# 3
# Eliminação das Redundâncias

O perfil EDTV define ao todo 45 elementos, alguns deles redundantes. Por exemplo, as propriedades *top*, *bottom*, *left*, *right*, que determinam o posicionamento do objeto de mídia na tela, podem ser definidas de três formas distintas: como propriedade do objeto de mídia, como parâmetro (ou atributo) do descritor associado ao objeto de mídia, ou como atributo da região associada ao descritor do objeto. Apesar dessa variedade de opções, em todos os casos o resultado final é o mesmo: a propriedade do objeto de mídia é inicializada com o valor especificado e o conteúdo do objeto é exibido na posição determinada. Este capítulo, trata, primeiramente, da definição dos procedimentos para eliminação desse tipo de redundância. Em seguida, a composição desses procedimentos é utilizada para obter um perfil mínimo, chamado NCL *Raw*, que é, ao mesmo tempo, livre de redundâncias e compatível com o EDTV. Dessa forma, documentos escritos nesse perfil podem ser exibidos pelos formatadores tradicionais. O capítulo está organizado da seguinte forma. A Seção 3.1 identifica as construções redundantes do EDTV e define os procedimentos de eliminação correspondentes. E a Seção 3.2 apresenta o perfil NCL Raw, definido a partir da composição desses procedimentos.



## 3.1
## Procedimentos de Eliminação

Esta seção identifica cada construção redundante do perfil NCL EDTV e define um procedimento de eliminação correspondente. Para cada elemento ou atributo redundante $r$ vamos definir um procedimento $\delta_r$ que o elimina. Seja $\Pi$ um documento EDTV que contém $r$. Então, o resultado da operação $\delta_r(\Pi)$ é um documento $\Pi'$ tal que $\Pi'$ é equivalente a $\Pi$, i.e. ambos definem a mesma apresentação, e $\Pi'$ não contém nenhuma ocorrência do elemento $r$.

### 3.1.1
### Elementos <importNCL> e <importedDocumentBase>

O elemento <importNCL> é usado para importar documentos NCL (cf. Seção 2.3.1). Esse elemento define dois atributos: *documentURI* e *alias*. O primeiro contém a URI do documento importado, e o último define o seu

rótulo de importação — i.e. o prefixo usado pelos elementos do documento importador para referenciar elementos do documento importado. O procedimento de eliminação do elemento <importNCL> consiste em expandir a sua definição [1]:

> "As bases [do documento importado] são tratadas como se cada uma tivesse sido importada por um elemento <importBase>. [...] O <body> importado, bem como quaisquer de seus nós, pode ser reusado dentro do <body> do documento NCL que realizou a importação."

Seja Π um documento EDTV e seja $I$ um elemento <importNCL> de Π que aponta para o documento externo Σ. O procedimento $\delta'_{\text{importNCL}}$ para eliminação de $I$ consiste dos seguintes passos:

1. Substituir o cada elemento <importNCL> por elementos <importBase> correspondentes. Ou seja, para cada base de elementos $B$ de Π é inserido um elemento <importBase> em $B$ tal que o seu atributo *documentURI* é igual ao *documentURI* de $I$, e seu atributo *alias* é uma nova *string* única no documento. Além disso, os atributos dos elementos elementos Π que fazem referência aos elementos do cabeçalho de Σ devem ser atualizados para usar o novo *alias*.

2. Expandir a definição dos elementos <media>, <context> ou <switch> de Σ que são referenciados por elementos de Π através do atributo *refer*. Ou seja, cada elemento <media>, <context> ou <switch> de Π que referencia algum elemento correspondente em Σ deve ser substituído pela definição do elemento referenciado, incluindo seus filhos. Além disso, para evitar conflitos de nomes e manter válidas as referências ao elemento importado, o *id* do elemento, o *id* dos seus filhos, e os atributos *refer* correspondentes devem ser atualizados.

O procedimento $\delta'_{\text{importNCL}}$ elimina um elemento <importNCL> de Π. O procedimento $\delta_{\text{importNCL}}$ consiste em aplicar $\delta'_{\text{importNCL}}$ ao documento até que todos os elementos <importNCL> tenham sido removidos. Finalmente, o procedimento $\delta_{\text{importedDB}}$, que elimina a base de documentos importados, consiste em aplicar $\delta_{\text{importNCL}}$ ao documento e, em seguida, remover o elemento <importedDocumentBase> do cabeçalho do documento.

