1
Introdução

1.1
Motivação

NCL (*Nested Context Language*) é uma linguagem declarativa para criação de documentos hipermídia. Um documento NCL, é uma aplicação XML que descreve o relacionamento entre objetos de mídia (textos, imagens, vídeos, etc.) no espaço e no tempo. A especificação de NCL é feita através de perfis, em que cada perfil define um conjunto de elementos e atributos suportados. NCL 3.0, a versão mais recente da linguagem, define dois perfis para televisão digital: o perfil avançado, EDTV (*Enhanced Digital Television*), e o perfil básico, BDTV (*Basic Digital Television*), subconjunto do primeiro [1, 2]. O perfil EDTV foi projetado para facilitar a autoria de aplicações de TV digital interativa. Além da sincronização de eventos, ele permite ao autor criar documentos com conteúdo adaptável, suporte à múltiplo dispositivos de exibição, efeitos de transição e animações.

A máquina de apresentação NCL, ou *formatador*, é o programa que lê documentos NCL e usa as bibliotecas multimídia do sistema para gerar uma apresentação correspondente. A distância semântica entre esses documentos, especificações declarativas de alto-nível, e a interface de programação dessas bibliotecas, em geral imperativa e de baixo-nível, exige a definição de uma estrutura de dados intermediária, chamada *modelo de apresentação*, que captura os dados e guia a apresentação do documento.

O processo de apresentação de um documento NCL consiste, portanto, de dois passos: conversão do documento e apresentação do modelo resultante. Na arquitetura do formatador cada passo é executado por um módulo correspondente. O primeiro módulo, chamado *conversor*, recebe como entrada um documento NCL e produz uma instância do modelo de apresentação. A saída do conversor serve como entrada para o segundo módulo, chamado *player*, que, a partir da instância do modelo (e da entrada do usuário), gera uma apresentação interativa correspondente. A Figura 1.1 apresenta a arquitetura básica do formatador NCL.

Nessa arquitetura, o modelo de apresentação funciona como interface de comunicação entre conversor e *player*. De certa forma, a complexidade e o grau de abstração do modelo determinam a distribuição das tarefas entre os módulos

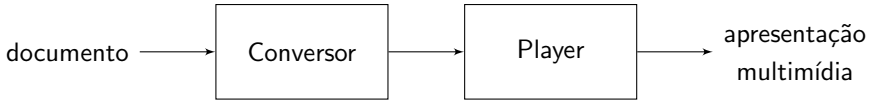

Figura 1.1 Arquitetura básica do formatador NCL.

do formatador. Um modelo próximo da linguagem simplifica o conversor, pois há pouca ou quase nenhuma manipulação estrutural, mas complica o *player*, que precisa transformá-lo em uma série instruções imperativas de baixo-nível. Por outro lado, um modelo próximo da máquina simplifica o *player*, pois aproxima o modelo das instruções, mas complica o conversor, que agora tem que converter as estruturas complexas do documento em primitivas do modelo. Essa interdependência entre conversor, modelo de apresentação e *player* sugere que a identificação do modelo adequado é um requisito fundamental para obtenção de um formatador eficiente e robusto.

A implementação mais recente do formatador NCL [3, 4], criada e mantida desde 2006 pelo Laboratório TeleMídia, é uma reescrita completa do formatador anterior HyperProp. A primeira versão do HyperProp [5] data de 1997. Na época, ainda não existia o NCL, apenas o NCM (*Nested Context Model*) [6], modelo conceitual que deu origem a linguagem. Em 2003, o HyperProp ganhou uma nova implementação capaz de apresentar documentos escritos na então recém-definida NCL [7]. Mais tarde, o desejo de expandir o alcance da linguagem para além dos computadores pessoais motivou o desenvolvimento da implementação atual, cujo objetivo principal era eliminar as limitações tecnológicas e arquiteturais do HyperProp que, de certa forma, impediam essa expansão. Em 2007, a implementação atual se tornou a implementação de referência do Ginga-NCL — *middleware* declarativo do sistema brasileiro de televisão digital [1].

Apesar das inovações, o modelo de apresentação da implementação de referência é praticamente idêntico ao do HyperProp, que tem como base o NCM. O problema com o NCM é que seu propósito original era facilitar a modelagem de cenários hipermídia e não facilitar uma apresentação eficiente desses cenários. Com isso, muitas das construções redundantes do NCM, projetadas para facilitar o processo de autoria, acabaram migrando para o modelo de apresentação das implementações. Em certo sentido, é natural que as implementações anteriores tenham utilizado o NCM como base para o seu modelo de apresentação. Primeiro, porque ele reflete a estrutura do documento. Além disso, o NCM foi amplamente testado e discutido — é, portanto, uma base segura. Finalmente, as versões mais recentes do NCM foram especificadas através de uma notação orientada a objetos, paradigma utilizado por ambas

as implementações.

