Sistema de Indexação e Projeções de HyperPro

Mariza A. S. Bigonha, José de Siqueira, Roberto da S. Bigonha
AbdelAli Ed-Dbali[1], Pierre Deransart[2]
Fabrício M. Schmidt, Flávia Peligrinelli

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte - MG (Brazil)

Abstract

The purpose of this paper is to present the Index Utility and the Index Based Projection of the HyperPro System. HyperPro is a hypertext tool that offers a way to handle two basic aspects: *Constraint Logic Programming* (CPL) documentation and development and text editing. For text editing it is based on the Thot system. A HyperPro program is a Thot document written in a report style. It is designed for logic programming but it can be adapted to other programming paradigms as well. HyperPro offers navigation and editing facilities, such indexes, document views and projections. It also offers document exportation. Projection is a mechanism for extracting and exporting relevant pieces of code according to specific criteria. Indexes are necessary to find the references and occurences of a relation in a document, i.e., where its predicate definition is found and where a relation is used in other programs or document versions and to translate hypertexts links into paper references.

Keywords: Constraint Programming, HyperPro, Logic Programming, HyperText

[1]University of Orléans(France)
[2]Inria - Institut National de Recherche en Informatique et en Automatique(France)

1 Introdução

A programação literária foi introduzida em 1983 por Donald Knuth com uma ferramenta chamada WEB [14] para documentar programas C e Pascal. Este paradigma estabelece que a documentação e o código de um programa devem estar em um único documento. Ambientes implementados para usar esse paradigma permitem que o usuário digite as partes de um programa em qualquer ordem e extraia a documentação e o código do mesmo arquivo quer seja para gerar um artigo ou para executar um programa e obter um resultado.

Um dos problemas existentes na programação lógica é a falta de boas ferramentas que auxiliem o programador no desenvolvimento de seus programas. Aquelas desenvolvidas para linguagens imperativas ou funcionais não podem ser usadas diretamente em programação lógica devido a certas peculiaridades deste paradigma, como por exemplo, o seu aspecto declarativo. Considere por exemplo a linguagem Prolog [12], ela já atingiu sua maturidade, contudo ainda não existe um ambiente que facilite a implementação e a documentação de seus programas. Muito embora a programação literária possa ser considerada como uma alternativa para o desenvolvimento de uma metodologia de documentação de programas, ela ainda não é suficiente para aplicações em programas neste paradigma. Com o CLP (*Constraint Logic Programming*) [10] o problema ainda é mais grave. Os programas CLP são relativamente pequenos trechos de código, mas normalmente constituem o núcleo de uma aplicação. Devido ao seu alto grau de expressividade eles estão mais próximos da especificação do que de um programa tradicional. Posto desta forma, foi proposto e desenvolvido um ambiente denominado HyperPro [4, 1, 2, 3, 5] (Veja Seção 3) cujo objetivo é documentar programas CLP de acordo com o paradigma de estilo literário. Nele, os programas são vistos como documentos executáveis.

Dentre as principais funcionalidades oferecidas por HyperPro destacam-se os índices e as vistas de diferentes partes do documento associados a diferentes utilidades tais como, teste de programas e verificação sintática de linguagens lógicas, tabela de conteúdo, versões e projeções. Vistas são formas de visualização de partes específicas do programa e são úteis ao programador durante o estágio do desenvolvimento da aplicação. Projeção é uma vista dinâmica do documento com partes selecionadas pelo usuário. Versões de programas são programas que diferem em pelo menos uma cláusula.

Um aspecto importante do HyperPro é que, com poucas alterações, ele pode ser adaptado para outros paradigmas como por exemplo o HyperProC para a linguagem C. HyperPro utiliza-se de Thot [8, 9], portanto todas as facilidades disponíveis neste sistema também o são em HyperPro. Thot é um ambiente desenvolvido para produzir documentos estruturados. É um poderoso editor WYSIWYG[3] que possui entre outras, facilidades de hypertexto.

Este artigo apresenta em destaque a implementação de duas novas funcionalidades do HyperPro, o Sistema de Índices e a Projeção Baseada em Índices. Outras funcionalidades como: vistas, projeções e definição de versões de programas são introduzidas e discutidas. Os leitores interessados nas demais funcionalidades do sistema e em mais detalhes sobre a implementação de Projeções e no sistema HyperPro como um todo podem consultar [4]. Os índices servem para indicar onde uma relação aparece no documento, onde sua definição de predicado é feita e onde a relação é usada nas diferentes versões do documento. Na Projeção Baseada em Índices o usuário pode selecionar qualquer entrada de índice e tem como resultado, em uma vista separada, todas as partes relacionadas no documento. Existem dois tipos de projeções baseadas em índices: (1) a projeção com CRI (*Cross Reference Index*) de predicados e (2) a projeção produzida pelo IV (*Índice de Versões*). Em (1), o usuário escolhe uma definição de predicado do CRI e a Projeção Baseada em Índices mostra o predicado corrente da definição e cada definição na qual ela aparece. Em (2), o usuário precisa chamar o Índice de Versões,

[3] *what you see is what you get*

apresentado na Seção 5.1. Ele contém as versões de cada predicado definido no documento, incluindo o número da página em que eles aparecem no documento HyperPro. Selecionada a versão neste índice, é mostrada automaticamente, em uma vista separada, todos os predicados da versão escolhida.

2 O Sistema THOT

O modelo Thot, desenvolvido no INRIA [8, 9, 11], é baseado no aspecto lógico dos documentos. Ele usa um meta-modelo que permite a descrição de numerosos modelos: artigo, livro, relatório, etc, onde cada um deles descreve uma classe de documento com estruturas muito parecidas. As diferenças entre os documentos Thot estão nos elementos que neles aparecem, nas relações entre esses elementos e nas formas como eles estão ligados.