### 3.1.2 Elemento <importBase>

O elemento <importBase> (cf. Seção 2.3.1) permite que bases de entidades de documentos externos possam incorporadas às bases de um documento qualquer. Além dos atributos *documentURI* e *alias*, o <importBase> possui um atributo *region*, usado para definir a região pai destinatária (no caso da importação de bases de regiões). O procedimento de eliminação do elemento <importBase> consiste em copiar os elementos definidos no documento importado para a base correspondente no documento de origem.

Seja $\Pi$ um documento EDTV qualquer e seja $B$ um elemento <importBase> de $\Pi$ que aponta para o documento externo $\Sigma$. O procedimento $\delta'_{\text{importBase}}$ para eliminação de $B$ considera os seguintes casos:

1. *B é filho do elemento <descriptorBase>*. Nesse caso, os descritores e *switches* de descritores definidos na base de descritores de $\Sigma$ são copiados para a base de descritores de $\Pi$ e seus atributos *id* e as respectivas referências são atualizadas. Além disso, as bases de regiões (pode haver mais de uma), a base de transições e a base de regras de $\Sigma$ também são importadas — casos (2), (3) e (5) abaixo —, já que descritores e *switches* de descritores de $\Sigma$ podem referenciar elementos definidos nessas bases.

2. *B é filho de um elemento <regionBase>*. Nesse caso, há duas possibilidades:

   (a) Se o atributo opcional *region* estiver definido, então os elementos das bases de regiões de $\Sigma$ devem ser inseridos como filhos do elemento <region> especificado — que pertence, necessariamente, a uma das bases de regiões de $\Pi$.

   (b) Caso contrário, o conteúdo das bases de regiões de $\Sigma$ deve ser copiado para a base correspondente em $\Pi$.

3. *B é filho do elemento <transitionBase>*. Basta copiar as transições definidas na base de transições de $\Sigma$ para a base de transições de $\Pi$. (Observe que todo documento define no máximo uma base de transições.)

4. *B é filho do elemento <connectorBase>*. Similar ao caso (3).

5. *B é filho do elemento <ruleBase>*. Similar ao caso (3).

Em todos os casos anteriores o *id* dos elementos importados é transformado em uma nova *string* única no documento e os atributos dos elementos de $\Pi$ que fazem referência a esses *id*s são atualizados. O procedimento anterior

elimina apenas um elemento <importBase> de Π. O procedimento $\delta_{\text{importBase}}$ consiste em aplicar $\delta'_{\text{importBase}}$ ao documento até que todos os elementos <importBase> tenham sido removidos.

### 3.1.3 Elementos <region> e <regionBase>

O elemento <region> (cf. Seção 2.3.2) define os parâmetros que determinam a dimensão e o posicionamento do objeto de mídia na tela. O procedimento de eliminação do elemento <region> consiste em transformar os seus atributos em parâmetros dos descritores associados. Isso funciona porque os atributos e parâmetros especificados na região e no descritor realizam a mesma função, i.e. eles inicializam as propriedades correspondentes dos objetos de mídia associados. Como os parâmetros definidos no descritor têm precedência sobre os atributos da região, apenas aqueles que não aparecem duplicados devem ser copiados.

Seja Π um documento EDTV qualquer e seja $R$ um elemento <region> de Π tal que $R$ define uma lista de atributos (exceto *id*) $a_1, \ldots, a_n$ em que cada atributo é um par $\langle \text{nome}, \text{valor} \rangle$. O procedimento $\delta'_{\text{region}}$ para eliminação de $R$ consiste em, para cada descritor $D$ que referencia $R$ através do atributo *region*, incluir em $D$ uma lista $p_1, \ldots, p_n$ de elementos <descriptorParam> tal que os atributos *name* e *value* de cada $p_i$ ($i \leq n$) correspondem, respectivamente, ao nome e valor definidos em $a_i$. Durante a avaliação de cada $a_i$ é preciso converter os valores relativos em valores absolutos — i.e. torná-los independentes da região pai. Observe que a base de regiões de $R$ pode conter o atributo opcional *device*, que especifica a classe dos dispositivos associados. Nesse caso, deve ser inserido em $D$ um elemento <descriptorParam> adicional, com o atributo *name* igual a "device" e o atributo *value* igual ao valor do *device* da base de regiões de $R$. Finalmente, o último passo do procedimento é remover o atributo *region* do descritor $D$.

