Projection of HyperPro Document

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

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Abstract

HyperPro is documentation and development tool for constraint logic programming systems. It integrates logic program elements and texts , so it may be viewed as a system for the development of executable documentation. Projection is a mechanism for extracting and exporting relevant pieces of code or text according to specific criteria.

Keywords: HyperPro, Constraint Logic Programming, Projections.

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

1 Introduction

The HyperPro system is a Thot-based documentation and development tool for Constraint Logic Programming (CLP) systems. It is described in [3]. Thot is a structured editing tool with hypertext editing facilities. For a presentation of the Thot editor functionalities and features, the interested reader can refer to the Thot documentation ([7],[9],[8]).

Basic HyperPro documents are stored internally in a tree structure which depends on each document style defined by the user.

Thot editor offers the possibility to integrate gradually new applications to existing ones, so the HyperPro system [3] evolved in different steps.

In this paper will be defined two newly HyperPro functionalities which are: *projections* and *exportation* introduced in the new HyperPro release of the Thot-based structure, presentation and translation schemas, called *Basic HyperPro* [2], which defines a special style for logic program .

Basic HyperPro aims to document CLP programs giving its users the possibility to edit, in a homogeneous and integrated environment, different versions of programs, comments about them, information for formal verification and debugging purposes, as the possibility to execute, debug and test the programs as well. All the attempts and development history of CLP programs can therefore be integrated and consistently documented within an unique environment gathering together a hypertext editor, different CLP interpreters and syntactical verifiers, as different debugging and verification tools as well.

2 Basic HyperPro Document Structure

The Basic HyperPro document is basically a Thot report document. It must have a title, a sequence of at least one section, a table of contents and a cross-reference index. Optionally, it can contain the date, the authors' names and their affiliations, keywords, bibliographical references, annexes and a versions index.

In Basic HyperPro a paragraph may be also a *relation definition*. At least one relation definition should be present in a Basic HyperPro document.

A *relation definition* is defined by a *relation title*, and a list of at least one *predicate definition*. The *relation title* is a predicate indicator, that is, the predicate name and its arity, or just a name, in which case it denotes a packet of goals or directives, that can be seen as a special case of relations. The relation title also contains a reference to a predicate definition, followed optionally by a list of references to predicate definitions that define different *versions* of a program.

A *predicate definition* is formed by three items : *informal comments*, *assertions* and *packet of clauses*. *Informal comments* are a sequence of paragraphs. *Assertions* are a sequence of lines of text and are optional.

Packets of clauses can be Prolog or `CLP(FD)` clauses [6]. Clauses include directives, goals, facts and rules. The predications in the clauses body may have references or not to the relations which define them. This will actually define the current version of a program in a Basic HyperPro document. At least one predicate definition is obligatory in every relation definition.

When a predicate definition is introduced, the fields for informal comments and packet of clauses are initially empty. The optional fields are presented if the user explicitly indicates so. However, the obligatory fields do not need to be filled in immediately after their insertion in the document.

3 Basic HyperPro Functionalities and Utilities

The main functionalities provided by Basic HyperPro are views of different parts of the document and indexes associated with different utilities, such as *program testing* and *syntactic verification* for some pre-defined CLP and logic programming languages. These functionalities and utilities are defined in terms of programs versions. In this paper we are interested in *projections* and *document exporting facilities*, the description of the other features can be found in [2].

3.1 Program Versions

Program versions is an important issue when defining, documenting and developing programs. Specially when developing and documenting CLP programs, the user needs to define, test and document different *versions* of his programs. For that, he may define for any relation, different versions of its predicate definitions which are documented and managed with the same utilities used to define programs. Therefore, a program may have several different coexisting versions which can be defined, named, viewed and tested as much as any program. Indeed, program versions are programs which differ at least in one clause.

3.2 Defining Program Versions

Whenever a program is defined and the user changes the *current predicate reference* of a relation composing it to point to another predicate definition, the user is defining a (new) program version. This is the simplest way to define a program version. However, to complete the definition the user has to insert *program reference* in the predications at the body of the current predicate definition and then name it. This is done with the naming utility. Note that, although the current predicate reference changed for the relation the user modified, all the named reference which pointed to the previous current predicate definition still define the corresponding programs. It is only a new version that is being defined and named, and all previous programs depending on other predicate definitions for this relation are still properly defined.