Documentos são definidos por um esquema de estrutura via linguagem S, e um esquema de apresentação, definido na linguagem P, ambos denominados estruturas genéricas [11]. A estrutura genérica define a organização lógica do programa e forma a base do modelo de documento do Thot. O esquema de estrutura define como um documento é organizado em relação aos elementos que o compõem, tais como parágrafos, títulos, seções, notas, referências cruzadas, etc. Estes elementos são organizados em uma estrutura de árvore e cada um tem uma posição e um papel definido dentro do documento.

O esquema de apresentação define, via linguagem P, como estes elementos são apresentados, por exemplo, define a fonte a ser usada em cada parte, seu tamanho, sua posição no texto, sua cor, etc. Uma característica deste esquema é que ele pode definir diferentes vistas para um mesmo documento. Estas vistas funcionam como um filtro que apresenta apenas a parte da estrutura do documento que se quer visualizar em dado momento.Por exemplo apenas os comentários, fórmulas, ou programas ao invés do documento todo.

Thot pode produzir documentos na forma abstrata em alto nível denominada forma canônica. Esta forma se adapta bem às manipulações do editor, mas não se adapta necessariamente a outras operações que possam ser aplicadas aos documentos. Por isso, o editor Thot oferece também a opção de salvar documentos em outros formatos definidos pelo usuário. Neste caso, para cada documento, um conjunto de regras de tradução pode ser definida, especificando como a forma canônica deve ser transformada. Estas regras são agrupadas nos Esquemas de Tradução escritos em uma linguagem T [11]. Associado ao esquema de estrutura de um modelo pode haver diferentes esquemas de tradução, cada um definindo regras para um formalismo diferente. Por exemplo, pode ser usado para exportar documentos para os formatadores TEX, LATEX, Scribe e troff ou para traduzir documentos para SGML e HTML.

No Thot, a geração de uma aplicação baseia-se em Esquemas de Aplicação e são escritos em uma linguagem chamada A (*Generation Application Language*) [11], que é usada para a definição da interface gráfica e criação de menus que podem ser associados às funções padrão do sistema ou a novas funções específicas. Uma aplicação é formada por comandos e ações. Os comandos são executados ao se escolher um item de menu e as ações são executadas quando os eventos aos quais elas estão associadas ocorrem. A ligação deste esquema com o esquema de estrutura se dá de duas formas, ou ele possui o mesmo nome deste com a extensão .a, ou então, é o esquema principal com o nome Editor.a, onde estão definidos os menus e comandos específicos associados.

Thot possui também um conjunto de ferramentas composto de duas bibliotecas: a biblioteca do núcleo do Thot e a biblioteca do editor Thot. A primeira permite que a aplicação lide, de forma automática, com a estrutura lógica e o conteúdo do documento, como: criar novos documentos, modificar documentos existentes, extrair informações de documentos, mostrar partes de documentos, gerenciar mensagens e caixas de diálogo, menus e formulários, etc. A segunda contém todas as

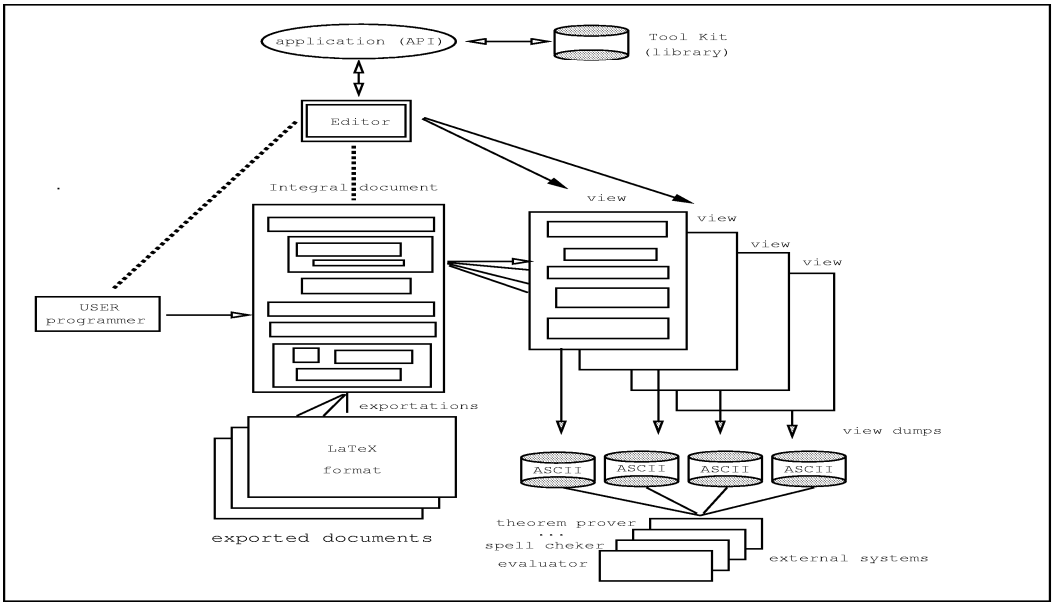

Figure 1: Arquitetura do HyperPro

facilidades incluídas na primeira mais as funções que mostram o aspecto gráfico do documento. O conjunto de ferramentas contêm cerca de 200 funções colocadas em grupos, onde cada grupo enfatiza um aspecto diferente do documento. Elas funcionam nos ambientes UNIX/X Windows e são acessadas via API(*Aplication Programming Interface*) [6].