A função $\delta'_{\text{region}}$ elimina um elemento <region> de Π. O procedimento $\delta_{\text{region}}$ consistem em aplicar a função anterior ao documento até que todas as regiões tenham sido removidas. Finalmente, o procedimento $\delta_{\text{regionB}}$ consistem em aplicar $\delta_{\text{region}}$ ao documento e, em seguida, remover todos os elementos <regionBase> do cabeçalho do documento.

### 3.1.4 Elementos <transition> e <transitionBase>

O elemento <transition> (cf. Seção 2.3.2) define um efeito de transição que pode ser associado ao início ou ao final da apresentação de uma interface. O procedimento para eliminação do elemento <transition> funciona de forma similar ao procedimento apresentado na Seção 3.1.3.

Seja $\Pi$ um documento EDTV qualquer e seja $T$ um elemento <transition> de $\Pi$ tal que $T$ define uma lista de atributos (exceto *id*) $a_1, \ldots, a_n$. O procedimento $\delta'_{\text{transition}}$ para eliminação de $T$ consiste em, para cada descritor $D$ que referencia $T$ através do atributo *transIn* (ou *transOut*), incluir em $D$ uma lista $p_1, ..., p_n$ de elementos <descriptorParam> tal que, para $1 \leq i \leq n$: (i) o atributo *name* de $p_i$ corresponde à *string* "transIn" — ou "transOut", caso $D$ referencie $T$ através desse atributo — concatenada ao nome de $a_i$; e (ii) o atributo *value* de $p_i$ corresponde ao valor de $a_i$.

No procedimento anterior, o valor do atributo *transIn* (ou *transOut*) pode ser uma lista de *id*s de transições separados por vírgula. Nesse caso, o índice de $T$ na lista (que começa em 0) precisa ser sufixado ao nome de cada elemento <descriptorParam> gerado. Por exemplo, considere o seguinte trecho de código:

```
<transition id="t1" type="barWipe" subtype="leftToRight"/>
<transition id="t2" type="fade" dur="5s"/>
<transition id="t3" type="clockWipe"/>
  ...
<descriptor id="d" transIn="t2" transOut="t1,t3"/>
```

Se aplicarmos o procedimento anterior a esse trecho o vamos obter o seguinte resultado:

```
<descriptor id="d">
  <descriptorParam name="transInType" value="fade"/>
  <descriptorParam name="transInDur" value="5s"/>
  <descriptorParam name="transOutType[0]" value="barWipe"/>
  <descriptorParam name="transOutSubType[0]" value="leftToRight"/>
  <descriptorParam name="transOutType[1]" value="clockWipe"/>
</descriptor>
```

A função $\delta'_{\text{transition}}$ elimina um elemento <transition> de $\Pi$. O procedimento $\delta_{\text{transition}}$ consiste em aplicar a função anterior ao documento até que todos os elementos <transition> tenham sido removidos. Finalmente, o procedimento $\delta_{\text{transitionB}}$ consiste em aplicar $\delta_{\text{transition}}$ ao documento e, em seguida, remover o elemento <transitionBase> do cabeçalho do documento.

### 3.1.5 Elemento <descriptor>

O elemento <descriptor> (cf. Seção 2.3.3) define as características audiovisuais dos objetos de mídia associados. Em geral, essas características são especificadas através de parâmetros, elementos <descriptorParam>. Porém, algumas delas, por exemplo *region*, *freeze* e *focusSrc*, podem ser definidas como atributos do elemento <descriptor>.

O procedimento de eliminação do elemento <descriptor> consiste, basicamente, em transformar os seus atributos e parâmetros em propriedades (elementos <property>) dos objetos de mídia associados. Como as propriedades definidas no objeto de mídia têm precedência sobre aquelas definidas no descritor, apenas as que não aparecem duplicadas devem ser copiadas. O procedimento definido a seguir assume que o descritor não possui os atributos *region*, *transIn* e *transOut*. Observe que esse procedimento não funciona para elementos <descriptor> declarados dentro de elementos <descriptorSwitch>. Esse último caso é tratado na Seção 3.1.6.

