HyperPro: Um Ambiente para Programação em Lógica no Estilo Literário

Mariza Andrade da Silva Bigonha
Roberto da Silva Bigonha
AbdelAli Ed-Dbali [1]
Flávia Peligrinelli Ribeiro
Pierre Deransart [2]
José de Siqueira

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Belo Horizonte - Brasil

Abstract

HyperPro is a tool that offers a way to handle two basic aspects: *Constraint Logic Programming* (CPL) and text editing. For text editing it uses the Thot system. A HyperPro program is a Thot document written in a report style. It was design for logic programming but it can be adaptable to other languages as well. HyperPro offers navigation and editing facilities, such index, document views and projections. It also offers document exportation.

Keywords: HyperPro, Constraint Logic Programming, Projections.

[1] University of Orléans, France
[2] Institut National de Recherche en Informatique et en Automatique–France

1 Introdução

O `HyperPro` [3, 4, 5, 6, 7, 1] é um ambiente experimental idealizado para desenvolver programação em lógica baseado no paradigma do estilo literário [14, 16, 17]. O paradigma da programação literária estabelece que a documentação e o código de um programa estejam em um único documento. As ferramentas implementadas 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.

Apesar de a linguagem Prolog [13] já ter atingido sua maturidade, ainda não existe um ambiente que facilite a implementação e a documentação de seus programas. Neste sentido, a programação literária surge como uma alternativa para o desenvolvimento de uma metodologia de documentação de programas no paradigma lógico.

O objetivo de HyperPro é documentar programas CLP [20] (*Constraint Logic Programming*) oferecendo ao usuário a possibilidade de editar diferentes versões e programas, comentários e informações de verificação formal, assim como a possibilidade de executar, depurar e testar os programas. Os programas são vistos como documentos executáveis. Um aspecto importante desse sistema é que, com poucas alterações, o `HyperPro` pode ser adaptado para outras linguagens. O sistema utiliza-se da ferramenta `Thot` [8, 9].

O `Thot` é um sistema desenvolvido para produzir documentos estruturados. É um poderoso editor WYSIWYG (*what you see is what you get*) que possui entre outras, facilidades de hypertexto. Ele permite que o usuário crie, modifique e consulte interativamente os documentos, além de permitir a produção de documentos homogêneos. O usuário pode, então, se concentrar na organização e no conteúdo dos seus documentos.

Com o intuito de prover as facilidades descritas acima foram desenvolvidos no sistema `HyperPro` padrões de estruturas de documentos para programas de acordo com a metodologia de programação em lógica. Esses padrões utilizam-se das linguagens do `Thot`. São elas: a linguagem S [10]: responsável pelas estruturas genéricas que formam a base do documento; a linguagem P [10]: responsável pela apresentação do documento; a linguagem T [10]: responsável pela exportação ou tradução de documentos da forma canônica do `Thot` para outros formalismos, por exemplo: HTML [22], LaTeX[21], etc; a API [11]: responsável pelos esquemas de interface e a linguagem A [12]: responsável pela geração de aplicações.

2 Sistemas de Programação Literária

Em vez de considerar o papel do programador como sendo o de definir instruções para o computador, passaremos a nos concentrar em explicar para as pessoas o que os programas de computadores fazem. Um programa deve ser escrito não só para as máquinas lerem, mas também para os seres humanos entenderem. Essa é a principal idéia de programação literária.

A programação literária foi introduzida em 1983 por Donald Knuth por meio de uma ferramenta chamada WEB [14, 15]. A partir daí, falar sobre programação literária significava falar sobre WEB. Este sistema permite que o código fonte coexista com o texto descritivo em um único documento e que as partes de um programa sejam rearranjadas em qualquer ordem auxiliando na explicação das funções dos programas.

Em meados dos anos 80, o método de programação literária se difunde e vários programas são escritos. WEB é adapatado para outras linguagens como C, Ada, Modula2, Fortran, etc [19]. Porém, com certo tempo de uso, muitos programadores ficaram insatisfeitos devido a complexidade do WEB. Essa complexidade acabou por criar uma barreira entre o

programador e esse estilo de programação. Dessa forma, a programação literária se tornou útil somente para aqueles que podiam construir suas próprias ferramentas para simplificar o WEB.