3 O Sistema HyperPro

HyperPro [1, 2, 3, 4, 5, 7], desenvolvido pelos autores, utiliza-se do editor de texto Thot e de sua API, (veja Figura 1). O usuário lhe fornece as entradas que são programas definindo a estrutura genérica do documento nas linguagens S e P. O documento completo é visualizado em uma só vista denominada vista integral. Além desta, outras cinco vistas foram definidas. HyperPro atualmente define três esquemas de exportação, para LaTeX, ascii e HTML. A estrutura de um documento é composta por: um título, pelo menos uma seção, uma tabela de conteúdo e um índice de referência cruzada. Ele pode conter, opcionalmente, nome de autores e suas afiliações, palavras-chave, data, referências bibliográficas, anexos e índices de versões.

Como o HyperPro é um sistema desenvolvido para auxiliar os programadores a depurar os programas no paradigma lógico, ele permite que os seus documentos contenham diversos programas e suas versões. Dado que estes programas e versões são escritos em CLP, eles são constituídos de relações. Portanto, em um documento neste sistema, um parágrafo pode ser uma definição de relação. Uma definição de relação é composta por um título e uma lista de pelo menos uma definição de predicado corrente (c.p.d.). O título é um indicador de predicado, nome do predicado e sua aridade, ou um nome. A c.p.d. pode ser apontada por uma referência hypertexto do Thot, chamada referência de predicado corrente (c.p.r.). Toda relação possui também uma barra de versões que é preenchida com os nomes das versões das quais ela participa. Uma definição de predicado é composta de quatro itens: comentários informais, que são seqüências de parágrafos; asserções e tipos, que são seqüências de linhas de texto opcionais e um conjunto de cláusulas, que podem ser cláusulas Prolog ou CLP. As clásulas

incluem diretivas, *goals*, fatos e regras.

3.1 Definição de Versões de Programas

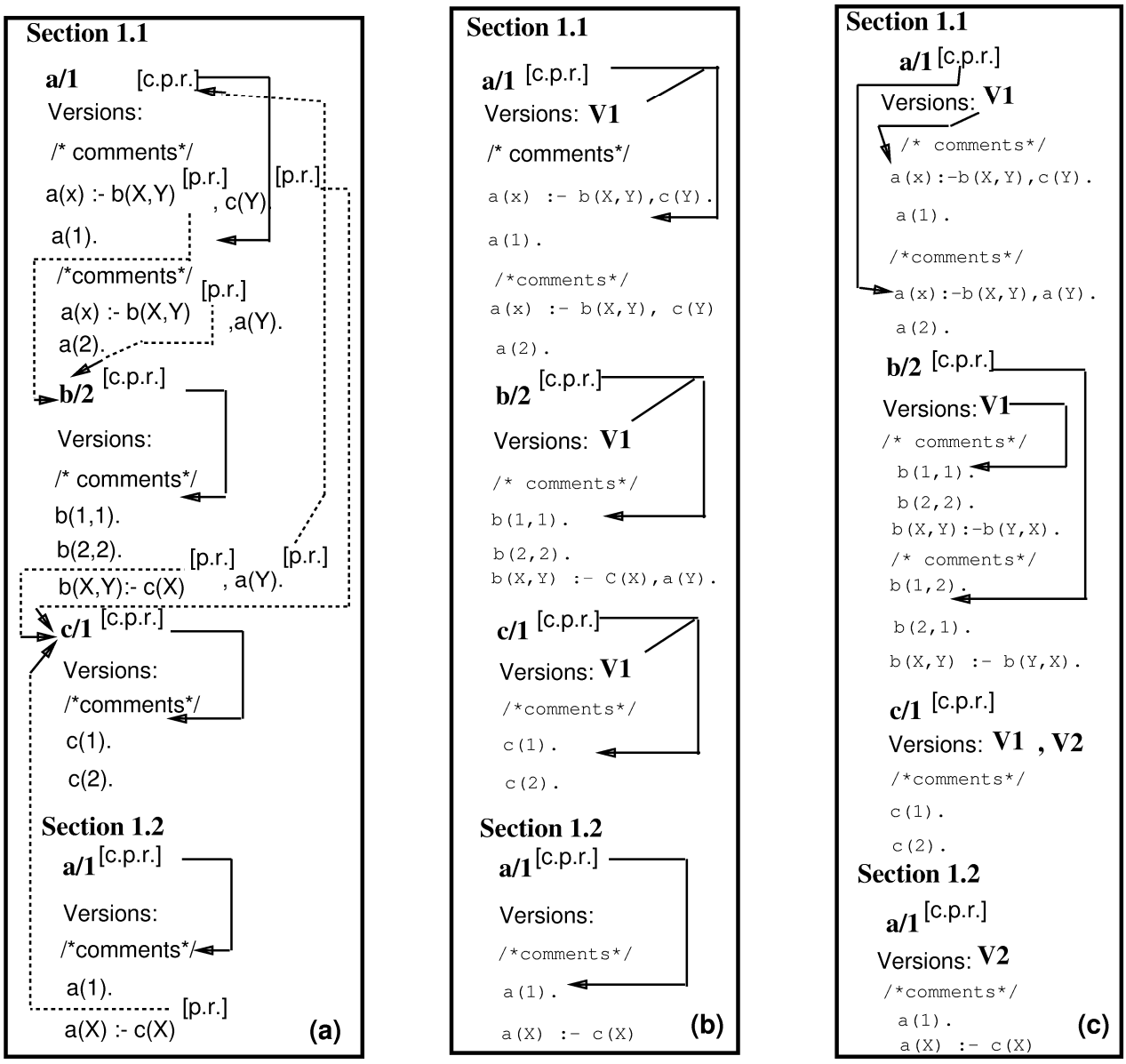

Figure 2: (a) Algumas Relações e suas Referências de Predicado Corrente(b) Depois de Nomear a Versão `V1` (c) Depois de Nomear a Versão `V2`

O fato de poder definir, testar e documentar diferentes versões de um programa, especialmente ao desenvolver e documentar programas CLP, fez com que fosse incluída em HyperPro uma funcionalidade que atendesse esta necessidade. A definição de versões de programas permite especificar para cada relação, distintas versões de suas definições de predicados. Associado ao fato dos predicados no corpo da cláusula poderem ou não ter referências às relações que os definem é possível definir a versão atual

de um programa.

