LindA: Uma Linguagem de Autoria Automática para Hipertexto

Judith Kelner
Ana Lúcia Cavalcanti
Alberto Pardo

Departamento de Informática
Universidade Federal de Pernambuco
Caixa Postal 7851, 50739 Recife - PE - Brasil

Resumo

Autoria Automática é uma solução para o problema da Qualidade de Autoria nos sistemas de hipertexto. Ela tem em LindA (Linguagem dos Autores de hipertexto) uma interface entre o usuário e o sistema de hipertexto.

Este artigo apresenta uma introdução a hipertexto, o porquê linguagens de *markup* (linguagens de "marcação" de textos) e introduz LindA com seus principais comandos, sua sintaxe e uma semântica parcial.

1 Introdução a Hipertexto

A idéia de hipertexto não é nova. Em 1945 Vannevar Bush, Consultor de Ciência do Presidente Roosevelt, introduzia alguns conceitos de hipertexto num artigo intitulado "As We May Think" [1], onde descrevia um sistema, *Memex*, como um suplemento para a memória do usuário, no qual textos e gráficos (notas, fotografias, desenhos, etc) seriam armazenados utilizando índices para recuperação de informações com a mesma função dos "links" de um hipertexto.

As idéias de Bush foram colocadas em prática na década de 60, quando Douglas Engelbart projetou o NLS (On-Line System) [1]. O NLS, cuja atual versão é chamada "Augment", foi o primeiro sistema de informação em um computador a utilizar múltiplas janelas e compartilhar telas, introduzindo o uso em ambientes distribuídos de editores estruturados.

O termo "hypertext" foi utilizado pela primeira vez por Ted Nelson na descrição do "Xanadu" [1,5] - um sistema eletrônico de publicações.

Atualmente, existe uma série de sistemas de hipertexto, entre os quais podemos citar: Notecards [1] da Xerox, Intermedia [3,6] da Universidade de Brown, ZOG [7] da Universidade de Carnegie-Mellon, GUIDE [4,8,9,10,11] da Universidade de Kent.Tais sistemas estão disponíveis para PCs e Macs, e o Hypercards [12] da Apple para Mac.

O conceito de hipertexto é bem simples [1]: vistas de uma tela de uma estação de trabalho, janelas são associadas a objetos em um Banco de Dados. Existem ligações físicas e gráficas entre estes objetos dando ao texto, a critério do autor, uma estrutura, a qual na maioria dos sistemas de hipertexto, pode ou não ser seguida pelo leitor, ou seja, o leitor pode criar visões pessoais para um documento.

Um sistema de hipertexto não precisa se limitar a armazenar textos. Qualquer tipo de informação que pode ser armazenada eletronicamente pode fazer parte de um hipertexto.

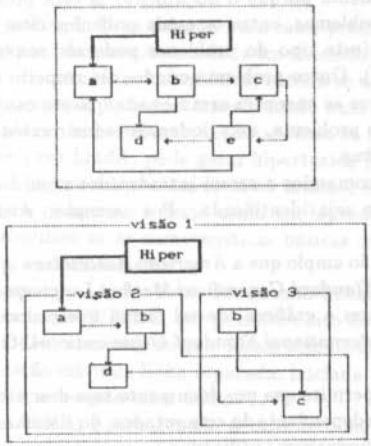

Figura 1: Representação pictorial de um hipertexto (topo) e representação de visões (base).

O termo hipertexto vem sendo empregado largamente, inclusive em alguns sistemas que não possuem as características essenciais de hipertexto. Dentre estes sistemas podemos citar: sistemas com janelas múltiplas, correio eletrônico e gerenciadores de teleconferência.

As principais características de um sistema de hipertexto são:

- facilidade de uso;
- utilização de recursos gráficos;
- visualização de um documento como um grafo;
- transparência da representação interna dos dados;
- capacidade de interligar documentos;
- interação grande entre o usuário e o documento, permitindo inclusive que o usuário crie visões pessoais do documento;
- estrutura dinâmica e sensível ao contexto do documento;
- possíbilidade de criar versões;
- simplificação do processo de documentação.

O uso mais geral de hipertexto é para documentos não-lienares ou não-sequenciais, o que não impossibilita a sua utilização em documentos sequenciais.