Surge então NOWEB [17, 18] que provê as utilidades do WEB mas menos complexo. O NOWEB foi desenvolvido no ambiente UNIX e é transportável para outras plataformas desde que essas possam simular *pipelines* e suportar ANSI C e AWK ou Icon.

O NOWEB e o NUWEB [16], apesar de muito similares, possuem algumas diferenças de sintaxe. O FUNNELWEB [16] é uma ferramenta complexa que inclui seus próprios *typesettings* de linguagem e comandos *shell*.

O projeto inicial do NUWEB foi baseado no NOWEB. 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 estendido.

As ferramentas WEB podem ser usadas para diferentes linguagens de programação, mas estas devem ser lexicamente similares a liguagem C. NOWEB não depende de nenhuma linguagem de programação, o que o torna mais simples, porém menos poderoso. 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 WEB trabalha mal com o LATEX. O NOWEB trabalha com ambos, TEXe LATEX. Contudo, o ponto mais importante dessas ferramentas é a verossimilhança: uma simples entrada produz um programa compilável e um documento publicável. Ambos satisfazem essas expectativas.

3 THOT e suas Linguagens

O modelo `Thot` é baseado no aspecto lógico dos documentos, ou seja, ele permite que o usuário trabalhe com certas entidades que ele tem em mente quando faz um documento. E são essencialmente essas entidades lógicas, por exemplo, parágrafos, seções, capítulos, notas, títulos e referências cruzadas, que fornecem ao documento sua estrutura lógica. Mas as diferenças entre documentos não são apenas as entidades que neles aparecem, mas também as relações entre essas entidades e as formas como elas estão ligadas.

O sistema `Thot` usa um meta-modelo que permite a descrição de numerosos modelos onde cada um descreve uma classe de documentos que, por sua vez, possuem estruturas muito parecidas. Cada documento possui uma estrutura específica que organiza suas partes. A estrutura genérica define as formas pelas quais uma estrutura específica pode ser construída, ou seja, descreve a organização lógica do documento.

Para programar no `Thot` existe disponível no sistema quatro linguagens: S, P, T e A.

3.1 A Linguagem S

Estruturas genéricas formam a base do modelo de documento do `Thot`. Cada estrutura genérica lógica, que define as classes de documentos e objetos, é especificada por um "programa" escrito em uma linguagem S [10], e chamado `Esquema de Estrutura`. A gramática de S, assim como as gramáticas das linguagens P e T [10], são descritas usando a meta-linguagem M, derivada de Bakus-Naur Form (BNF).

Cada `Esquema de Estrutura` começa com a palavra-chave STRUCTURE e termina com END. A palavra-chave STRUCTURE pode ser seguida pela palavra-chave EXTENSION, no

caso do esquema definir uma extensão, seguido do nome da estrutura que o esquema define e de um ponto e vírgula.

Em um esquema completo, a definição do nome da estrutura é seguido pelas declarações de *default*, atribuições globais, parâmetros, regras de estrutura, elementos associados, unidades, elementos de esqueleto e exceções.

As seções ATTR, STRUCT, ASSOC E UNITS definem novos atributos, e fazem com que novos elementos, novos elementos associados e novas unidades adicionem suas definições ao esquema principal.

Os elementos cuja posição na estrutura do documento não é fixa são chamados elementos associados. Eles são definidos como estruturas em que o conteúdo pode ser organizado logicamente pelos construtores de elementos primitivos ou construídos. Pode acontecer que os elementos associados estejam totalmente desconectados do documento, como por exemplo um comentário. Elementos associados introduzem um novo uso para o construtor *reference*. Ele serve não somente para criar elos entre os elementos da estrutura principal do documento, mas também liga os elementos associados a estrutura principal.

O modelo é suficientemente flexível para levar em consideração todas as fases da vida útil de um documento. Por exemplo, se uma estrutura genérica específica deve conter um título, uma citação, uma introdução e pelo menos dois capítulos e uma conclusão, isso significa que o documento deverá conter todos esses elementos. O Exemplo 1 mostra um trecho do `Esquema de Estrutura` que define o HyperPro usando a linguagem S.