O usuário pode definir c.p.r da forma que lhe aprouver e, em seguida, nomear a versão que será constituída pelas relações cujos c.p.d. são apontados pelas c.p.r previamente definidas pelo usuário. Assim, toda vez que o usuário modificar a c.p.r. de uma relação para apontar para outra definição de predicado, uma nova versão deve ser definida. Para definir uma nova versão seleciona-se a primeira relação que compõe a versão e, em seguida, seleciona-se o comando para nomear a versão e digita-se o nome da mesma. Automaticamente, todas as relações participantes da versão terão a sua barra de versões acrescidas do nome da nova versão. Este nome é uma referência hypertexto para o c.p.d. da relação, apontado por seu c.p.r. no momento em que o comando para se nomear a versão foi disparado.

Figura 2(a) exibe um conjunto de relações com as suas c.p.r. apontando para as suas respectivas c.p.d. Olhando a figura, note que na primeira relação optou-se para que a c.p.r. apontasse para a primeira definição de predicado, ao invés da segunda. Figura 2(b) exibe a modificação obtida da Figura 2(a) após o usuário ter selecionado a primeira relação `a/1` e nomeado a versão `V1`. Note que a barra de versões de todas as relações que participam da cadeia de chamadas iniciada em `a/1` recebem o nome da versão, `V1`, que é uma referência hypertexto para a c.p.d. apontada pela c.p.r. da relação.

Veja que, quando uma relação é selecionada para nomear uma versão, a sua definição de predicado naturalmente conterá predicações que fazem referências a outras relações presentes no documento, cujas definições de predicado também irão conter predicações que fazem referências a outras relações no documento e assim recursivamente. Esta cadeia de referências iniciada com a relação escolhida para nomear a versão é que irá definir quais relações irão fazer parte da versão, e que, conseqüentemente, terão a sua barra de versões acrescidas do nome da nova versão nomeada. A forma como esta cadeia pode ser definida e assim percorrida também é tarefa do usuário. Ele deve colocar referências hypertextos em todas as predicações no corpo de cada definição de predicado que apontem para a relação que as define. Estas referências são chamadas referências de programas (p.r.). A definição das p.r. é importante, pois especifica a relação que define a predicação sem que haja dualidade. De fato, podem haver duas relações com o mesmo nome no documento, como é o caso da relação `a/1` da Figura 2(a). Note-se que predicações dentro da c.p.d. da relação que as define, como no caso de chamadas recursivas ao predicado, não necessitam de p.r. Na Figura 2(b), as p.r. não são mostradas para não sobrecarregar demais a figura. A Figura 2(c) mostra a situação da Figura 2(b) após a relação `a/1` na `Seção 1.2` ter sido selecionada e a versão nomeada como `V2`. Logo após, a relação `c/1` tem sua barra de versões acrescida de `V2`.

3.2 Vistas

O documento pode ser visto por meio de várias perspectivas chamadas vistas, que são especificadas na apresentação genérica. O documento todo é visualizado em uma única vista, a vista integral.

As demais vistas, *Comment_View* mostra apenas os comentários relativos às definições de predicado. *Assertion_View* apresenta exclusivamente as partes de asserções relativas às definições de predicado. *Program_View* apresenta somente as partes de cláusulas e definição de predicados. *Typing_View* visualiza apenas os tipos relacionados às definições de predicado. *Table_of_contents* permite ver a tabela de conteúdos. Esta vista aparece automaticamente com a abertura do documento. As vistas podem ser abertas simultaneamente e são automaticamente sincronizadas. Um aspecto importante em relação a esta funcionalidade é a possibilidade de editar em uma vista específica ao invés de escrever na vista integral, e a informação aparecer em todas as vistas abertas.

4 Projeções

Em um ambiente integrado de programação e documentação, é importante que o usuário tenha uma forma de testar seu programa separado do resto do documento. Esta é a idéia da projeção. Projeção é uma vista do documento com partes selecionadas pelo usuário. Elas diferem de vistas pré-definidas no processo de seleção. Vistas são estáticas e já estão incorporadas no Thot, as projeções são dinâmicas e devem ser implementadas no HyperPro. HyperPro oferece as projeções: manual, recursiva, de versões, automática (*por expressão regular*) e a projeção baseada em índices (veja Seção 4.1). A descrição mais detalhada do funcionamento e uso das projeções pode ser encontrado em [4].

Na projeção manual escolhe-se manualmente várias partes do documento, chamadas elementos, e esses elementos são mostrados em uma vista separada. É possível incluir e excluir qualquer elemento nessa vista. Na projeção recursiva, seleciona-se uma definição de relação, e o HyperPro exibe todos os pacotes de cláusulas relacionados àquela definição incluindo o predicado corrente e todos os predicados referenciados na corrente a qual a relação está inserida. Esta projeção é útil para detectar qual predicado tem sua referência definida ou para descobrir o lugar onde ela foi definida. Na projeção automática o usuário fornece uma expressão regular e o HyperPro exibe todas as porções do documento, escolhidas pelo usuário, em que a expressão aparece. A menor granularidade para essa projeção é uma palavra. Na projeção de versão, seleciona-se uma versão a ser vista separadamente. Esta vista conterá todos os pacotes de cláusulas que compõe a versão escolhida.