2 O Porquê Markup

Com o crescente uso de microcomputadores, tornou-se um hábito a utilização de editores de textos e a produção de documentos eletrônicos. Este fato reduz algumas etapas no processo

de editoração de um documento, principalmente porquê o documento já está pronto para ser editado. Entretanto isso introduz novos problemas, entre os quais podemos citar a incompatibilidade dos dispositivos computacionais (este tipo de problema pode ser resolvido usando interface para softwares ou dispositivos [15]). Outro problema criado, diz respeito ao conteúdo do documento, ou seja, qual a informação que se encontra armazenada? Neste caso, não existe ainda uma solução completa para resolver o problema, mas podemos reduzir substancialmente esse problema utilizando linguagens de *markup*.

Uma linguagem de *markup* consiste de comandos a serem introduzidos num documento de tal forma que a estrutura lógica do mesmo seja identificada. Por exemplo: comandos para identificar títulos, secções, parágrafos, etc.

O uso deste tipo de linguagem tornou-se tão amplo que a *American Association of Publishers*-AAP desenvolveu um protocolo chamado de *Standard Generalized Markup Language*-SGML para suportar uma interface entre autores, editores e gráficos, de tal forma que o documento seja produzido de uma só vez, e pelo autor. A *International Standard Organization*-ISO padronizou o SGML.

SGML é uma linguagem de *markup* que permite que um documento seja descrito a partir de sua estrutura. Este fato torna o documento independente do computador, do sistema operacional e do formato (ponto mais importante desta linguagem) utilizados.

Em SGML, um documento é visto como um elemento que, por sua vez, será composto de outros elementos. A relação existente entre estes elementos formam a estrutra do documento.

A linguagem SGML é constituída de identificadores genéricos (GIs-generic identifiers) que são utilizados para identificar cada elemento do documento. Os elementos podem possuir também atributos de valores. As GIs não fazem nenhuma referência ao formato do documento. Este tipo de especificação será realizado quando da editoração do documento.

A seguir apresentamos um exemplo de um documento em (uma transliteração portuguesa de) SGML, com a numeração das linhas incluída somente para referência [15].

```
1  <! ELEMENT artigo(titulo, autor, paragrafo*) >
2  <! ELEMENT (titulo | paragrafo) (#CHARS) >
3  <! ELEMENT autor(#CHARS) tipo(principal | co-autor) >
4  < artigo >
5  < titulo >
6  Padrao da SGML proposto pela ISO
7  <autor tipo = ''principal'' >
8  Le Van Huu
9  < paragrafo >
10 Este e o primeiro paragrafo
11 </ paragrafo >
12 </ artigo >
```

Observamos com o exemplo acima que escrever documentos usando SGML consiste de duas fases: a) definição do tipo do documento (ou seja da sua estrutura lógica) e b) *markup* do texto de acordo com a estrutura lógica definida, criando uma estrutura física do documento.

3 LindA: Uma Linguagem para Escrever Hiperdocumentos

Um dos problemas sérios em sistemas de hipertexto é a ausência de uma linguagem de apoio para autoria de um documento. Em [2] Brown diz que: *"Hypertext documents are hugely different from paper documents, and it is totally unrealistic to expect authors immediately to master*

the art of hypertext writing". Uma solução para estes problemas é a utilização de LindA, uma linguagem de *markup* para Autoria Automática em sistemas de hipertexto como apoio na autoria de hiperdocumentos. Por Autoria Automática, enteda-se a elaboração de documentos escritos de uma forma linear, sendo a posteriori transformado num hiperdocumento. Usando LindA, o autor não precisa conhecer as peculiaridades do sistema de hipertexto que irá usar. Além disso, o mesmo texto "fonte", em LindA, pode gerar hipertextos para diferentes sistemas. O texto de entrada para LindA é linear, acrescentando-se ao mesmo, comandos estruturais de hipertexto. O compilador de LindA transcreve o texto como um hiperdocumento.

Neste trabalho, abordam-se as características básicas de LindA com os seus principais comandos, sua sintaxe e sua semântica.

Um pre-texto escrito em LindA é um documento linear, e pode ser elaborado usando um editor de texto (por exemplo WRITE, que permite o uso de texto e "graphics"). A forma geral do pre-texto é a de um texto normal, ao qual foram adicionados os comandos de LindA. Todos os comandos de LindA estão em uma linha separada, iniciada pelo caractere "`>`". Os comentários podem ser introduzidos e deverão estar entre "`//`".

Um pre-texto, em LindA, é armazenado dentro de uma `Lib` (Biblioteca de documentos em LindA), onde cada documento será identificado por um rótulo.