3.2 A Linguagem P

Devido ao modelo adotado pelo `Thot`, a apresentação do documento está separada da sua estrutura e conteúdo. O conceito de apresentação engloba o chamado *layout* de página, composição e modelo do documento. O esquema de apresentação define o conjunto de operações que exibe o documento na tela ou o imprime. A apresentação do documento é definida genericamente por uma linguagem chamada P. Um programa feito nessa linguagem, é chamado `Esquema de Apresentação`.

O `Esquema de Apresentação` começa com a palavra-chave PRESENTATION e termina com END. A palavra-chave PRESENTATION é seguida pelo nome da estrutura genérica na qual a apresentação será aplicada. Este deve ser o mesmo nome usado no `Esquema de Estrutura`. O Exemplo 2 define a `Estrutura de Apresentação` para o sistema HyperPro utilizando a linguagem P do `Thot`.

O elo entre a estrutura e a apresentação é claro: a organização lógica do documento é usada para compor a apresentação, já que a proposta da apresentação é mostrar a organização do documento.

A descrição da apresentação também define a apresentação genérica, já que ela descreve a aparência da classe de documentos ou objetos.

Para preservar a homogeneidade entre os documentos, a apresentação é descrita como um simples conjunto de ferramentas simples que dá suporte ao *layout* de grandes documentos tão bem quanto a composição de objetos, como figuras gráficas ou fórmulas matemáticas.

Para assegurar a homogeneidade das ferramentas para documentos tanto quanto para os objetos que eles contenham, toda apresentação no `Thot` é baseada na noção de caixa, como em TEX. Uma caixa está associada a uma *string* de caractere, linha do texto, página, parágrafo, título, fórmula matemática ou tabela de células.

3.3 A Linguagem T

O `Thot` pode produzir documentos de forma abstrata em alto nível chamada de 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 esse motivo, o editor `Thot` oferece a escolha de salvar documentos na forma canônica ou em um formato definido pelo usuário. Neste caso, o documento `Thot` é transformado pelo programa de tradução. Essa facilidade pode também ser usada para exportar documentos do `Thot` para outros sistemas usando outros formalismos. Por exemplo, a tradução pode ser usada para exportar documentos para formatadores básicos como Tex, Latex [21], Scribe e troff. Também pode ser usado para traduzir documentos para SGML e HTML [22].

Para cada documento ou classe do objetos, um conjunto de regras de tradução pode ser definido, especificando como a forma canônica deve ser transformada. Estas regras de tradução estão agrupadas nos `Esquemas de Tradução` [10] escritos em uma linguagem chamada T. A mesma estrutura lógica pode ter diferentes `Esquemas de Tradução` cada um definindo regras para um formalismo diferente. Os esquemas de tradução são genéricos.

Um `Esquema de Tradução` começa pela palavra-chave TRANSLATION, seguida do nome da estrutura genérica para qual está sendo definida e um ponto e vírgula, e termina pela palavra-chave END. O nome da estrutura genérica deve ser o mesmo da `Esquema de Estrutura`. O Exemplo 3 apresenta um trecho do Esquema de Tradução do sistema HyperPro para HTML.

3.4 A Linguagem A

No sistema `Thot`, a geração da aplicação baseia-se em `esquemas de aplicação` que são escritos em uma linguagem chamada A [12] (*Generation Application Language*). Esta linguagem é usada para a definição da interface gráfica e para a criação de menus os quais podem ser associados às funções padrões 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. Ambos estão especificados nas seções DEFAULT, ELEMENTS e ATTRIBUTES do `esquema de aplicação`. Os comandos de edição do `Thot` geram dois tipos de eventos cujos nomes diferem pela extensão:

`.Pre`: quando o comando padrão é chamado pelo usuário antes de ser processado pelo editor e retorna um booleano;

`.Post`: quando a ação é executada depois que o editor executou o comando padrão e não possuem valor de retorno.

Um `esquema de aplicação` se relaciona a um `esquema de estrutura` e 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.

