido durante a execução da apresentação, ele precisa ser criado entre os objetos e não na especificação de layout.

Além disso, XLink não permite a definição de tipos de relação independente da definição de seus participantes. Isto impede o reuso da definição da relação em elos estendidos distintos com o mesmo comportamento de navegação e, como conseqüência, também introduz outra limitação:

- XLink permite a definição de repositórios de elos mas não oferece a definição de repositórios de tipos de regras de navegação. Este recurso torna-se importante quando tipos complexos de regras de navegação podem ser definidos.

4.1.1 Estendendo o padrão XLink com XConnector

Para incorporar as facilidades de XConnector a XLink (Muchaluat-Saade, 2002a), a definição de uma regra de navegação (elemento do tipo *arc*) deve ser modificada. Seus atributos chamados *from*, *to*, *show* e *actuate* se tornam obsoletos e um novo atributo chamado *xconnector*, usado para identificar a URI de um conector válido, é introduzido. Além disso, elementos do tipo *arc* devem definir um conjunto de elementos filhos de um novo tipo chamado *bind*. Cada elemento-filho do tipo *bind* especifica qual recurso participante desempenha que papel específico do conector usado pelo elemento pai. Para isso, elementos do tipo *bind* devem ter um atributo *label* para identificar um rótulo de um participante XLink e um atributo *role* para identificar um dos papéis do conector. Note que a semântica desse atributo *role* é diferente do atributo XLink homônimo. Com essas

modificações, um único elemento do tipo *arc* pode relacionar qualquer número de rótulos de participantes de um elo estendido XLink, introduzindo maior flexibilidade na especificação do comportamento de navegação.

Apesar da extensão proposta manter os atributos XLink que especificam comportamento de navegação em um elemento do tipo *arc*, sugere-se que o comportamento de navegação seja agora especificado por um conector. A Figura 14 ilustra a definição de um conector causal com a mesma semântica que uma regra de navegação XLink com atributos *actuate*=”onRequest” e *show*=”replace”, identificado por “xlink-replace-onRequest”. Esse conector especifica que quando um evento de clique do mouse terminar sua ocorrência, um evento de apresentação deve ser iniciado.

```
<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>
```

Figura 14. Conector representando uma regra de navegação XLink com atributos *actuate*=”onRequest” e *show*=”replace”

Note que, como rótulos podem ser compartilhados por qualquer número de participantes em XLink, o atributo que representa a cardinalidade máxima (*max*) de cada papel do conector, especificado na Figura 14, foi definido com o valor ilimitado (“*unbounded”*). Apesar de XLink permitir o compartilhamento de rótulos, na verdade, uma regra de navegação, que relaciona mais de dois participantes, é tratada como vários elos ponto-a-ponto representando todas as combinações possíveis entre origens e destinos. Como o padrão XLink não restringe o comportamento da aplicação nesse caso e o autor de um conector pode escolher o comportamento desejado em XConnector, o qualificador do papel ação identificado como *“to”* foi definido como *“excl”* na expressão de ações do *glue* ilustrado na Figura 14.

Usando o mesmo exemplo dado na seção anterior (veja Figura 13), a Figura 15 mostra como fica sua especificação usando a extensão proposta. Note que as três regras de navegação usadas na Figura 13 agora foram transformadas em duas, que fazem referência ao mesmo conector "xlink-replace-onRequest" definido na Figura 14. Isso é possível, pois a extensão proposta permite associar mais de um rótulo a um mesmo papel, já que a cardinalidade máxima do papel é ilimitada na definição do conector usado.

```
<courseload xlink:type="extended">

   <person xlink:type="locator" xlink:href="students/patjones62.xml"
           xlink:label="student"/>
   <person xlink:type="locator" xlink:href="students/peterkorb60.xml"
           xlink:label="student"/>
   <person xlink:type="locator" xlink:href="professors/jaysmith7.xml"
           xlink:label="professor"/>

   <course xlink:type="locator" xlink:href="courses/cs101.xml"
           xlink:label="course"/>

   <go xlink:type="arc" xconnector:xconnector="xlink-replace-onRequest">
      <bind xlink:type="bind" xconnector:label="student"
            xconnector:role="from"/>
      <bind xlink:type="bind" xconnector:label="professor"
            xconnector:role="from"/>
      <bind xlink:type="bind" xconnector:label="course"
            xconnector:role="to"/>
   </go>

   <go xlink:type="arc" xconnector:xconnector="xlink-replace-onRequest">
      <bind xlink:type="bind" xconnector:label="course"
            xconnector:role="from"/>
      <bind xlink:type="bind" xconnector:label="professor"
            xconnector:role="to"/>
   </go>

</courseload>
```