Num pre-texto, será necessário "marcar" as estruturas existentes no hipertexto, isto é, os nós, os links, os rodapés, etc, devem ser identificados no pre-texto para permitir ao compilador de LindA criar o hiperdocumento.

A seguir apresentamos uma descrição sucinta dos principais comandos de LindA, após o que são mostrados alguns exemplos triviais do uso da linguagem.

- `Lib b1` introduz `b1` como o nome da biblioteca onde será inserido o documento definido pelo pre-texto que se segue;
- `Doc d1` define `d1` como o nome do documento em pauta;
- `Node n1 (v1, ..., vn) tag` é o cabeçalho de um nó, onde `n1` é o nome do nó, que, como veremos, é opcional. Os `vi` são os nomes das visões nas quais o nó é vísivel. `Tag` pode ter os valores `t`, indicando um nó do tipo texto, sendo este o *default*, `g` quando o nó for do tipo gráfico e `c` para código;
- `Foot (Palavras) (Rodape)` identifica um rodapé associado a uma palavra ou sentença de um nó;
- `But (str1:n1)` define um link entre nós, onde `str1` identifica o string que fará a referência ao nó `n1`;
- `Alt string (str1:n1, ... , strm:nm) stri` define um link alternativo para o nó `ni`;
- `Exp (texto1) (texto2)` cria uma expansão `texto2` dentro de um nó para o `texto1`;
- `End` define o fim de um pre-texto ou fim de comando em LindA.

Um exemplo trivial de um pre-texto em LindA está descrito a seguir:

```
>Lib Biblioteca_LindA

>Doc Exemplo1_LindA
```

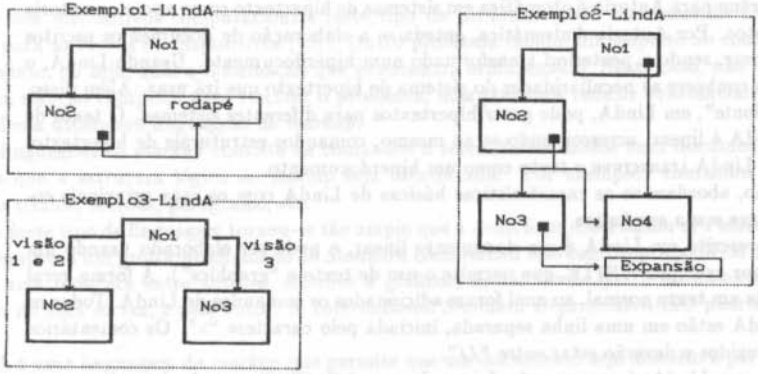

Figura 2: Representação pictórica dos hipertextos gerados pelos Exemplos 1, 2 e 3 de pre-texto em LindA.

```
>Node No1 t
Aqui temos um pre-texto escrito em LindA. Este e o primeiro no do documento,
e e um no do tipo texto.

>Node No2
Este e o segundo no do documento. O tipo deste no tambem e texto, porque
texto e o valor
>Foot (default) (default e o valor assumido quando um parametro for omitido)
Dentro deste no existe um rodape.

>End
```

Um outro exemplo de um pre-texto em LindA define o uso de alguns dos principais comandos da linguagem.

```
>Lib Biblioteca_LindA

>Doc Exemplo2_LindA

>Node No1 t
Neste ponto temos um exemplo de um pre-texto onde descreveremos os principais
comandos de
>But (LindA:No4)

>Node No2 t
Por enquanto os nos definidos no pre-texto sao do tipo texto, so por questao
de ilustracao. Voce pode introduzir no seu pre-texto questoes, onde a resposta
levara a caminhos diferentes. O comando Alt exemplifica isto.
```

```
>Alt Voce deseja visualizar um no com expansao? (Sim:No3,Nao:No4)

>Node No3
Em alguns casos desejamos ter uma expansao de um pedaco de um texto. LindA
permite esta expansao e tem um comando especifico.
>Exp (o comando Exp) (o comando Exp permite que um texto seja conectado a sua
expansao, permitindo ao usuario visualizar a informacao compacta ou expandida)

>Node No4
LindA e uma linguagem de markup para Autoria Automatica em sistemas de
hipertexto. O compilador de LindA transcreve um pre-texto para um hiperdocumento.

>End
```

LindA apresenta alguns comandos que poderão ser definidos globalmente para um documento, isto é, uma vez definido em qualquer ponto do documento pode haver uma referência ao mesmo. Estes comandos são opcionais.