3.3 Document Exporting Facilities

HyperPro allows the user to export the whole document in two different formats: ASCII and Latex [5]. The new release of HyperPro, the Basic HyperPro, included HTML in their facility list.

The ASCII exporting facility simply makes a dump of ASCII codes of the document into a file.

The Latex exporting facility dumps a Basic HyperPro document into a file in Latex format. The logical structure of the original document is entirely reflected in the Latex file, except for the hypertext links, for obvious reasons. However, the indexes are mirrored in the Latex document, so that the hypertext version of the original document is faithfully rendered in paper, as much as possible.

The HTML exporting facility translates the original Basic HyperPro document in such way to be viewed by any available web browser, where the original hypertext links appear as such in the HTML version of the document. It is available to the user the possibility to translate their document into two html version, one consisting of a unique file or a html version with multifiles. However, the logical structure of the document division in different

unities such as chapter, sections and subsections is not maintained, and only the unities titles appear in the main HTML document in the case of multifile. Neither is translated in the HTML export of a HyperPro document the table of contents or indexes.

3.4 Indexes

Thot provides an index utility which allows the user to define different indexes. To create an index, the user inserts manually pairs of marks around the word or words he wants to appear in the index. After this is done, for all words he chooses, the user creates the corresponding index through a Thot menu.

Based on this utility, Basic HyperPro provides the user with two different indexes: *Cross-reference index*, and *Index of versions*.

The *cross-reference index* indicates where a relation appears in the document, where is found its current predicate definition and where the relation is used in other parts of the programs defined in the document, independently of program versions. It is an absolute index of relation definitions.

The cross-reference index will be created automatically. This means that pairs of indexes marks should be put automatically around predicates everywhere they appear in the clauses entered. Once this is done, the user can create in an usual way the cross-reference index. This facility is not yet available, it is being implemented.

The *index of versions* shows where versions are firstly defined, i.e., where in the document is found the predicate definition pointed by the first named reference appearing in the document, as all corresponding predicate definitions pointed by named reference for each relation included in the program or version as well. Modifications in the version, i.e., version deletion or additions to it should be mirrored in the index. But this is not automatically done by Basic HyperPro, since index construction in Thot depends entirely on the user to put the pairs of marks around the parts he wants to appear in the index.

3.5 Views of the Document

Thot allows the user to define views of his document, such that, chosen portions of it are presented separately in a view. Views are synchronized in such a way to facilitate selecting, moving and editing consistently and easily in a particular view as much as in the main view. In addition, any operation in a view are reflected in all the others. Also, we provide in Basic HyperPro the possibility to project and view separately some parts of the document selected by the user following some criteria, this is called *projections* [1], [2].

Basic HyperPro provides several different views of the same document: the main view of the whole document; view of the table of contents; view of the comments; view of the assertions; view of the packets.

The main view and the table of contents are displayed automatically, when a document is opened. The *main view* shows all the document as it is defined by the user, and while he defines it. It contains all the subsections, sections and chapters defined, with all the relations definitions and versions references. The *table of contents* displays all the subsections, sections and chapters defined in the document and their references, such as their ordering number and page.

The *views of comments*, of *assertions*, and of *packets* are shown on demand by the user through the appropriate Thot menu bar. Each view displays all and only the comments, or assertions or packets defined in the document.

3.6 Projections View

A projection view displays selected portions of the document in a separate view. The selection process depends on the projection wanted. Projections differ from pre-defined views on the selection process. Also, views are static and are already incorporated in Thot's machinery, the projections are dynamic and are provided by Basic HyperPro.

The Basic HyperPro offers five differents projections views: manual projection, recursive projection, regular expression projection, version projection and index based projection. There are two different indexes based projection view, one of them is produced from the Cross Reference Index (CRI) of predicates and the other one is produced from version index.

In the *manual projection view* the user is free to select any part of the document to be viewed.

In the *recursive projection view*, the user can choose one or more relation definitions and the Basic HyperPro will show in a separated view all the packet of clauses related to that relation including the current predicate and all the predicates referenced in the chain which the relation is inserted.

In the *regular expression projection view,* the user gives some regular expression and Basic HyperPro shows in a separate view all the portions of the document where the regular expression appears.