4.1 Projeção Baseada em Índices

Na Projeção Baseada em Índices seleciona-se qualquer entrada do índice e mostra em uma vista separada todas as partes do documento relacionadas aos elementos pertencentes à entrada selecionada do índice.

Para criar esta projeção foi necessário introduzir atributos de visibilidade no esquema de apresentação e de estrutura do documento Hyperpro. Para que um elemento qualquer apareça em uma vista é necessário que seja definida a visibilidade do elemento no esquema de apresentação. As regras de apresentação do elemento devem definir uma visibilidade maior que a sensibilidade definida para a vista em que será exibido o documento. Portanto, no esquema de estrutura foram incluídos dois atributos de visibilidade, um para os elementos a serem exibidos na Projeção Baseada em Índices de Versões e outro para os elementos a serem exibidos na Projeção Baseada no Índice de Referências Cruzada. Além disto foram incluídas em seu esquema de apresentação as declarações das vistas correspondentes ao Índice de Versões e ao Índice de Referências Cruzada. A visibilidade do HyperPro foi definida com um valor mínimo para estas vistas, zero, e, finalmente, a sensibilidade de cada vista foi definida para conter um valor expressivo, no caso oito.

A função para criar as projeções baseadas no Índice de Versões e no Índice de Referências Cruzada realiza as operações: exibe um cursor para escolher o elemento para o qual será exibida a projeção e o retorna; o ancestral do elemento escolhido é acessado e se ele for do tipo *lista de versões* a projeção baseada no Índice de Versões é executada. Se for do tipo *lista de relações*, a projeção baseada no Índice de Referência Cruzada é executada. Veja a vista da Projeção Baseada em Índices na Figura 3.

5 Índices

Para construir automaticamente os índices, optou-se por definir novos elementos correspondentes aos índices e implementar os esquemas de estrutura e de apresentação para cada elemento índice. Estes elementos foram adicionados ao esquema de estrutura e apresentação dos documentos HyperPro. Esta

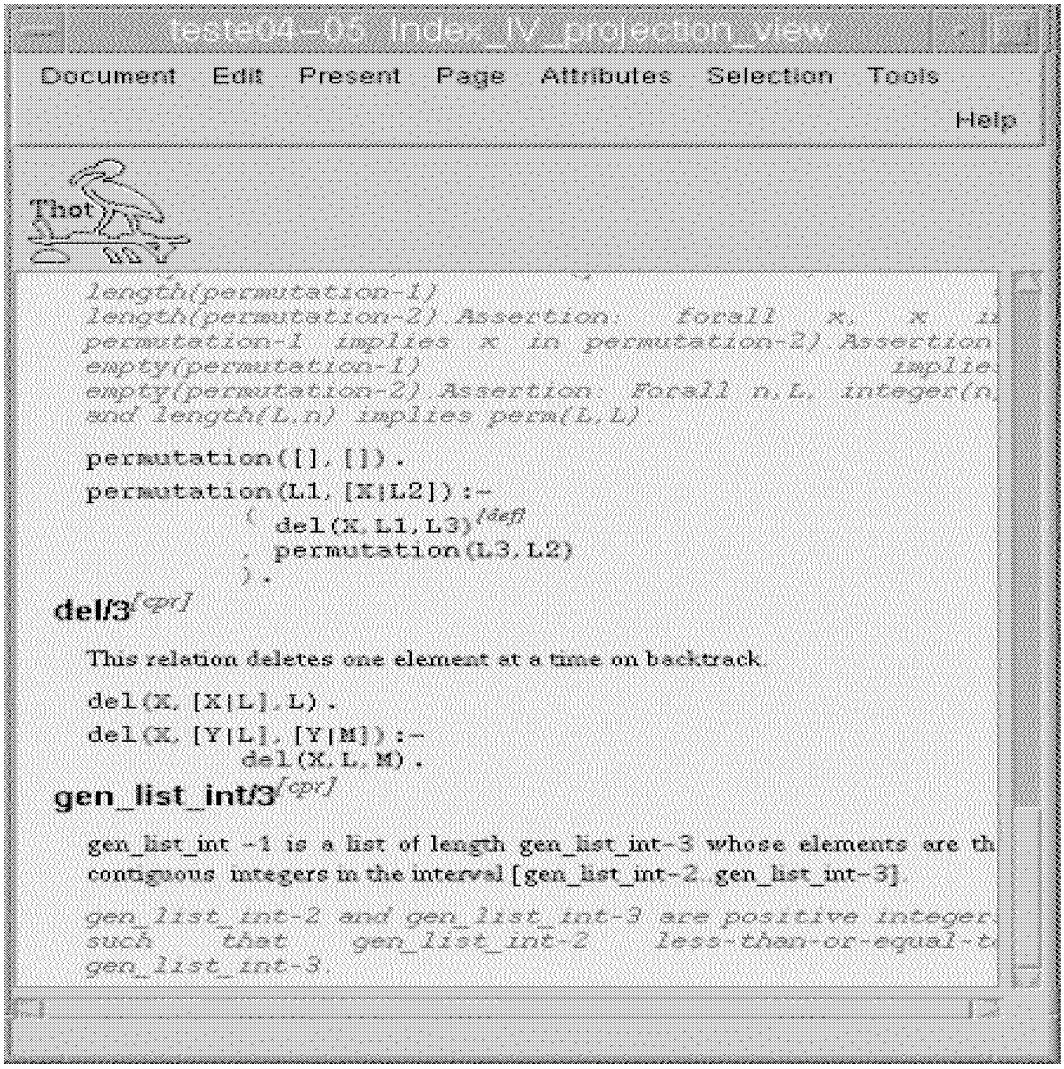

Figure 3: Vista com a Imagem da Projeção do Índice de Versões

forma foi escolhida porque a estrutura dos índices não era condizente com a estrutura do documento para o qual ele estava sendo construído. Se as estruturas fossem condizentes, os índices poderiam ter sido criados como vistas do documento, via seu esquema de apresentação.