- `Vis (v1, ... , vn)` onde `vi` é um string que identifica quais as visões do documento;
- `Key (k1, ... , kn)` onde `ki` é um string que identifica as palavras chaves existentes no documento, permitindo um posterior uso pelo comando FIND;
- `Inc Node n1` onde `(n1)` é o nome do nó que será incluido neste ponto do documento;
- `Inc Doc a1` onde `(a1)` é o nome do arquivo que será incluído neste ponto do documento;
- `Def Node n1 (p1 @ ... @ pn) (v1, ... , vn) ... ^ pi^ ...` End define um nó que será copiado pelo comando `Use`, onde `pi` identifica os parâmetros e `vi` identifica as visões do nó, `@` é o caracter de separação;
- `Ord n1 ... Nex n2` define uma nova ordem para os nós, onde `(n1)` é o nome do primeiro nó da nova sequência, `(n2)` aponta para um nó da sequência anterior ou para um fim de documento se `n2` não for dado;
- `Use n1 ('str1' @ ... @ 'strm') n2` inclui uma cópia do nó `n1` definido no comando `Def Node`, passa os parâmetros `'stri'` e `n2` será o nome do nó.

LindA permite ao autor definir visões diferentes para um documento. Por exemplo, num documento, dependendo do usuário, poderá ocorrer que alguns nós não sejam visíveis para este usuário. Além das visões, LindA permite a definição de palavras chaves no pre-texto para um posterior uso no hiperdocumento pelo comando FIND. A seguir exemplificamos o uso de alguns comandos opcionais de LindA.

```
>Lib Biblioteca_LindA

// este e um comentario num pre-texto em LindA //

>Doc Exemplo3_LindA
```

```
>Vis (Visao1, Visao2, Visao3)

// Este documento possui tres visoes diferentes//

>Key (LindA, visivel, usuario)

// Este comando define LindA, visivel e usuario como palavras chaves //

>Node No1 (Visao1, Visao2, Visao3)
Este exemplo mostra o uso de alguns comandos opcionais de LindA. Dependendo
do grau de conhecimento do usuario ele tera o seu acesso restrito. Este no
e visivel por todos os usuarios.

>Node No2 (Visao1, Visao2)
Este no sera visivel apenas para as visoes 1 e 2. A visao 3 nao tem acesso
ao mesmo.

>Node No3 (Visao3)
Este e um no que so sera visivel pela visao 3.

>End
```

Escrever documentos em LindA é o mesmo que escrever documentos sequenciais ou lineares e acrescentar ao texto uma linguagem de *markup*. O usuário de LindA só precisa estar familiarizado com os comandos de LindA e consequentemente com a estrutura de hipertexto que será gerada por cada um desses comandos.

LindA permite ao usuário converter os seus documentos lineares em hiperdocumentos sem que o mesmo conheça as peculiaridades do hipertexto a ser utilizado.

4 Conclusões

LindA, uma linguagem para Autoria Automática em hipertexto, tem como objetivo principal facilitar ao autor na elaboração de hiperdocumentos, primordialmente aqueles que não estão familiarizados com a estrutura dos hiperdocumentos.

LindA objetiva também minimizar os problemas de transferência dos documentos elaborados linearmente ou sequencialmente em hiperdocumentos. Isto é possível porque os comandos de LindA podem ser utilizados em qualquer tipo de estrutura de documento.

O compilador de LindA, em fase final de elaboração, poderá gerar código objeto para uma classe de sistemas de hipertexto, permitindo ao autor produzir hiperdocumento para o sistema de hipertexto que mais lhe for conviniente.

Agradecimentos

Agradecemos ao Prof. Silvio Meira (DI/UFPE) pela sua colaboração na elaboração deste trabalho. A apresentação no IFIP-Petrópolis do Prof. Egidio Astesiano (Genoa) foi fundamental para a elaboração da semântica da linguagem.

Apêndice A – Definição formal de LindA

A necessidade de definir precisamente uma linguagem que satisfaça aos objetivos para os quais ela foi projetada nos levou a especificar formalmente uma sintaxe e uma semântica para LindA.