The *version projection view* shows, in a separate view, all the packets of clauses composing a version of the program chosen by the user in the document.

The *index-based projection view* should allow the user to select any index entry and have all the correlated parts of the document shown in a separate view.

4 Basic HyperPro Facilities and Utilities Interfaces

In this section we present the user's schematic interfaces such as menus and sub-menus for the functionalities and utilities that have been present in Section 3. Some functionalities and utilities appear in specific Thot menus, like the export, views and index facilities. However, the manual and regular expression projection view, the version view, the recursive view and the index-based projection views facilities appear in the Thot's *Tools* menu, as the others utilities as well. The reason is that Thot allows anyone to include his own facilities using the Thot toolbox, which are accessible through the Tools menu.

Views are selected in the Thot menu *Views*. There the user chooses the entry point to the sub-menu *Open a view...* and in there, he chooses the view he wants to open. The views the user can open are, therefore: text view; view of the table of contents; view of the program; view of the comments; view of the assertions; view of the types; manual projection view; manual projection view; recursive projection view; regular expression projection view; index based projection view; version projection view and notes view.

The exporting facilities are accessed by the *Document* menu, and the *Save as...* sub-menu. A window opens where the user can then choose the exporting facility he wants to use, Thot, ASCII, Latex, Html, RTF.

We will not present here the index utility interface. The reader can find about it and how to use this facility in the Thot User Manual ([9]).

4.1 Tools Menu

Thot's tools menu offers the user access to the following utilities:

⟨*Other Thot usual menu entries*⟩	Tools	
	Make indexes	▷
	Versions	▷
	Projections	▷
	Tests	▷
	Sintax verification	▷
	HyperPro preferences	

Manual
By relation (recursively)
By regular expression
By version
By Index based

All utilities interface is done through *communications windows*, where the user controls the utility and where the data input is done, and some utilities may open a specific window to work as their output interface.

In the following sections we describe for each projection its dialog box if it exists.

4.1.1 Manual Projection Dialog Box

To use the manual projection when the user clicks on it, a dialog box like the one presented in Figure 1 appears with the buttons: select, delete, cancel and done, where:
select⇒ selects part of the document to be shown.
delete⇒ deletes an element from the projection view.
cancel⇒ cancels the projection view.
done⇒ closes the dialogue box and the projection view remains.

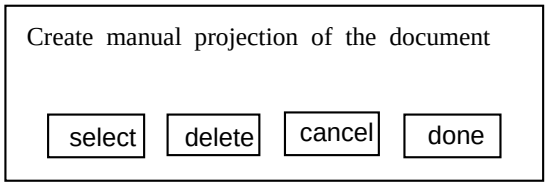

Figure 1: Dialog Box of Manual Projection

To include an element in the projection, the user should select that element with the selection button and automatically it will appear in the `Manual_projection_view`. It is allowed to include several elements in the same view. The user can do that by pressing the selection button again.

To include an element in another view, the system will automatically close the current projection view and open another one with the newly element.

To delete an element, press the delete button and click on the element to be deleted. The element will be deleted but the view remains open because the user may want to keep the

others elements in the view, if they exists.

4.1.2 Recursive Projection Dialog Box

The recursive projection that is available to the user attaches the attributes to the entire relation definition so it doesn't need a dialogue box.

To execute the recursive projection in this current version, the user has to click on the menu `Tools`, `Projections` and then, `By relation (recursively)`. Clicking on that, a small hand appears in the Basic HyperPro document. After that, the user has to choose one relation definition in the document and click on it. The `Recursive_projection_view` will open automatically with the relation definition selected and all the relations which are related to it.

We are working in the new version of this projection to allow the user choosing between the entire relation definition or only in the predicate of the relation.

The dialog box for this new version is presented in Figure 2. In this dialogue box, the button `CPR` will be used to attach the attribute on relation title, and the target of `CPR`. The button `All` will be used to attach the attribute relation definition. Actually `All` represents the current implementation of the recursive projection.