Seja $\Pi$ um documento EDTV qualquer e seja $D$ um elemento <descriptor> de $\Pi$ tal que $D$ possui uma lista de atributos (exceto *id*) $a_1, \ldots, a_m$ e uma lista de parâmetros $p_1, \ldots, p_n$ em que cada $a_i$ $(i \leq m)$ e $p_j$ $(j \leq m)$ denotam um par $\langle nome, valor \rangle$. O procedimento $\delta'_{\text{descriptor}}$ para eliminação de $D$ consiste de dois passos:

1. Transformar os atributos de $D$ em elementos <descriptorParam>. Ou seja, para cada atributo $a_i$ é inserido em $D$ um novo parâmetro $p$ cujos atributos *name* e *value* correspondem, respectivamente, ao nome e valor definidos em $a_i$.

2. Transformar os parâmetros de $D$ em propriedades das mídias associadas. Para cada mídia $M$ que referencia $D$ é inserida uma lista de propriedades em $M$ cujos atributos *name* e *value* de cada propriedade corresponde, respectivamente, ao nome e valor definidos no parâmetro $p_k$ $(k \leq m+n)$ do descritor $D$. Essas novas propriedades, assim como os parâmetros do descritor, não podem ser usadas diretamente em *links* e portanto devem declarar o atributo *externable* com valor "false".

A função $\delta'_{\text{descriptor}}$ elimina apenas um elemento <descriptor> de $\Pi$. O procedimento $\delta_{\text{descriptor}}$ consiste em aplicar a função anterior ao documento até que todos os descritores tenham sido removidos. Para eliminar a base de descritores do documento é preciso eliminar, além dos descritores, todas as ocorrências do elemento <descriptorSwitch>. O procedimento de eliminação desse elemento é assunto da próxima seção.

### 3.1.6 Elementos <descriptorSwitch> e <descriptorBase>

O elemento <descriptorSwitch> (cf. Seção 2.3.3) permite associar um conjunto de descritores alternativos a um objeto de mídia. A utilização de um determinado descritor depende da regra associada. As regras, elementos <rule> e <compositeRule>, são testes Booleanos sobre valores de propriedades do objeto de mídia *settings* (cf. Seção 2.3.6).

O procedimento de eliminação do elemento <descriptorSwitch> (e dos seus filhos) consiste em criar um elo para cada elemento <bindRule>, de forma que, se o objeto de mídia for apresentado (ação *start*) e o teste realizado pela regra for verdadeiro, então o valor de cada parâmetro do descritor correspondente deve ser atribuído a uma propriedade homônima do objeto de mídia (ação *set*). Para que esse procedimento funcione, é fundamental observar que:

1. A condição de cada elo gerado faz referência a uma ou mais variáveis do *settings*. Portanto, é preciso que esse nó esteja visível no mesmo contexto em que os elos são gerados.

2. O usuário pode especificar um descritor *default*, através do elemento <defaultDescriptor>, que é utilizado caso nenhuma regra seja avaliada como verdadeira.

3. A ordem de avaliação dos elos gerados é relevante. Ou seja, os elos gerados devem ser avaliados sempre na mesma ordem especificada pelos elementos <bindRule>.

Antes de definir precisamente o procedimento de eliminação do elemento <descriptorSwitch> precisamos definir um procedimento $\tau$ que converte uma regra $R$ qualquer (simples ou composta) em um elemento <assessmentStatement> $A$ equivalente. Para facilitar a apresentação do procedimento vamos assumir que cada propriedade do nó *settings* aparece no máximo uma vez em cada regra. Na prática, para evitar conflito entre os atributos *role* gerados, é preciso adicionar a cada *role* um índice que conta o número de ocorrências da propriedade.

O procedimento $\tau$ é definido da seguinte forma:

1. Se $R$ é uma regra simples com os atributos *variable* $= x$, *comparator* $= y$ e *value* $= z$, então $A$ é igual a:

```
<assessmentStatement comparator=y>
  <attributeAssessment role=x eventType="nodeProperty"/>
  <valueAssessment value=z/>
</assessmentStatement>
```

2. Caso contrário, $R$ é uma regra composta com uma lista $r_1, \ldots, r_n$ de filhos e com o atributo *operator* $= w$. Nesse caso, $A$ é igual a:

```
<compoundStatement operator=w>
  τ(r1) ··· τ(rn)
</compoundStatement>
```