Os índices foram construídos via as aplicações que manipulam os elementos definidos para os índices e o documento para o qual os mesmos serão construídos. Para implementar as aplicações que irão criar os índices automaticamente foram utilizadas a linguagem A e o conjunto de ferramentas de API, ambos do Thot. A linguagem A permite aplicações baseadas no conceito de documento ativo. Um documento ativo é um documento eletrônico que se transforma ou atua no seu próprio ambiente computacional, quando certos comandos de edição são requisitados pelo usuário.

Associado aos itens do menu relativos à criação dos Índices de Versões (IV) e de Referência Cruzada (IRC) foram implementadas em C, duas funções. Estas funções usam várias funções do conjunto de ferramentas da API, e constróem o IV quando o usuário seleciona o item de menu relativo à criação do Índice de Versões ou o IRC quando o usuário seleciona o item de menu relativo a criação do Índice de Referência Cruzada. Estas funções percorrem a estrutura do documento HyperPro, para o qual será

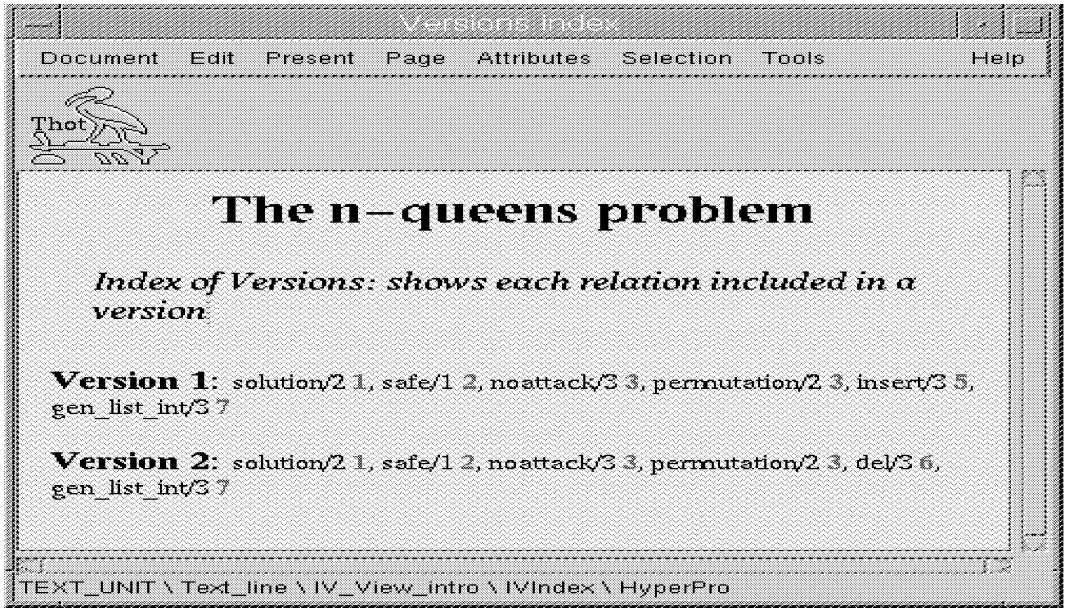

Figure 4: Vista com a Imagem do Índice de Versões

construído o IV ou o IRC, acessando e armazenando os elementos necessários para a construção dos mesmos. Para armazenar e recuperar estes elementos, foi criado um arquivo de inclusão em C. Neste arquivo se encontram: a estrutura de dados que armazena estes elementos, as funções de inserção que posiciona estes elementos na estrutura e as funções de recuperação que obterão os elementos no segundo passo do algoritmo.

5.1 Índices de Versões

O algoritmo para construir o Índice de Versões é feito em dois passos. No primeiro passo é acessada cada relação que participa do documento e, para cada uma dessas relações, são acessadas as versões das quais elas participam e o c.p.d. que a relação assume em cada uma dessas versões. As versões são então organizadas em uma lista e a relação, nome/aridade e c.p.d., é armazenada nas listas correspondentes às versões das quais ela participa. Esta estrutura é uma lista encadeada de listas encadeadas.

No segundo passo, informações coletadas no primeiro passo são então usadas para a construção do elemento IV. Um elemento IV é criado, e automaticamente para cada versão os elementos correspondentes as relações participantes da versão são criados e fixados com os nomes das relações, assim como as referências aos c.p.d. são criadas e fixadas para os c.p.d. correspondentes. Estas referências são mostradas com o número da página em que aparece o c.p.d referenciado. Clicando-se neste número o c.p.d referenciado é então exibido no documento HyperPro. Para exibir as informações no IV, foi criado um elemento cujo esquema de estrutura e de apresentação atende às necessidades deste índice. Veja a imagem com o resultado da execução do IV na Figura 4.