Um `esquema de aplicação` começa com a palavra-chave APPLICATION seguida da palavra EDITOR ou do nome do `esquema de estrutura` relacionado e termina com a palavra-chave END.

Um `esquema de aplicação` pode ser composto de uma ou mais seções do tipo:

DEFAULT: possui associação de eventos e ações e em geral se aplica a todos os tipos de elementos e a todos os atributos definidos no `esquema de estrutura` correspondente.

ELEMENTS: contem ações que são chamadas por um elemento.

ATTRIBUTES: define ações que são chamadas por um dado atributo.

Pelo menos uma dessas seções deve estar presente nos `esquemas de aplicação` associados. O `esquema de aplicação` principal, `EDITOR.a`, possui ainda duas outras seções que não aparecem nos outros esquemas: O Exemplo 4 mostra um trecho do `esquema de aplicação EDITOR.a` do protótipo do `HyperPro Básico`.

USES: essa seção possui os nomes dos outros `esquemas de aplicação` bem como a lista com todos os módulos necessários à aplicação. Essa seção é opcional.

MENUS: última seção do `EDITOR.a` que define os menus da barra de menus. Para cada item do menu, o `esquema de aplicação` associa um comando específico exceto para os menus padrão.

3.5 A API

O conjunto de feramentas do `Thot` é um conjunto de funções C que lidam com documentos estruturados no ambiente UNIX/ X Windows. Geralmente uma aplicação usa essas funções para, entre outras tarefas, criar documentos novos, modificar documentos existentes, extrair informações de documentos e mostrar partes de documentos.

O conjunto de ferramentas do `Thot` possui cerca de 200 funções colocadas em grupos onde cada um enfatiza um aspecto diferente do documento, todas baseadas no modelo `Thot` de documentos. Os grupos são:

Grupo de Aplicação: possui funções de gerência da API.

Grupo de Interface: permite que a aplicação estenda ou modifique o `Thot`

Grupo de Mensagens: permite a aplicação gerenciar mensagens e caixas de diálogo.

Grupode Diálogo: com estas funções a aplicação é capaz de gerenciar menus e formulários.

Grupo de Documentos: essas funções gerenciam os esquemas e todas as ações relacionadas ao documento em si.

Grupo Árvore: essas funções lidam com operações relacionadas a árvore de estrutura que representa a organização lógica do documento.

Grupo de Conteúdo: possui funções que manipulam as folhas da árvore de estrutura.

Grupo de Atributos: contém funções que lidam com os atributos dos elementos.

Grupo de Referência: funções que manipulam as referências do documento, *links* de hypertexto.

Grupo de Linguagem: contém funções que gerenciam o idioma usado em um texto.

Grupo de Apresentação: lida com a apresentação específica do elemento.

Grupo de Vistas: possui funções que manipulam as várias vistas do documento.

Grupo de Seleção: funções que manipulam a seleção de elementos.

Essas funções são acessadas por meio da API(*Aplication Programming Interface*) [11]. O conjunto de ferramentas do `Thot` é composto de duas bibliotecas: a biblioteca do *kernel* 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. A segunda contém todas as facilidades incluídas na primeira e ainda acrescenta funções que mostram o aspecto gráfico do documento.

4 HyperPro

Um documento HyperPro [1, 2, 3] é basicamente um documento `Thot`. Ele possui um título, pelo menos uma seção, uma tabela de conteúdo e um índice de referência cruzada. Opcionalmente, ele pode conter a data, o nome dos autores e suas afiliações, palavras-chave, referências bibliográficas, anexos e índices de versões.

Em um documento HyperPro, um parágrafo também pode ser uma definição de relação. Uma definição de relação é definida por um título e uma lista de pelo menos uma definição de predicado. O título é um indicador de predicado, nome do predicado e sua aridade, ou um nome.

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(FD)`.

Clásulas incluem diretivas, *goals*, fatos e regras. Os predicados no corpo da clásula podem ou não ter referências a relações que os definem. Este fato permite definir a versão atual de um programa no documento `HyperPro Básico`.