A.1 Sintaxe de LindA

A sintaxe de LindA é definida a seguir com o uso do tradicional método BNF (Backus-Naur Form).

```
PRETEXTO    ::= Lib ID Doc ID CORPO End
              | Lib ID Doc ID Vis (VISAO) CORPO End
              | Lib ID Doc ID Key (PALAVRAS) CORPO End
              | Lib ID Doc ID Vis (VISAO) Key (PALAVRAS) CORPO End
              | Lib ID Doc ID Key (PALAVRAS) Vis (VISAO) CORPO End

CORPO       ::= NO | INC | DEF_NODE | USE | ORD | CORPO CORPO

NO          ::= NO_C/ROT | NO_S/ROT

INC         ::= Inc Node ID | Inc Doc ID

DEF_NODE    ::= Node ID CORPO_DEF End
              | Def Node ID (PARAMETROS) CORPO_DEF End
              | Def Node ID (VISAO) CORPO_DEF End
              | Def Node ID (PARAMETROS) (VISAO) CORPO_DEF End

USE         ::= Use ID (ARGUMENTOS) | Use ID (ARGUMENTOS) ID

ORD         ::= Ord ID NO_S/ROT CORPO Next ID
              | Ord ID NO_S/ROT CORPO Next
              | Ord ID NO_S/ROT Next ID
              | Ord ID NO_S/ROT Next

NO_C/ROT    ::= Node ID CONTEUDO | Node ID VISAO CONTEUDO

NO_S/ROT    ::= Node CONTEUDO | Node VISAO CONTEUDO

CORPO_DEF   ::= ^ ID ^  CORPO_DEF
              | CORPO_DEF ^ ID ^
              | CONTEUDO
              | CORPO_DEF CORPO_DEF

CONTEUDO    ::= CRAFICO | CODIGO | TEXTO | CONTEUDO CONTEUDO

TEXTO       ::= PALAVRAS | BUT | ALT | EXP | FOOT | TEXTO TEXTO

BUT         ::= But (PALAVRAS : ID)
```

```
ALT          ::= Alt PALAVRAS (ALTERNATIVAS)

EXP          ::= Exp (TEXTO) (TEXTO)

FOOT         ::= Foot (PALAVRAS) (RODAPE)

RODAPE       ::= PALAVRAS | BUT | ALT | EXP_FOOT | RODAPE RODAPE

EXP_FOOT     ::= Exp (RODAPE) (TEXTO)

VISAO        ::= PALAVRAS

PALAVRAS     ::= PALAVRA | PALAVRAS PALAVRAS

ALTERNATIVAS::= PALAVRAS : ID | ALTERNATIVAS ALTERNATIVAS

CODIGO       ::= FONTE | OBJETO

ARGUMENTOS   ::= PALAVRAS | ARGUMENTOS @ ARGUMENTOS

PARAMETROS::= ID | PARAMETROS @ PARAMETROS
```

A.2 Uma Semântica Parcial de LindA

A semântica indutiva estrutural (SIS) [13] de LindA é parcialmente apresentada a seguir. SIS é uma nova versão da Semântica Operacional Estruturada (SOS) introduzida e desenvolvida por Plotkin [14].

A SIS de uma linguagem é uma definição indutiva em um sistema lógico de dedução guiada pela sintaxe abstrata da linguagem. SIS permite definições por sistemas indutivos que descrevem diferentes estratégias de computação em vários níveis de detalhe. Por exemplo, é possível escrever definições semânticas com o uso de sistemas "big-", "small-" ou "mixed-step".

O sistema utilizado para a definição da SIS de LindA é o "big-step", o qual define a avaliação dos objetos sintáticos diretamente, sem passos intermediários.

Uma SIS é um conjunto de relações em um sistema indutivo. As cláusulas que definem cada relação são guiadas pela estrutura sintática dos objetos cuja semântica está sendo definida.

Alguns dos domínios semânticos utilizados na definição da SIS de LindA são dados na Tabela 1.

Na Tabela, $(\mathtt{A} \rightarrow \mathtt{B})_f$ é o domínio dos mapeamentos finitos de A para B. $\mathtt{A}^*$ é o domínio das listas de elementos de A.

Um documento é uma lista de seqüências de nós ($\mathtt{No}^*$). Cada seqüência pode estar ligada a outra por meio de um apontador (**Seguinte**) para um nó de outra seqüência.

O ambiente associa a cada nome de definição de nó o corpo da definição e a lista de parâmetros.