5.2 Índice de Referência Cruzada

O algoritmo para criar o Índice de Referência Cruzada é dividido em dois passos. No primeiro passo é acessada cada relação que participa do documento e, para cada uma dessas relações, são acessados os elementos correspondentes ao seu título, ao seu c.p.d e às chamadas que a relação possa ter

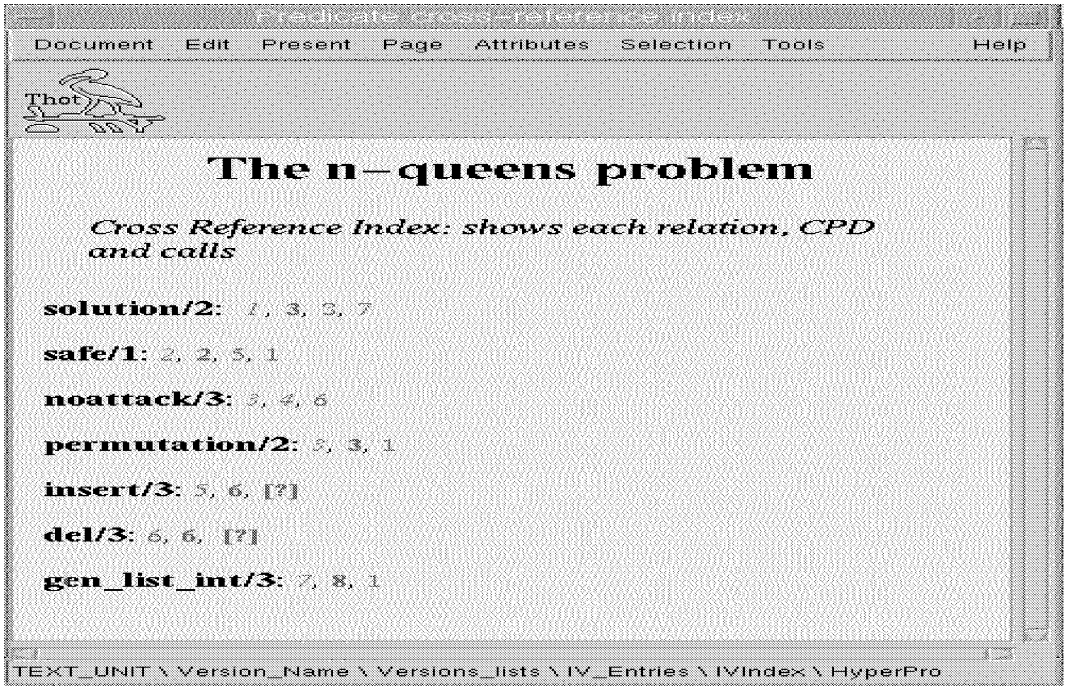

Figure 5: Vista com a Imagem do Índice de Referências Cruzadas

no documento. As relações são então organizadas em uma lista e o seu título c.p.d e chamadas são armazenadas na sublista da relação correspondente. Esta estrutura é uma lista encadeada de listas encadeadas. Para exibir as informações no IRC, foi criado um elemento cujo esquema de estrutura e de apresentação atende às necessidades deste índice. No segundo passo para a construção do IRC, as informações coletadas no primeiro passo são utilizadas para a construção do documento correspondente ao IRC, definido pelos esquemas citados anteriormente. Um elemento IRC é criado e automaticamente são criados os elementos correspondentes a cada uma das relações, que correspondem às entradas do índice. Para cada relação, as referências correspondentes ao título, c.p.d e chamadas à relação são criadas e definidas para os elementos correspondentes no documento HyperPro, que foram coletados no passo anterior. Estas referências aparecem com o número da página em que aparecem os elementos referenciados. Clicando-se neste número, o elemento referenciado é então exibido no documento HyperPro. A Figura 5 mostra a vista com o resultado da execução do Índice de Referência Cruzada.

Note que pode existir duas relações com mesmo nome e aridade, mas que participem de versões diferentes, e por isso possuem c.p.d. distintas. Este fato não afeta o índice, a relação terá duas entradas distintas no mesmo, uma para cada versão. Isto é possível pois cada relação possui um identificador próprio para o elemento no documento HyperPro que a representa.

6 HyperPro × Outros Sistemas de Programação Literária

Documentação de programas ainda não é um problema satisfatóriamente resolvido, principalmente na área de programação em lógica. Em meados dos anos 80, o método de programação literária se difunde e vários programas são escritos com a introdução de WEB para Pascal [13] por Donald Knuth. Pouco depois, WEB é adapatado para outras linguagens como C, Ada, Modula2, Fortran, etc [17, 14]. Porém, com certo tempo de uso, a complexidade de WEB cria uma barreira entre o programador e

esse estilo de programação, e, a programação literária se torna útil somente para aqueles que podem construir suas próprias ferramentas para simplificar o WEB. Surge então NOWEB [15] que provê as utilidades do WEB mas menos complexo. O NOWEB foi desenvolvido no ambiente UNIX e é portável a outras plataformas desde que elas possam simular *pipelines* e suportar ANSI C e AWK ou Icon. NOWEB não depende de nenhuma linguagem de programação, o que o torna mais simples, porém menos poderoso, enquanto o WEB tem certas características de dependência de linguagem tais como: *prettyprinting*, *typesetting* comentários usando TEX, expansão de macros, avaliações de expressões constantes e conversão de *strings* literais para índices em uma *string pool*. O NOWEB usa TEX e o LATEX, o LATEX é usado para emitir o índice e as informações de referência para informar o número de página. O WEB trabalha mal com o LATEX.