Seja $\Pi$ um documento NCL qualquer e seja $S$ um elemento <descriptorSwitch> de $\Pi$ tal que $S$ possui uma lista $b_1, \ldots, b_n$ de elementos <bindRule> e uma lista $d_1, \ldots, d_n$ de elementos <descriptor>. Seja $M$, contido em um contexto $C$, um elemento <media> que referencia $S$ através do atributo *descriptor*. O procedimento $\delta'_{\text{descSwitch}}$ para eliminação de $S$ consiste dos seguintes passos:

1. Incluir em $C$ um novo objeto de mídia que reúsa o nó *settings*, caso esse nó ainda não exista.

2. Incluir em $C$ uma propriedade *counter* que será utilizada para forçar uma ordem de avaliação nos elos gerados no passo 4. O nome dessa propriedade deve ser único em $C$; seu valor inicial é irrelevante.

3. Adicionar o seguinte elo a $C$ (caso o conector referenciado não exista, ele também deve ser adicionado à base de conectores de $\Pi$):

```
<link xconnector="onBeginSet">
  <bind role="onBegin" component=M/>
  <bind role="set" component=C interface=counter>
    <bindParam name="val" value="1"/>
  </bind>
</link>
```

Esse elo funciona como um "gatilho" que dispara a avaliação das regras do <descriptorSwitch>.

4. Transformar cada <bindRule> em um par elo-conector equivalente. Ou seja, para cada elemento <bindRule> $b_i$ de $S$ ($i \leq n$) com atributo *constituent* $= d_i$ e regra associada $r_i$, inserir na base de conectores de $\Pi$ os seguintes conectores (com *id*s únicos $c_i$ e $c'_i$):

```
<causalConnector id=ci>
  <connectorParam name="val"/>
  <compoundCondition operator="and">
    <simpleCondition role="onEndAttribution"/>
    <assessmentStatement comparator="eq">
      <attributeAssessment role="testCounter"
```

```
        eventType="attribution" attributeType="nodeProperty"/>
      <valueAssessment value=i/>
    </assessmentStatement>
    τ(r_i)
  </compoundCondition>
  <simpleAction role="set" value="$val"
    max="unbounded" qualifier="seq"/>
</causalConnector>
```

Em que $c'_i$ é similar a $c_i$ porém, no lugar de $\tau(r)$, $c'_i$ utiliza $\neg\tau(r)$, i.e. a negação de $\tau(r)$. Para obter essa negação basta adicionar o atributo *isNegated* ao elemento <compoundStatement> mais externo de $\tau(r)$; ou, caso o elemento mais externo de $\tau(r)$ seja um *assessmentStatement*, substituir o valor do atributo *comparator* pela sua negação.

Além disso, os seguintes elos $l_i$ e $l'_i$ devem ser adicionados a $C$, em que $p_1, \ldots, p_m$ é a lista dos nomes de propriedades do nó *settings* que aparecem em $\tau(r)$, e $\langle k_1, v_1\rangle, \ldots, \langle k_j, v_j\rangle$ é a de nomes e valores dos $j$ parâmetros do descritor $d_i$ referenciado por $b_i$.

```
<link id=l_i xconnector=c_i>
  <bind role="onEndAttribution" component=C interface=counter/>
  <bind role="testCounter" component=C interface=counter/>
  <bind role=p_1 component=settings interface=p_1/>
    ...
  <bind role=p_m component=settings interface=p_m/>
  <bind role="set" component=M interface=k_1>
    <bindParam name="val" value=v_1/>
  </bind>
    ...
  <bind role="set" component=M interface=k_j>
    <bindParam name="val" value=v_j/>
  </bind>
</link>
<link id=l'_i xconnector=c'_i>
  <bind role="onEndAttribution" component=C interface=counter/>
  <bind role="testCounter" component=C interface=counter/>
  <bind role=p_1 component=settings interface=p_1/>
    ...
  <bind role=p_m component=settings interface=p_m/>
  <bind role="set" component=C interface=counter>
    <bindParam name="val" value=(i + 1)/>
  </bind>
```

```
</link>
```

O propósito de $l_i$ é simular a avaliação da regra $r_i$ associada a $b_i$. Se $r_i$ for avaliada como verdadeira então cada parâmetro do descritor $d_i$ referenciado por $b_i$ é transformado em uma ação *set* correspondente. Observe que para que $c_i$ funcione é preciso adicionar a $M$ os elementos <property> correspondentes (caso eles não existam). O *link* $l'_i$ é utilizado para dar continuidade a avaliação das regras caso a $r_i$ seja avaliado como falso.