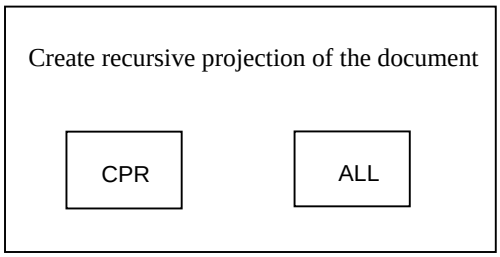

Figure 2: New Dialog Box of Recursive Projection

4.1.3 Regular Expression Projection Dialog Box

To use a regular expression projection it is available a dialog box like the one presented in Figure 4, where the functions are:
Search for⇒ the word to be searched and displayed in the projection.
go⇒ finds the word which is to be searched.
select⇒ include the element which the word is inserted in the projection.
cancel⇒ to cancel the projection.

In this dialog box we choose a word from the Basic HyperPro document typing it in the box bellow `Search for`. It is available four options to direct the user in finding the occurrence of the regular expression in the document: after the word selected, before it, inside a definition selected or in the entire document. By default it will be always selected `after the definition selected`.

4.1.4 Version Projection Dialog Box

The version projection does not need a dialog box. When the user calls this function, a message appears on Basic HyperPro's main window, presented in Figure 4, showing the user

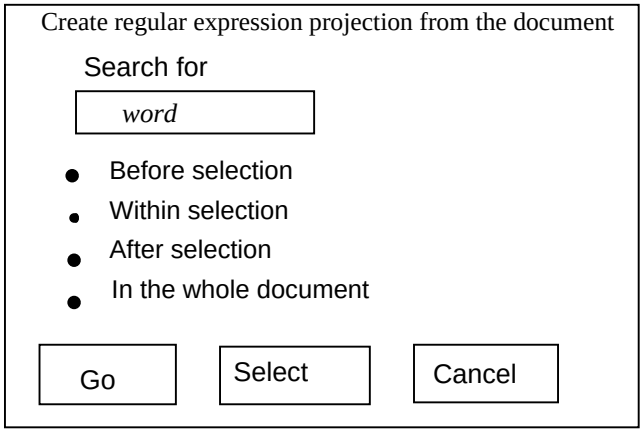

Figure 3: HyperPro Main View

what to do.

To use version projection the user needs to select the menu `Tools`, `Projections`, and `By version`. After that, the user have to click on a version and it will automatically be open in the `Version_projection_view`.

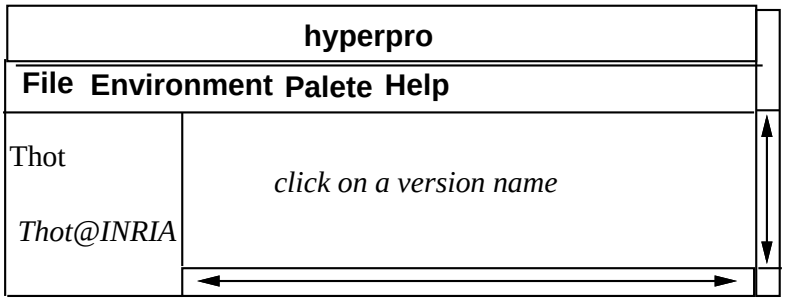

Figure 4: Dialog Box of Version Projection

4.1.5 Index-based Projection Dialog Box

The dialog box for this projection looks like the one showed in Figure 5:

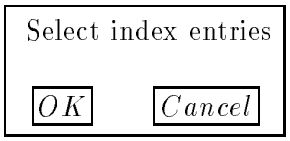

Figure 5: Dialog Box of Index Based Projection

The user clicks on index entries until "OK", then the view is opened.

5 Projections View Implementation

To implement the projections presented in Section 3.6 we had at least two ways to do it. The first one, which is a non trivial solution, consists of adapting a copy of a function from the Thot's Library to our needs and incorporating the modified version in the library.

The second one consists of adding in the Basic HyperPro structure a global variables attribute, and then add in the presentation file of Basic HyperPro a view to have a projection.

The presentation of Basic HyperPro document may be programmed by means of P language [10], and for the view projections of HyperPro, the control of visibilities had to be encoded into the presentation schema as it is described in the sequel. The presentation rule of the visible attribute must have a visibility which must be bigger than the sensibility of Projection view. To make an element visible in the Projection view, it is necessary to attach it to the visible attribute. This solution is easy to implement but presents two drawbacks:

1. it is related with the modification of the structure schema. If the structure and presentation schema is changed, some problems in opening old documents will appear. Besides that, from the user point of view, this is uncomfortable.