Ferramentas mais novas, como o FUNNELWEB e NUWEB [15] são, ao contrário de sua predecessora, WEB, independentes de linguagem mas continuam complexas. O projeto inicial do NUWEB foi baseado no NOWEB, mas apesar de muito similares, possuem algumas diferenças de sintaxe. O NUWEB usa *pipeline* e é um programa C. Sua estrutura o torna portátil pois somente é necessário um compilador C. Também é mais rápido porque não há partes interpretadas e a sobrecarga da criação de *pipelines* é eliminada, mas ele é difícil de ser extendido. O FUNNELWEB é uma ferramenta complexa que inclui seus próprios *typesettings* de linguagem e comandos *shell*.

Além desses sistemas, citamos Cweb [16] e Spider [17]. Cweb é uma ferramenta para produzir documentação de programas em uma combinação com a linguagem C e *troff*, uma linguagem para formatação de textos. Cweb difere de WEB na escolha da linguagem usada. Spider foi projetada para desenvolver programas em ADA. Spider é um gerador WEB. Usando Spider o usuário pode construir um WEB sem precisar entender osd detalhes de uma implementação WEB.

Mais importante é a verossemelhança das ferramentas estudadas: uma simples entrada produz um programa compilável e um documento publicável. Todas satisfazem essas expectativas, mas nenhuma passou do estágio de experimentação ou ou foi usado com programas de grande porte.

Nosso propósito ao projetar e implementar HyperPro foi oferecer um sistema que permita registrar toda a experiência acumulada durante o desenvolvimento de uma aplicação baseada em programação lógica com restrições e mantê-la: programação, depuração, testes, etc. O alto nível de expressividade da programação lógica com restrições torna possível considerar um programa como uma especificação executável. Por outro lado, ela possui a flexibilidade de um documento textual. Todas as informações referentes ao programa desenvolvido e sua manutenção são registradas em um único documento.

7 Conclusão

HyperPro é um sistema de documentação para programação em lógica com restrições. Ele foi desenvolvido dentro do programa de cooperação internacional entre o *Institut National de Recherche en Informatique e Automatique* (INRIA) e o Departamento de Ciência da Computação da UFMG. Suas principais funções são os índices, as vistas de diferentes partes do documento associados a diferentes utilidades tais como, teste de programas e verificação sintática de linguagens lógicas, tabela de conteúdo, versões e projeções. O Sistema de Índices e a Projeção Baseada em Índice, apresentados neste artigo, completam o ambiente HyperPro. HyperPro possui também funções para exportar e importar documentos de vários formatos, por exemplo, LATEX, ascii e html.

References

[1] Siqueira, José de, Schmidt, Fabrício, *Manual do Usuário para Índices e Projeções Baseadas em Índices para o HyperPro Básico*, UFMG, RT DCC 013/1999.

[2] Siqueira, José de, Schmidt, Fabrício, *Manual de Sistema para Índices e Projeções Baseadas em Índices para o HyperPro Básico*, RT DCC 012/1999, UFMG.

[3] Peligrinelli, Flavia, Bigonha, Mariza, *HyperPro: Sistema de Programa e Documentação em um Ambiente de Programação Baseado no Paradigma Literário*, Technical report of the Programming Language Laboratory, DCC- UFMG, LLP003/1999, 03/1999.

[4] Bigonha, Mariza A.S., Ed-Dbali, AbdelAli, Bigonha, Roberto S., Peligrinelli, Flavia, Deransart, Pierre, Siqueira, J. de, *Projection of HyperPro Document*, III ISBLP, 1999, pages:171-183.

[5] Pierre Deransart AbdelAli Ed-Dbali Mariza A. S. Bigonha Roberto S. Bigonha Jose de Siqueira, "Basic HyperPro Functionalities and Utilities", *RT 023/97, DCC-UFMG*, 12/1997.

[6] Quint, V. and Vatton, I., *The Thot Kit API*, INRIA Rocquencourt, technical report, 07/10/1997.

[7] Deransart, P., Bigonha, R., Parot, P., Bigonha, M., Siqueira, J. de, *A Hypertext Based Environment to Write Literate Logic Programs*, I SBLP, 1996, pages:1-16.

[8] Quint, V. and Vatton, I., *Grif: an interactive System for structured Document Manipulation*, Proceedings of the International Conference on Text Processing and document Manipulation, 1986, November, 200-213, Cambridge University Press.

[9] Quint, Vicent & Vatton, Iréne, *Hypertext Aspects of the GRIF Structured Editor: Design and Applications*, Rappports de Recherche # 1734, Julliet 1992.

[10] M. Bergère and G. Ferrand et alii, *La Programmation Logique avec Contraintes Revisitée en Termes d'Arbre de Preuve et de Squelettes*, Orléans, 1995, LIFO 96-06.

[11] Quint, V., *The Languages of Thot*, Internal report, INRIA-CNRS, translated by Ethan Munson, version of 06/25/1996.

[12] Deransart, Pierre and Ed-Dbali, Abdelali and Cervoni, Laurent, Springer Verlag, *Prolog, The Standard; Reference Manual*, 1996.

[13] Knuth, Donald, *The Web System of Structured Documentation*, Technical Report 980, Stanford Computer Science, California, 09/1983.

[14] Knuth, Donald, *Literate Programming*, CSLI lecture notes, Stanford, CA, Center for the study of language and information, 1992, 27, 349–358.

[15] Ramsey Norman, *The noweb Hacker's Guide*, CSD, Princeton University, 09/1992 (Revised 08/1994).

[16] Thimbleby, H., *Experiences of 'Literate Programming' using Cweb(a variant of Knuth's Web)*, The Computer Journal, Vol. 29, No. 3, 201-211, 1986.

[17] Ramsey Norman, *Literate Programming: Weaving a language-independent Web* , Communications of the ACM, 32(9): 1051-1055, 09/1989.