4.1 Arquitetura do Protótipo

a.eps

Figure 1: Arquitetura do HyperPro

A Figura 1 ilustra como o HyperPro funciona. Ele utiliza-se de um editor de texto, o `Thot`, e sua API. As API's (*Application Program Interface*) do `Thot` permitem desenvolver aplicações específicas que potencialmente podem atuar na edição de um documento. O kit de ferramentas do `Thot` é um conjunto de funções de edição, escritas em C, que pode ser usado na construção das API's; tais funções executam operações em ambientes estruturados Unix X-Window.

O usuário entra com programas escritos na linguagem S e na linguagem P que definem a estrutura genérica do documento. O documento todo é visualizado em uma só vista integral do documento. Além desta vista, mais quatro vistas podem ser definidas. HyperPro permite definir diferentes esquemas de exportação de documentos na forma canônica para outros tipos de formalismos, por exemplo, LaTex, ascii, etc.

4.2 Funcionalidades do HyperPro

As principais funções do HyperPro são 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.

A interface de HyperPro aparece no menu `Tools` do `Thot`. Isso acontece porque o `Thot` permite que qualquer usuário inclua suas próprias facilidades usando sua caixa de ferramentas, acessível por meio do menu `Tools`. O menu `Tools` oferece as seguintes utilitidades: `Make indexes, Versions, Projections, Tests, Sintax verification` e `HyperPro preferences`. O menu `Projections` oferece ao usuário acesso às seguintes projeções: `Manual, By relation, By regular expression, By version` e `By index based`.

As vistas são selecionadas no menu `Document` do `Thot`. Lá o usuário pode escolher o sub-menu `Open a view` e então, escolhe a vista desejada, inclusive as projeções.

4.2.1 Vistas

O documento pode ser visto por meio de várias perspectivas chamadas vistas, que são especificadas na apresentação genérica. Essas vistas são formas de visualização de partes específicas do programa que são úteis ao programador durante o estágio do desenvolvimento da aplicação, por exemplo, a parte dos comentários.

O documento todo é visualizado em uma única vista, denominada vista integral do documento. As vistas podem ser abertas simultaneamente e são automaticamente sincronizadas. Em vez de escrever na vista integral, o usuário pode editar em uma vista específica, e a informação aparece nas demais vistas abertas. Quatro tipos de vistas foram especificadas: comment_view, assertion_view, typing_view e program_view além da tabela de conteúdos. As Figuras 2, 3, 5, 6, 6 mostram as vistas dos comentários, asserções, tipos, tabela de conteúdos e programa, respectivamente.

Comment_View : a vista dos comentários ilustrada na Figura 2 permite ao usuário ver apenas os comentários relativos às definições de predicado.

Assertion_View : a vista das asserções mostrada na Figura 3 permite ao usuário visualizar exclusivamente as partes de asserções relativas às definições de predicado.

Program_View : Vista do programa permite o usuário ver somente as partes de *clauses* e definição de predicados. A vista está sendo mostrada na Figura 4

Typing_View : a vista de tipos ilustrada na Figura 5 permite ao usuário visualizar apenas os tipos relacionados às definições de predicado.

Table_of_contents : a vista do programa apresentada na Figura 6 permite que o usuário veja a tabela de conteúdos.

4.2.2 Versões

Um programa é um conjunto de pacotes de cláusulas. Um documento pode conter diferentes programas. O usuário decide como o documento estará organizado, definindo seus programas por meio das seções. O usuário pode definir seus programas selecionando convenientemente no documento suas definições de predicado e colocando referências.

O HyperPro permite que o usuário teste manualmente e automaticamente seu programa.

O usuário pode definir para qualquer relação de definição, diferentes versões de uma definição de predicado que é documentada e gerenciada com as mesmas utilidades usadas para definir programas. De fato, versões de programas são programas que diferem em pelo menos uma cláusula.

4.2.3 Índices

O HyperPro possui o seguinte tipo de índice: índice de referência cruzada, que indica onde a relação aparece no documento, onde a sua definição de predicado é encontrada e onde a relação é usada em outros programas ou versões no documento.

4.2.4 Conversões e Exportações

O HyperPro é um sistema aberto, o que significa que ele pode produzir documentos para outros sistemas através de um mecanismo de exportação/importação.