A evolução da linguagem NCL é outro fator que contribui para o aumento da complexidade das implementações. Na medida em que funções eram adicionadas à linguagem, classes surgiam no modelo. Muitas dessas funções eram, na verdade, apenas formas alternativas de se especificar algum comportamento pré-existente. Porém, como não havia uma separação clara entre as funções primitivas e as demais, definidas a partir das primitivas, todas acabaram virando primitivas. Com o tempo, o modelo foi ficando cada vez maior e mais complexo. Na prática, essa complexidade se traduziu em mais código — atualmente 85 classes — e, consequentemente, em mais erros de codificação (ou *bugs*).

O elemento <switch> é um exemplo típico de construção redundante que acabou "migrando" para o modelo de apresentação das implementações. O <switch> de NCL funciona de forma semelhante ao *switch* das linguagens imperativas. Dada uma condição, um conjunto de casos é especificado. Se o valor da condição for igual ao valor de um dos casos então o código associado ao caso é executado — ou, em NCL, o conteúdo associado ao caso é apresentado. Da mesma forma que em C podemos substituir um *switch* por um conjunto de instruções *if-else*, em NCL podemos simular o comportamento do <switch> usando construções mais simples. Apesar disso, o <switch> possui representação correspondente tanto no modelo de apresentação quanto na API do *player* da implementação de referência.

A forma óbvia de simplificar o modelo de apresentação atual é simplificar a linguagem, i.e. remover do perfil EDTV tudo aquilo que é redundante. Como é de se esperar, essa linguagem mais simples produz um modelo mais simples. O problema com essa abordagem é que ela prejudica o autor de documentos, pois retira da linguagem construções projetadas para facilitar o processo de autoria.

A outra alternativa é delegar ao módulo conversor o trabalho de remover o que não é essencial. Ou seja, fazer do conversor o responsável por traduzir cada construção redundante do EDTV em uma série de primitivas e gerar o modelo de apresentação a partir do documento resultante. Dessa forma, as construções redundantes podem ser vistas como macros que quando expandidas geram o documento "real". Podemos identificar duas fases nesse processo:

1. *pré-processamento*, em que o documento EDTV é transformado em outro equivalente, porém mais simples, e

2. *parsing*, em que esse documento "bruto" é convertido no modelo de apresentação.

Apesar de ser (provavelmente) maior que o original, o novo documento é mais simples e, portanto, induz um modelo de apresentação mais simples. Do ponto de vista do autor, o perfil EDTV permanece inalterado.

Além de simplificar o modelo, a última abordagem traz alguns benefícios interessantes. Por exemplo, as fases podem ser executadas em momentos distintos. Além disso, o resultado do pré-processamento pode ser armazenado para agilizar (ou mesmo evitar) o processo de conversão. Outra vantagem é que a separação em fases mantém a linguagem da autoria isolada da linguagem da máquina de apresentação. Ou seja, mudanças no EDTV que não alteram a sua expressividade afetam apenas o conversor. Essa adaptabilidade é um requisito importante para o formatador NCL, visto que a linguagem está em constante evolução.

1.2 Objetivos

O objetivo geral desta dissertação é delimitar com precisão o menor subconjunto de elementos do perfil EDTV da NCL capaz de representar, de forma equivalente[1], qualquer documento EDTV válido. Todo documento no perfil básico BDTV é um documento EDTV válido. Portanto, tudo o que esta dissertação define sobre o EDTV vale também para o perfil BDTV. Dentre os objetivos específicos da dissertação podemos destacar:

- identificação dos elementos e atributos redundantes do EDTV,
- definição um procedimento de eliminação para cada redundância identificada,
- definição do menor perfil (em número de elementos) de expressividade equivalente, porém compatível com o EDTV — i.e. todo documento nesse perfil é também um documento EDTV,
- implementação de um conversor de documentos que permita reusar e/ou combinar os diversos procedimentos de eliminação.

A caracterização do subconjunto essencial do EDTV é um requisito fundamental para a definição de um modelo de apresentação mais simples. Apesar disso, a definição desse modelo não é objeto da dissertação, visto que essa definição envolve outros fatores que dependem da implementação do formatador.

[1] Dois documentos NCL são *equivalentes* se, e somente, ambos especificam a mesma apresentação.

1.3 Organização da Dissertação

Esta dissertação está organizada da seguinte forma. O Capítulo 2 apresenta a sintaxe e a semântica dos elementos que compõem o perfil EDTV da NCL. O Capítulo 3 identifica cada redundância presente no EDTV e define um procedimento de eliminação correspondente. Ainda nesse capítulo, é definido um perfil mínimo, compatível com o EDTV, chamado NCL *Raw*. O Capítulo 4 apresenta a arquitetura de um conversor de documentos EDTV para documentos *Raw*, e discute em detalhes a sua implementação. Finalmente, o Capítulo 5 apresenta as contribuições da dissertação e propostas de trabalhos futuros. A dissertação possui ainda dois apêndices. O Apêndice A contém a gramática completa do perfil EDTV, e o Apêndice B apresenta uma biblioteca *players* multimídia inspirada no perfil NCL *Raw*.