2. It is necessary to declare, in the presentation schema, n views for n kinds of projections or any dynamically opened views. Another problem is that we can not have simultaneously two instances of the same kind of view. For instance, we can not have two projection views on different versions.

We have chosen to implement the first option thinking that it would be better because we did not have to make any changes in the structure and presentation schema, which should be frozen by this time. But it did not work because it is hard to understand and modify the *Application Program Interface* (API's) code [4]. So, we have chosen to implement the second option.

In order to implement this option, we made some changes in the structure and presentation schemas. In the structure schema, we included five variables representing the visible attributes, one for each projection presented in Section 3.6.

In the presentation schema we included the declaration of five views in the section where views were declared corresponding to the five projections presented in Section 3.6. Set the Basic HyperPro visibility to zero in those views and, at last, set the attribute with sensibility eight for each view.

After these modifications had been included in the Basic HyperPro schemas, we were able to start implementing the projections.

In the following sections we describe for each projection its functionality and implementation.

5.1 Manual Projection

The manual projection is a dynamic view in which the user can work with selected element. In this projection, the user can select several parts of the document, called elements, and these elements will be shown in a separated view. The granularity of the selected element is defined to be paragraphs and relation definition. Therefore, when the user clicks in any of these elements or in their descendants, the projection will show the whole paragraph or relation definition.

To implement the manual projection we first get the clicked element, next we check if the clicked element is a paragraph or a relation definition traversing the abstract Basic HyperPro

structure tree. Otherwise we verify if the element has already an attribute, if not, we create a new attribute and attach it to the element.

Shortly, to include an element in the projection view is the same as to attach an attribute to this element, and to delete the element from the projection view is the same as to remove the attribute from the element.

The manual projection view for the *n-queens problem* is showed at Figure 8. In this example, the output of manual projection possesses two paragraphs and one relation definition.

5.2 Recursive Projection

In the recursive projection view, the user can choose one or more relation definitions and the Basic HyperPro will show in a separated view all the packet of clauses related to that relation including the current predicate and all the predicates referenced in the chain which the relation is inserted.

The recursive projection view is useful for helping the user to detect which predicates have not yet its predicate reference set or to find out where it is set to.

The recursive projection available to the user attaches the attributes to the entire relation definition so it does not need a dialogue box. This projection works as follow: when the user chooses a relation, the projection finds out all the others relations definitions which is related to it, traversing the Basic HyperPro structure tree, and then attaches the attribute to them.

In the implementation of this projection, it was also used the Thot library to deal with lists. The list contains all the referenced predicates, it starts building the list of relation definitions that must be in the projection, find the definitions references in the packet of clauses and put them into the list. The library's functions used are those: to add an element to a list, to delete an element from the list, and to destroy the list.

In order to complete the implementation of recursive projection the next step is to implement a function to allow the user choosing between selecting the entire relation definition or only the current predicate of the relation. Figure 9, at the end of this document, shows an example of recursive projection view for the *n-queens problem.*

5.3 Regular Expression Projection

The regular expression projection is a view facility where the user can give a regular expression and as a result Basic HyperPro will provide in a separate view all portions of the document where the regular expression appears. The smallest granularity for this projection is a word.

To implement the dialog box for this projection we use one adaptation of the function Search which is in the menu edit of Thot. The function to search the regular expression works as follow: when it finds the string, it attaches the attribute to the element.

5.4 Version Projection

The user can, in the version projection, to select a version to be viewed in a separate view. This view will contain all the packet of clauses composing the version chosen.

The version projection algorithm was implemented in the following way: verify if the user had clicked in a version name or not, if so, from the main root of Basic HyperPro structure tree search for every time the version selected appears, returning the root element of the main abstract tree. After finding the places where the versions appears, find out the places where the reference is pointing to and attach the attribute to that element.

5.5 Index Based Projection

In this projection view, the user will be able to select any index entry and have as a result all the correlated parts of the document shown in a separate view. There will be two kinds of index based projection: the projection produced from Cross Reference Index (CRI) of predicates explained in Section 5.5.1 and the projection produced from version index explained in Section 5.5.2.

5.5.1 Index Based Projection Produced from CRI of Predicates

In this index based projection, the user will choose a predicate definition from the Predicate Cross Reference Index and the *Index Based (CRI) Projection* will show, in a separate view, the current predicate of the definition and every definition that it appears. CRI means *Current Reference Index*.