O HyperPro produz documentos em um nível abstrato chamado de forma canônica. Pode ser definida uma série de regras de tradução de documentos. Na versão atual, foram definidos dois esquemas diferentes de tradução: exportação de documentos para LaTeXe exportação de documentos para ASCII. Também foram feitos esquemas de exportação de certas partes do documento: exportação do programa, exportação dos comentários informais, exportação das asserções e exportação dos tipos.

Os arquivos de tradução (.T) contém as regras de tradução que convertem o documento do formato PIVOT (.PIV) para os diferentes formalismos.

4.2.5 Projeções

Em um ambiente integrado de programação e documentação, é importante que o usuário tenha uma maneira de testar seu programa separado do resto do documento. Daí surge a idéia da projeção. Uma projeção é uma vista do documento com partes selecionadas pelo usuário.

O processo de seleção depende da projeção desejada. As projeções diferem de vistas pré-definidas no processo de seleção. Além disso, vistas são estáticas e já estão incorporadas no `Thot`, as projeções são dinâmicas e devem ser implementadas no HyperPro. Uma descrição mais detalhada do funcionamento e uso das projeções pode ser encontrado em [1].

O HyperPro oferece cinco projeções diferentes: a projeção manual, a projeção recursiva, a projeção de versões, a projeção automática (*por expressão regular*) e a projeção baseada em índices.

Projeção Manual: a projeção manual é uma vista dinâmica na qual o usuário pode trabalhar com elementos selecionados. Nesta projeção, seleciona-se várias partes do documento, chamadas elementos, e esses elementos são mostrados em uma vista separada (Veja Figura 7). O usuário pode incluir e excluir qualquer elemento nessa vista.

Brevemente, incluir um elemento na projeção é o mesmo que ligá-lo a um atributo, e remover o elemento da projeção é o mesmo que remover o atributo do elemento.

Projeção Recursiva: na projeção recursiva, o usuário pode selecionar uma definição de relação, e o HyperPro mostrará em uma vista separada 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 (Veja Figura 8). A projeção recursiva é útil para ajudar o usuário a detetar qual predicado tem sua referência definida ou descobrir aonde ela está definida.

Projeção Automática: a projeção automática é uma facilidade de vista onde o usuário fornece uma expressão regular e como resultado o HyperPro mostra em uma vista separada todas as porções do documento, escolhidas pelo usuário, em que a expressão aparece. A menor granularidade para essa projeção é uma palavra (Veja Figura 9).

Projeção Baseada em Versões: o usuário pode, na projeção de versão, selecionar uma versão a ser vista separadamente. Esta vista conterá todos os pacotes de cláusulas que compõe a versão escolhida (Veja Figura 10).

Projeção Baseada em Índices: nessa projeção, o usuário poderá selecionar qualquer entrada de índice e terá como resultado todas as partes relacionadas do documento em uma vista separada. Existe dois tipos de projeções baseadas em índices: (1) a projeção com o índice CRI (*Cross Reference Index*) de predicados e (2) a projeção produzida pelo índice de versões. Em (1), o usuário escolhe uma definição de predicado do índice CRI e a Projeção baseada em índices mostra, em uma vista separada, o predicado corrente da definição e cada definição na qual ela aparece. Em (2), o usuário precisa chamar o índice de versões. Esta vista de índice contém as versões de cada predicado definido no documento, incluindo o número da página em que eles aparecem no documento do HyperPro. Selecionada a versão neste índice, o HyperPro mostra automaticamente, em uma vista separada, todos os predicados da versão escolhida.

5 Conclusão

O sistema que apresentamos neste trabalho é a segunda versão da ferramenta `HyperPro` [6]. Este sistema é o resultado 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.

HyperPro é um sistema de documentação para programação em lógica. As principais funções atualmente disponíveis no sistema 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.

HyperPro possui também várias funções para exportar e importar documentos de vários formatos, por exemplo, latex e html.