DOMÍNIO	É UM MODELO DE
$\texttt{Libs} = (\texttt{Id} \rightarrow \texttt{Lib})_f$	conjunto de bibliotecas
$\texttt{Lib} = (\texttt{Id} \rightarrow \texttt{Doc})_f$	biblioteca
$\texttt{Doc} = (\texttt{No}^* \times \texttt{Seguinte})^*$	documento
$\texttt{No} = \texttt{Nome} \times \texttt{Conteudo}$	nó de um documento
$\texttt{Nome} = \texttt{Id} \cup \{\texttt{S/NOME}\}$	nome de um nó
$\texttt{Seguinte} = \texttt{Id} \cup \{\texttt{FIM}\}$	apontador de nó
$\texttt{Conteudo} = (\texttt{Grafico} \cup \texttt{Fonte} \cup \texttt{Objeto} \cup \texttt{Texto})^*$	conteúdo de um nó
$\texttt{Env} = (\texttt{Id} \rightarrow \texttt{CORPO\_DEF} \times \texttt{Id}^*)_f$	ambiente formado pelas definições de nós
$\texttt{Texto} = (\texttt{Palavras} \cup \texttt{But} \cup \texttt{Alt} \cup \texttt{Exp} \cup \texttt{Foot})^*$	texto contido em um nó
$\texttt{Arquivos} = (\texttt{Id} \rightarrow \texttt{CORPO})_f$	sistema de arquivos

Tabela 1: Alguns domínios semânticos.

Regras semânticas

Na avaliação de um pre-texto ($\mathcal{P}$), o corpo c do documento $\texttt{id}_2$ é avaliado e armazenado na biblioteca $\texttt{id}_1$ do sistema.

$$\frac{< \texttt{c},\ [([\,],\ \texttt{FIM})],\ \{\} > \xrightarrow[arqs]{\mathcal{C}} <d,\ e> \qquad \texttt{c} \xrightarrow{\mathcal{I}} <iu,\ ir>}{<\textbf{Lib}\ \texttt{id}_1\ \textbf{Doc}\ \texttt{id}_2\ \texttt{c}\ \textbf{End},\ libs> \xrightarrow[arqs]{\mathcal{P}} libs[v/\texttt{id}_1]}$$

$$\texttt{id}_1 \in \texttt{dom}(libs) \qquad \texttt{id}_2 \notin \texttt{dom}(libs(\texttt{id}_1)) \qquad v = libs(\texttt{id}_1)[d/\texttt{id}_2]$$

$$ir \subseteq iu \ \ (\texttt{Os identificadores referidos tem que ter sido declarados})$$

A relação de avaliação de um corpo ($\mathcal{C}$) é definida em termos das relações que avaliam os diferentes tipos de corpos. Por exemplo, para um corpo do tipo NO tem-se:

$$\frac{\texttt{cn} \xrightarrow{\mathcal{NO}} no}{< \texttt{cn},\ d,\ e> \xrightarrow[arqs]{\mathcal{C}} <d',\ e'>} \qquad d' = (nos +\!\!+ [no], seg)\ \ \textbf{cons}\ \ \texttt{tl}(d) \qquad \texttt{hd}(d) = (nos,\ seg)$$

A avaliação de uma seqüência de corpos é dada por:

$$\frac{< \texttt{c}_1, d,\ e> \xrightarrow[arqs]{\mathcal{C}} <d_1,\ e_1> \qquad < \texttt{c}_2,\ d_1,\ e_1> \xrightarrow[arqs]{\mathcal{C}} <d',\ e'> \qquad \texttt{c}_1 \xrightarrow[arqs]{\mathcal{IU}} iu \qquad \texttt{c}_2 \xrightarrow[arqs]{\mathcal{IU}} iu'}{< \texttt{c}_1\ \texttt{c}_2,\ d,\ e> \xrightarrow[arqs]{\mathcal{C}} <d',\ e'>}$$

$$iu \cap iu' = \emptyset \ \ (\texttt{Não deve haver choque de identificadores})$$

onde $\mathcal{IU}$ resulta nos identificadores usados no corpo.

Todo comando **Ord** define como primeiro nó da seqüência sendo definida um nó sem nome. Este nó é avaliado ($\mathcal{NOSR}$) à parte, e lhe é atribuído o nome $\mathtt{id}_1$. Neste contexto o corpo c é avaliado.

$$\frac{\mathtt{nsr} \xrightarrow{\mathcal{NOSR}} cont \qquad < \mathtt{c},\ [([(\mathtt{id}_1,\ cont)],\ \mathtt{id}_2)],\ e > \xrightarrow[arqs]{\mathcal{C}} < d,\ e' >}{< \mathbf{Ord}\ \mathtt{id}_1\ \mathtt{nsr}\ \mathtt{c}\ \mathbf{Next}\ \mathtt{id}_2,\ e > \xrightarrow[arqs]{\mathcal{ORD}} < d,\ e' >}$$