The Figure 6 presents the skeleton of this projection. Here, `pdu` means *predicate definition and use*. This projection is implemented in the same way as recursive projection. The difference between than is related to the place the user clicks on the predicate. In this case, it clicks on the *predicate cros reference index view* instead of the hyperpro document as in the recursive projection.

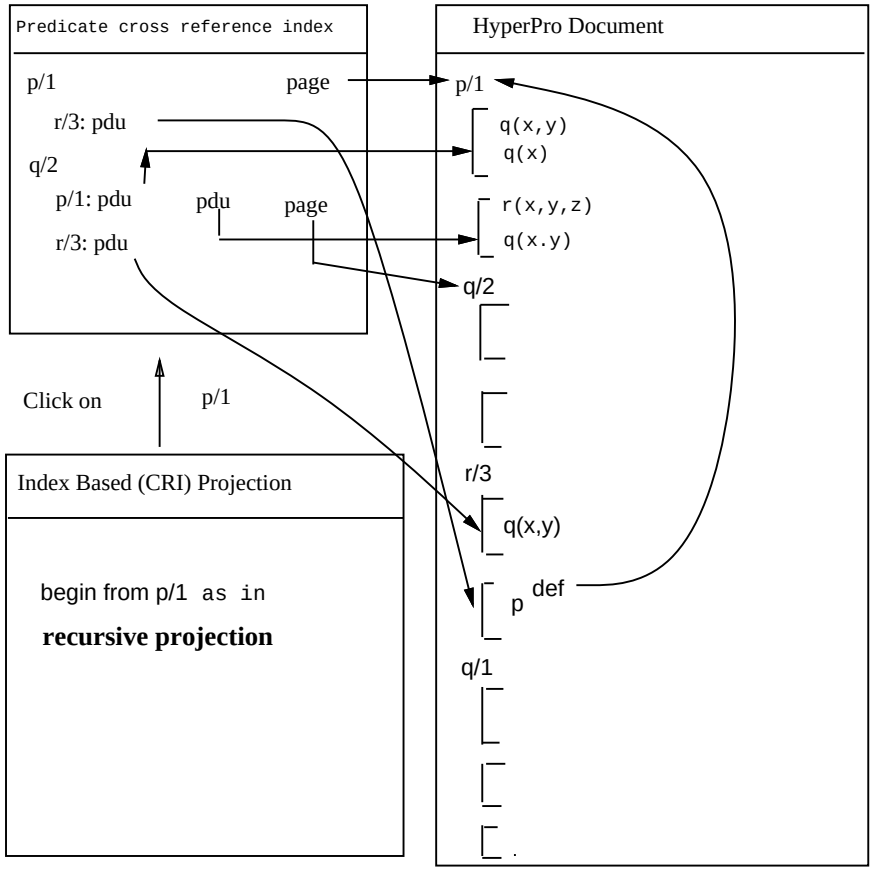

Figure 6: Scheme of Index Projection produced from CRI of Predicates

5.5.2 Index Based Projection Produced from Version Index

For the index based projection the user needs to have the version index. This view contains the versions for each predicate defined in the document, including the page number where they have appeared in the Basic HyperPro document. From this index he will click on a version. After that, Basic HyperPro will provide automatically in a separate view, as showed in Figure 7, the Index_based_projection_VI, all the predicates of the chosen version. VI means *Version Index*. This projection is implemented in the same way as version projection. The difference between than is related to the place the user clicks on the version. In this case, it clicks on the *version index view* instead of the hyperpro document as in the version projection. Figure 7 presents the skeleton of this projection.