References

[1] Bigonha, Mariza A. S., AbdelAli, Ed-Dbali, Bigonha, Roberto S., Ribeiro, Flávia P., Deransart, P., Siqueira, José, "Projection of HyperPro Document", *a ser publicado nos Anais do III Simpósio Brasileiro de Linguagens de Programação*, Porto Alegre, 5-7 de maio/1999.

[2] Peligrinelli, Flavia, Bigonha, Mariza A. S., et all, "Um Ambiente para Desenvolver Programação em Lógica Baseado no Paradigma do Estilo Literario", *Trabalho selecionado para o Primeiro Lugar na área de Ciências Exatas e da Terra* durante a Sétima Semana de Iniciação Científica
e *Publicado nos Resumos da Sétima Semana de Iniciação Científica da UFMG*, página 204, 1998.

[3] Deransart, P., Bigonha, Roberto S, Ed-Dbali, AbdelAli, Siqueira, Jose, Bigonha, Mariza A S, *Basic HyperPro Functionalities and Utilities*, Relatório Técnico n^o 23, Departamento de Ciência da Computação, ICEx, UFMG, 1997, 17 páginas.

[4] Deransart, P., Bigonha R.S., Parot, P., Bigonha, M.A.S., Siqueira, J., *A Hypertext Based Environment to Write Literate Logic Programms*, Relatório Técnico n^o 15, Departamento de Ciência da Computação, ICEx, UFMG, 1996.

[5] Parot, Patrick, *The HyperPro Experimental System*, Relatório Técnico n^o 16/96, Departamento de Ciência da Computação, ICEx, UFMG, 1996.

[6] Deransart, P., Bigonha Roberto S., Parot, P., Bigonha, Mariza A.S., Siqueira, J., *A Literate Logic Programming System*, *Anais do I Simpósio Brasileiro de Linguagens de Programação da* SBC, Belo Horizonte, 1996, páginas:1-16.

[7] Deransart, P., Bigonha Roberto S., Parot, P., Bigonha, Mariza A.S., Siqueira, J., *A Hypertext Based Environment to Write Literate Logic Programms*, *Anais do* JICSLP'96, Bonn, Germany, setembro/1996, páginas:247-252.

[8] Quint, V., *The Thot user manual*, 1995

[9] Quint, V. and Vatton, I., *Hypertext aspects of the Grif structured editor : design and applications*, INRIA Rocquencourt, 1992, July.

[10] Quint, V., *The Languages of Thot*, Internal report, INRIA-CNRS, 1992, May.

[11] Quint, V. and Vatton, I., *The Thot Tool Kit API*, July 10, 1997.

[12] Quint, V. and Vatton, I., *The Thot Application Generation Language*, May 7, 1997.

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

[14] Knuth, Donald D., *Literate Programming*, The Computer Journal, Vol. 27, No. 2, 1984, pp. 97-111.

[15] Knuth, Donald, *The* `web` *System of Structured Documentation*, Technical Report 980, Stanford Computer Science, Stanford, California, September 1983.

[16] Ramsey Norman, *Literate-Programming Tools Can be Simple and Extensible*, Departament of Computer Science, Princeton University, November 1993.

[17] Ramsey Norman, *Literate Programming Simplified*, IEEE Software, V.11(5), 97-105, September 1994.

[18] Ramsey Norman, *The noweb Hacker's Guide*, Departament of Computer Science, Princeton University, September 1992 (Revised August 1994).

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

[20] Leler, William, *Constraint Progamming Languages: their specification and generation*

[21] Lamport, Leslie, *LATEX: A Document Preparation System*, Editora Addison-Wesley Pub Co

[22] Venetianer, Tomas, *HTML - Desmistificando a Linguagem da Internet*, Editora mcGraw-Hill Ltda, São Paulo.

file=vistacomentario.ps

Figure 2: Visão de Comentários

file=vistaassercao.ps

Figure 3: Visão de Asserção

file=vistaprograma.ps

Figure 4: Visão de Programas

file=vistatipo.ps

Figure 5: Visão de Tipos

file=tableofcontents.ps

Figure 6: Visão da Tabela de Conteúdos

file=Manual.ps

Figure 7: Visão da Projeção Manual

file=Recursiva.ps

Figure 8: Visão da Projeção Recursiva