5. Caso $S$ defina um elemento <defaultDescriptor> então um elo $l_{i+1}$ similar ao $l_i$ (definido no passo 4) deve ser adicionado a $C$. Este novo elo testa se valor de *counter* é $n+1$ e, em seguida, atribui a cada propriedade de $M$ o valor do parâmetro homônimo no descritor $d_{i+1}$, referenciado pelo elemento <defaultDescriptor> de $S$.

O procedimento $\delta'_{\text{descSwitch}}$ substitui cada elemento <descriptorSwitch> por conjunto de elos equivalentes. Esses elos funcionam da seguinte forma. Assim que o objeto de mídia $M$ é apresentado, o elo "gatilho" (criado no passo 3) dispara o processo de avaliação sequencial das regras. Cada regra $r_i$ possui dois elos associados: $l_i$ e $l'_i$. A condição de $l_i$ é verdadeira se, e somente se, a condição de $r_i$ é verdadeira e a regra corrente é a regra $i$. A condição de $l'_i$ é verdadeira se, e somente se, a condição de $r_i$ é falsa e a regra corrente é a regra $i$. Portanto, se $l_i$ é disparado então $l'_i$ não é disparado e vice-versa. Além disso, $l_i$ e $l'_i$ só podem ser verdadeiras se o valor de *counter* for $i$. Logo, cada par $l_i$ e $l'_i$ é avaliado em sequência até que um dos seguintes casos ocorra:

- Algum $l_k$ é avaliado como verdadeiro. Nesse caso, o processo de avaliação é interrompido (pois $l_i$ não atribui valor à *counter*), e as ações *set* correspondentes são executadas.

- O último elo $l'_n$ é avaliado como verdadeiro. Nesse caso, $l'_n$ atribui o valor $n+1$ a *counter*. O único elo que testa esse valor, caso ele exista, é o elo adicionado no passo 5, que atribui os parâmetros do elemento <defaultDescriptor> de $S$ ao objeto de mídia $M$. Se esse elo existir, o processo de avaliação é interrompido e as ações *set* correspondentes são executadas. Caso contrário, o processo de avaliação termina pois nenhum $l_i$ ou $l'_i$ $(1 \leq i \leq n+1)$ pode ser verdadeiro.

O procedimento anterior assume que o objeto de mídia $M$ que referencia o elemento <descriptorSwitch> é filho de um contexto. Dessa forma, se $M$ for

filho de um <switch>, antes de aplicar o procedimento é preciso converter o <switch> em um contexto usando o procedimento descrito na Seção 3.1.7.

A função $\delta'_{\text{descSwitch}}$ elimina apenas um elemento <descriptorSwitch> de $\Pi$. O procedimento $\delta_{\text{descSwitch}}$ consiste em aplicar a função anterior ao documento até que todos os elementos <descriptorSwitch> tenham sido removidos. Finalmente, o procedimento $\delta_{\text{descriptorB}}$ consistem em aplicar os procedimentos $\delta_{\text{descriptor}}$ e $\delta_{\text{descSwitch}}$ (nessa ordem), e, em seguida, remover o elemento <descriptorBase> do cabeçalho do documento.

### 3.1.7 Elementos <switch>

O elemento <switch> permite definir um conjunto nós alternativos cuja apresentação é associada a regras declaradas no cabeçalho do documento. Todo <switch> define um conjunto de interfaces, elementos <switchPort>, que mapeiam componentes internos do *switch*. O procedimento de eliminação do <switch> consiste, basicamente, em converter o *switch* em um contexto e usar portas e elos para simular os elementos <switchPort> e <bindRule> respectivamente. Observe que as restrições para simulação de regras através de elos, discutidas na Seção 3.1.6, também se aplicam no caso do elemento <switch>.