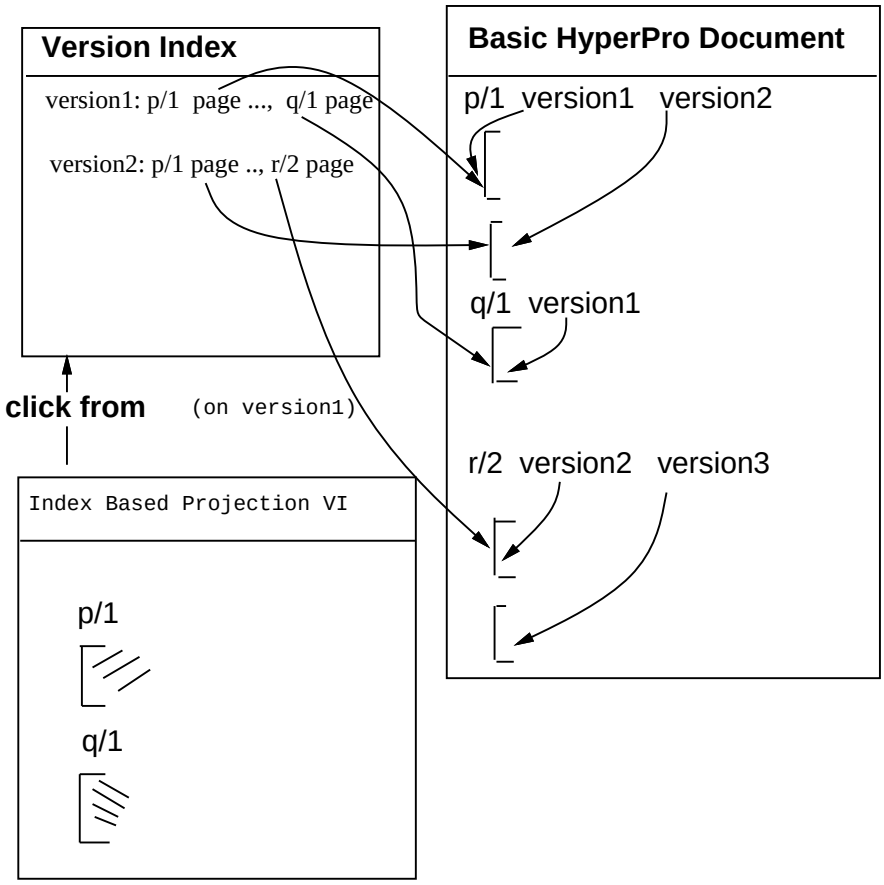

Figure 7: Scheme of Index Projection produced from Version Index

6 Conclusion

HyperPro is a documentation development tool for logic programming systems. It offers several facilities to view and handle documents at different levels of abstraction and from different point of view. Particularly, HyperPro aims to document CLP programs giving its users the possibility to edit, in a homogeneous and integrated environment, different versions of programs, comments about them, information for formal verification and debugging purposes, as the possibility to execute, debug and test the programs as well. All the attempts

and development history of CLP programs can therefore be integrated and consistently documented within a unique environment gathering together a hypertext editor, different CLP interpreters and syntactical verifiers, as different debugging and verification tools as well. It also possesses several functions for exporting the document in different formats such as: html, latex, ascii, and producing projections, which are especial excerpts of the document, according to different criteria, such as, the program goal, pieces of code directly marked by the user, program versions, etc.

In this paper we have presented the implementation results of the projections views which are manual projection, recursive projection, regular expression projection and version projection. The index based projection is under development.

References

[1] 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 7ª Semana de Iniciação Científica. *Publicado nos Anais de Resumos da 7ª Semana de Iniciação Científica da UFMG*, página 204, 1998.

[2] Pierre Deransart AbdelAli Ed-Dbali Mariza A. S. Bigonha Roberto S. Bigonha Jose de Siqueira, "Basic HyperPro Functionalities and Utilities", *Relatório Técnico 023/97, Departamento de Ciência da Computação, UFMG*, dezembro de 1997.

[3] Deransart, P., Bigonha, R., Parot, P., Bigonha, M., Siqueira, J. de, *A Hypertext Based Environment to Write Literate Logic Programs*, I Congresso Brasileiro de Linguagens de Programação, Belo Horizonte, 4 a 6 de setembro de 1996.

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

[5] Lamport, L., *Latex: A Document Preparation System*, Addison-Wesley Publishing Company, 1986.

[6] M. Bergère and G. Ferrand and F. Le Berre and B. Malfon and A. Tessier, *La Programmation Logique avec Contraintes Revisitée en Termes d'Arbre de Preuve et de Squelettes*, LIFO, Orléans, 1995, LIFO 96-06, February.

[7] 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.