A avaliação do comando **Inc Doc** é o resultado da avaliação do corpo do documento contido no arquivo $\mathtt{id}_1$ do sistema.

$$\frac{< arqs(\mathtt{id}),\ d,\ e > \xrightarrow[arqs]{\mathcal{C}} < d',\ e' >}{< \mathbf{Inc\ Doc}\ \mathtt{id},\ d,\ e > \xrightarrow{\mathcal{INC}} < d',\ e' >} \qquad \mathtt{id} \in \mathrm{dom}(arqs)$$

A avaliação do comando **Def Node** altera o ambiente pela inclusão de mais uma definição de nó. Os parâmetros usados no corpo da definição devem estar todos na lista de parâmetros e esta não deve ter parâmetros repetidos. $\mathcal{PAR}$ avalia a lista de parâmetros e $\mathcal{PU}$ resulta na lista de parâmetros usados no corpo da definição.

$$\frac{\mathtt{ps} \xrightarrow{\mathcal{PAR}} lp \qquad \mathtt{cd} \xrightarrow{\mathcal{PU}} pu}{< \mathbf{Def\ Node}\ \mathtt{id}\ (\mathtt{ps})\ \mathtt{cd}\ \mathbf{End},\ e > \xrightarrow{\mathcal{DN}} e'}$$

$$\forall \mathtt{p} \in pu.\ \exists \mathtt{i} \in \{1..\ \#lp\}.\ lp(\mathtt{i}) = \mathtt{p}$$

$$\forall \mathtt{i}, \mathtt{j} \in \{1..\ \#lp\}.\ \mathtt{i} \neq \mathtt{j} \Rightarrow lp(\mathtt{i}) \neq lp(\mathtt{j})$$

$$e' = e[(\mathtt{cd},\ lp)/\mathtt{id}]$$

A avaliação do comando **Use** resulta em um nó de nome $\mathtt{id}_2$ cujo conteúdo é o resultado da avaliação do corpo da definição $\mathtt{id}_1$ do ambiente e com os argumentos fornecidos. $\mathcal{ARG}$ avalia a lista de argumentos. $\mathcal{EXPANSOR}$ resulta no corpo sintático de $\mathtt{id}_1$ já expandido com os argumentos e $\mathcal{CONT}$ avalia o conteúdo de um nó.

$$\frac{\mathtt{as} \xrightarrow{\mathcal{ARG}} la \qquad < e(\mathtt{id}_1),\ la > \xrightarrow{\mathcal{EXPANSOR}} \mathtt{con} \qquad \mathtt{con} \xrightarrow{\mathcal{CONT}} cont}{< \mathbf{Use}\ \mathtt{id}_1\ (\mathtt{as})\ \mathtt{id}_2,\ e > \xrightarrow{\mathcal{USE}} no}$$

$$no = (\mathtt{id}_2,\ cont)$$

$$\mathtt{id}_1 \in \mathrm{dom}(e)$$

As relações de avaliação de um nó com ou sem rótulo ($\mathcal{NOCR}$ e $\mathcal{NOSR}$, respectivamente) são definidas basicamente em termos da que avalia o conteúdo de um nó ($\mathcal{CONT}$). Esta última, por sua vez, é definida em termos das que avaliam os diferentes tipos de conteúdos.

As relações que avaliam gráficos e códigos são omitidas por questões de espaço e por serem triviais. A relação que avalia um texto ($\mathcal{TEX}$) é definida em termos das que avaliam os diferentes tipos de texto. As palavras são avaliadas pela relação $\mathcal{PALS}$, resultando numa lista.

A avaliação de um comando **But** resulta em um par onde estão as palavras que formam a referência e o nome do nó que referenciam.

$$\frac{\texttt{pals} \xrightarrow{\mathcal{PALS}} lpals}{\textbf{But}\ (\texttt{pals} : \texttt{id}) \xrightarrow{\mathcal{BUT}} (lpals,\ \texttt{id})}$$

A avaliação de um comando **Alt** resulta em um par onde estão as palavras que formam a referência e a tabela de alternativas. Esta tabela consiste de uma lista de pares formados por palavras de referência e o nome de nó referenciado.

$$\frac{\texttt{pals} \xrightarrow{\mathcal{PALS}} lpals \qquad \texttt{alts} \xrightarrow{\mathcal{ALTS}} tab\_alts}{\textbf{Alt}\ \texttt{pals}\ (\texttt{alts}) \xrightarrow{\mathcal{ALT}} (lpals,\ tab\_alts)}$$