Seja $\Pi$ um documento NCL qualquer e seja $S$ um elemento <switch> de $\Pi$ tal que $S$ possui uma lista $b_1, \ldots, b_n$ de elementos <bindRule>, uma lista $m_1, \ldots, m_n$ de componentes, e uma lista de $p_1, \ldots, p_k$ de elementos <switchPort>, em que cada $p_i$ $(0 \leq i \leq k)$ mapeia algum subconjunto de mídias de $S$. O procedimento $\delta'_{\text{switch}}$ para eliminação de $S$ consiste dos seguintes passos:

1. Converter $S$ em um contexto $C$ contendo $m_1, \ldots, m_n$.

2. Incluir em $C$ uma nova mídia que reúsa o nó *settings*.

3. Transformar cada elemento <switchPort> $p_1, \ldots, p_k$ em uma porta correspondente em $C$. Cada $p_i$ $(0 \leq i \leq k)$ deve mapear um novo objeto de mídia *trigger* que servirá de âncora para os elos inseridos no passo 4. Ou seja, $k$ objetos de mídia *trigger* devem ser inseridos em $C$, e cada $p_i$ aponta para algum *trigger* distinto. Além disso, uma porta adicional, que representa o *switch* como um todo, deve ser incluída em $C$, junto com um novo *trigger*.

4. Transformar cada sequência de elementos <bindRule> associado lista de elementos <mapping> de cada <switchPort> em uma sequencia de

elos/conectores ancorados nos *triggers* correspondentes. Os elos/conectores avaliam as regras (na ordem correta) e, caso uma delas seja verdadeira, apresenta a mídia associada (ação *start*). Esse tipo de conversão é apresentada em detalhes na Seção 3.1.6.

5. Para cada ação diferente de "start", criar um elo ancorado no *dummy* que aplica a ação aos objetos de mídia mapeados por cada porta.

A função $\delta'_{\text{switch}}$ elimina apenas um elemento <switch> de $\Pi$. O procedimento $\delta_{\text{switch}}$ consiste em aplicar a função anterior ao documento até que todos os elementos <switch> tenham sido transformados em contextos.

## 3.2 Perfil Raw

A composição dos procedimentos apresentados na seção anterior define uma nova linguagem $L$, subconjunto do EDTV, com algumas características interessantes. Primeiro, a estrutura de $L$ é mais simples e consistente. Todo documento é formado a partir da combinação de seis conceitos primitivos: mídia, âncora, propriedade, contexto, porta e *link*[1]. Outra característica importante, é que, por construção, $L$ mantém ao mesmo tempo a expressividade e a compatibilidade com o EDTV. Ou seja, para todo documento EDTV existe um equivalente em $L$ que também é um documento EDTV válido. Essas propriedades fazem de $L$ uma base adequada para a definição de um novo perfil, chamado NCL *Raw*. A Tabela 3.1 apresenta a gramática do perfil *Raw*

| Elemento | Atributos | Conteúdo |
|---|---|---|
| <ncl> | *id, title, xmlns* | (head?, body?) |
| <head> | - | (connectorBase?, meta*, metadata*) |
| <body> | *id* | (port \| media\| context\| link\| meta\| metadata)* |
| <context> | *id* | (port\| media\| context\| link\| meta\| metadata)* |
| <port> | *id, component, interface* | - |
| <media> | *id, src, refer, instance, type* | (area\| property)* |
| <area> | *id, coords, begin, end, text, position, first, last, label* | - |
| <property> | *name, value, externable* | - |
| <link> | *id, xconnector* | (linkParam*, bind)+ |

**Tabela 3.1** Gramática do perfil NCL Raw.

Os elementos estruturais <ncl>, <head>, e <body> foram mantidos para preservar a compatibilidade – eles são obrigatórios no perfil EDTV. Po-

[1]Podemos considerar os conectores como parte da definição dos *links*.

rém, regiões, transições, descritores e *switch* de descritores foram removidos. A única forma de definir propriedades é, portanto, através do elemento <property>, que também é usado para definir transições. Os mecanismos de importação <importNCL> e <importBase> foram removidos da linguagem. Desta foram, o atributo *refer* só pode ser usado para referenciar *id*s definidos no mesmo documento. Além disso, o reúso sintático não é permitido – só é permitido o reúso de objetos de apresentação. O perfil Raw elimina todos os elemento relacionados com o elemento <switch> – inclusive as regras definidas no cabeçalho. A única maneira especificar alternativas é através das condições nos *links*. Com a eliminação das regras, os únicos elementos que restaram do cabeçalho são os conectores, que definem as relações usadas nos *links*.