Figura 15. Exemplo de elo usando a extensão XLink/XConnector

Comparando o exemplo da Figura 13 com sua nova definição mostrada na Figura 14 e na Figura 15, pode-se achar que a complexidade para usar XConnector é muito maior que a de XLink. Se fosse considerado que a maioria dos usuários tivesse que definir conectores, essa conclusão faria sentido. Entretanto, a idéia é fazer com que usuários experientes escrevam bases de conectores representando diversos tipos de relações hipermídia, tais como a mostrada na Figura 14. Usuários comuns só precisam usá-las para criar elos, escrevendo código XML similar ao apresentado na Figura 15, o que é bastante razoável e mais simples.

Note que pode haver problemas de consistência quando conectores são usados para criar elos estendidos XLink. Um exemplo é quando o número de

participantes, que compartilham o mesmo rótulo em um elo XLink, for menor que a cardinalidade mínima ou maior que a cardinalidade máxima de um papel associado a esse rótulo. Outro exemplo é quando um usuário utiliza um conector de forma errada e não define *binds* corretamente para todos os seus papéis. Problemas também podem acontecer quando um elo se refere a URI de um conector que se tornou inválida por um motivo qualquer. Em todos esses casos, é responsabilidade dos formatadores de documento identificar as inconsistências.

Pode-se ressaltar algumas vantagens de estender XLink incorporando as facilidades de XConnector (note, entretanto, que nenhuma vantagem do XLink é perdida):

- Aumento da expressividade da linguagem – XConnector permite a definição de novos tipos de regras de navegação representando relações de sincronização espaço-temporal multimídia com semântica causal ou de restrição:
  - usando XConnector, as regras de navegação podem realmente definir relações multiponto especificando condições e ações compostas relacionando diversos papéis;
  - além dos eventos usuais de apresentação e clique do mouse, outros tipos de evento podem ser usados para definir condições e ações, tais como pré-busca de conteúdo, por exemplo;
  - o ciclo de vida completo de um evento, como definido em sua máquina de estados, poder ser usado na definição de condições e ações, dando aos autores mais flexibilidade para especificar relações.
- Aumento do reuso da linguagem – um tipo de regra de navegação já definido, representado por um conector, pode ser reusado por diversos elementos do tipo *arc* no mesmo elo estendido XLink ou até em outros elos estendidos.
- Manutenção da facilidade de uso apesar do aumento de expressividade – aplicando a mesma idéia das *linkbases* de XLink para conectores, bases de conectores podem ser definidas para criar bibliotecas de conectores, que podem ser reusados para criar elos distintos.

- Facilidade de manutenção dos elos – quando um conector é modificado, todos os elos que o referenciam são automaticamente atualizados. Apesar de poder ser colocada como uma facilidade, essa característica também pode trazer dificuldades, pois quando um conector é modificado, pode ser que os elos que o utilizam devam ser modificados também, para refletir as mudanças em sua definição. Caso contrário, a definição do elo pode ficar inconsistente.

4.2 Extensões em Outras Linguagens

Assim como XConnector foi usado como módulo da linguagem NCL 2.0, e como parte das propostas de extensões ao padrão XLink para incorporar facilidades introduzidas pelo uso de conectores hipermídia, pode-se considerar a extensão de outras linguagens utilizando XConnector.

Comparando a metodologia proposta por conectores hipermídia com a modularização da linguagem XHTML, um perfil chamado XHTML+XConnector, poderia ser criado. Esse perfil permitiria a criação de outros tipos de relações na linguagem XHTML, além da relação tradicional de referência existente, que poderiam ser usados para a autoria de elos XHTML. Para dar suporte à definição de conectores, o módulo XConnector deveria ser adicionado ao perfil e um novo módulo teria que ser criado. Esse módulo novo deveria permitir a definição de elos XHTML através do uso de conectores, de forma similar ao feito no módulo *Linking* de NCL 2.0. Browsers XHTML teriam que ser adaptados para tratar especificações de sincronização feitas através de XConnector, e então XHTML finalmente teria o suporte necessário para especificar relações de sincronização através de elos.