As avaliações dos comandos **Exp** e **Foot** são semelhantes.

$$\frac{\texttt{t}_1 \xrightarrow{\mathcal{TEX}} texto_1 \qquad \texttt{t}_2 \xrightarrow{\mathcal{TEX}} texto_2}{\textbf{Exp}\ (\texttt{t}_1)\ (\texttt{t}_2) \xrightarrow{\mathcal{EXP}} (texto_1,\ texto_2)}$$

$$\frac{\texttt{pals} \xrightarrow{\mathcal{PALS}} lpals \qquad \texttt{r} \xrightarrow{\mathcal{ROD}} rod}{\textbf{Foot}\ (\texttt{pals})\ (\texttt{r}) \xrightarrow{\mathcal{FOOT}} (lpals,\ rod)}$$

As variáveis semânticas usadas estão enumeradas a seguir:

$libs \in$ Libs	$arqs \in$ Arquivos	$d, d', d_1 \in$ Doc
$e, e', e_1 \in$ Env	$no \in$ NO	$nos \in$ NO*
$cont \in$ Conteudo	$lp \in$ Id*	$la \in$ (Palavra*)*
$texto_1, texto_2 \in$ Texto	$lpals \in$ Palavras	$tab\_alts \in$ Tabela_alt
$rod \in$ Rodape	$seg \in$ Seguinte	$pu, ir, iu, iu' \in \mathcal{FP}$(Id)

As variáveis sintáticas estão descritas a seguir:

id, $\texttt{id}_1$, $\texttt{id}_2 \in$ Id	c, $\texttt{c}_1$, $\texttt{c}_2 \in$ CORPO	nsr $\in$ NO_S/ROT
ps $\in$ PARAMETROS	cd $\in$ CORPO_DEF	cn $\in$ NO
as $\in$ ARGUMENTOS	$\texttt{t}_1$, $\texttt{t}_2 \in$ TEXTO	pals $\in$ PALAVRAS
alts $\in$ ALTERNATIVAS	con $\in$ CONTEUDO	r $\in$ RODAPE

Uma definição mais completa está disponível com os autores.

Referências

[1] J. Conklin: "A Survey of Hypertext", *MCC TR*, **Nr. STP-356-86**, Rev 1, Austin, TX, Feb 1987.

[2] P. J. Brown: "Hypertext : The Way Forward". Personal Communication. University of Kent, Canterbury, 1987.

[3] L. N. Garrett, K. E. Smith and N. Meyrowitz: "Intermedía: Issues, Strategies and Tactics in the Design of a Hypermedia Document System", IRIS, Brown University.

[4] J. Conklin and C. Richter: "Support for Exploratory Design", *MCC TR*, **Nr. STP-117-85**, Austin, TX, Oct 1985.

[5] J. Fiderio: "A Grand Vision", *Byte*, Oct 1988, pp 237-244.

[6] N. Meyrowitz: "Intermedia: The Architecture and Construction of on Object-Oriented Hypermedia System and Aplications Framework", *ACM*, Sep 1986.

[7] A. Newell, D. L. McCracken, G. Robertson and R. M. Akscyn: "ZOG and the USS Carl Vinson", Carnegie-Mellon University, 1982.

[8] P. J. Brown: "Viewing documents on screen", University of Kent, Canterbury, Kent.

[9] P. J. Brown: "Interactive Documentation", *Software -Practice and Experience*, **Vol. 16**, Mar 1986, **pp 291-299**.

[10] P. J. Brown: "Report on an application of Guide in decision support". Personal Communication. Jun 1987.

[11] P. J. Brown: "Linking and Searching within Hypertext", *Electronic Publishing Origination, Dissemination and design*, **Vol. 1**, Apr 1988.

[12] Tekla S. Perry: "Hypermedia: finally here". IEEE, Nov 1987.

[13] Egidio Astesiano: "Operational Semantics". In *Lectures Notes of the State of the Art Seminar on Formal Description of Programming Concepts*, IFIP TC2 WG 2.2, Petrópolis, Abril 1989.

[14] G. Plotkin: "A structural approach to operational semantics". In *Lectures Notes*, AArhus University, 1981.

[15] Huu Le Van and Elisa Terreni: "A Language to describe formatting directives for SGML documents". Personal Communication. Department of Computer Science University of Milan.