[8] Quint, V. and Vatton, I., *Hypertext aspects of the Grif structured editor: design and applications*, Rapports de Recherche #1734, INRIA Rocquencourt, 1992, July.

[9] Quint, V., *The Thot User Manual*, Internal report, INRIA-CNRS, 1995.

[10] Quint, V., *The Languages of Thotl*, Internal report, INRIA-CNRS, translated by Ethan Munson, version of June 25, 1996.

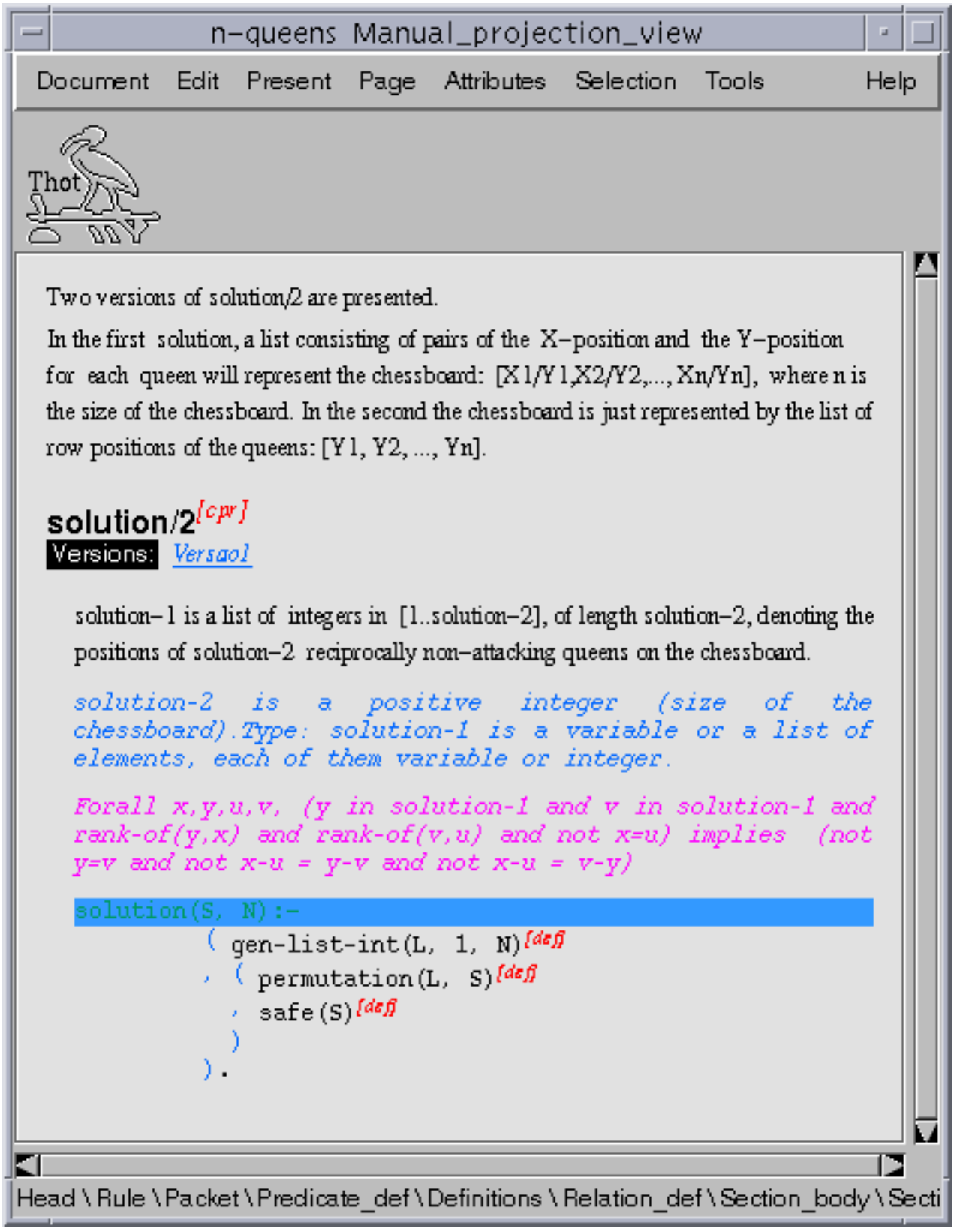

Figure 8: Output of Manual Projection View

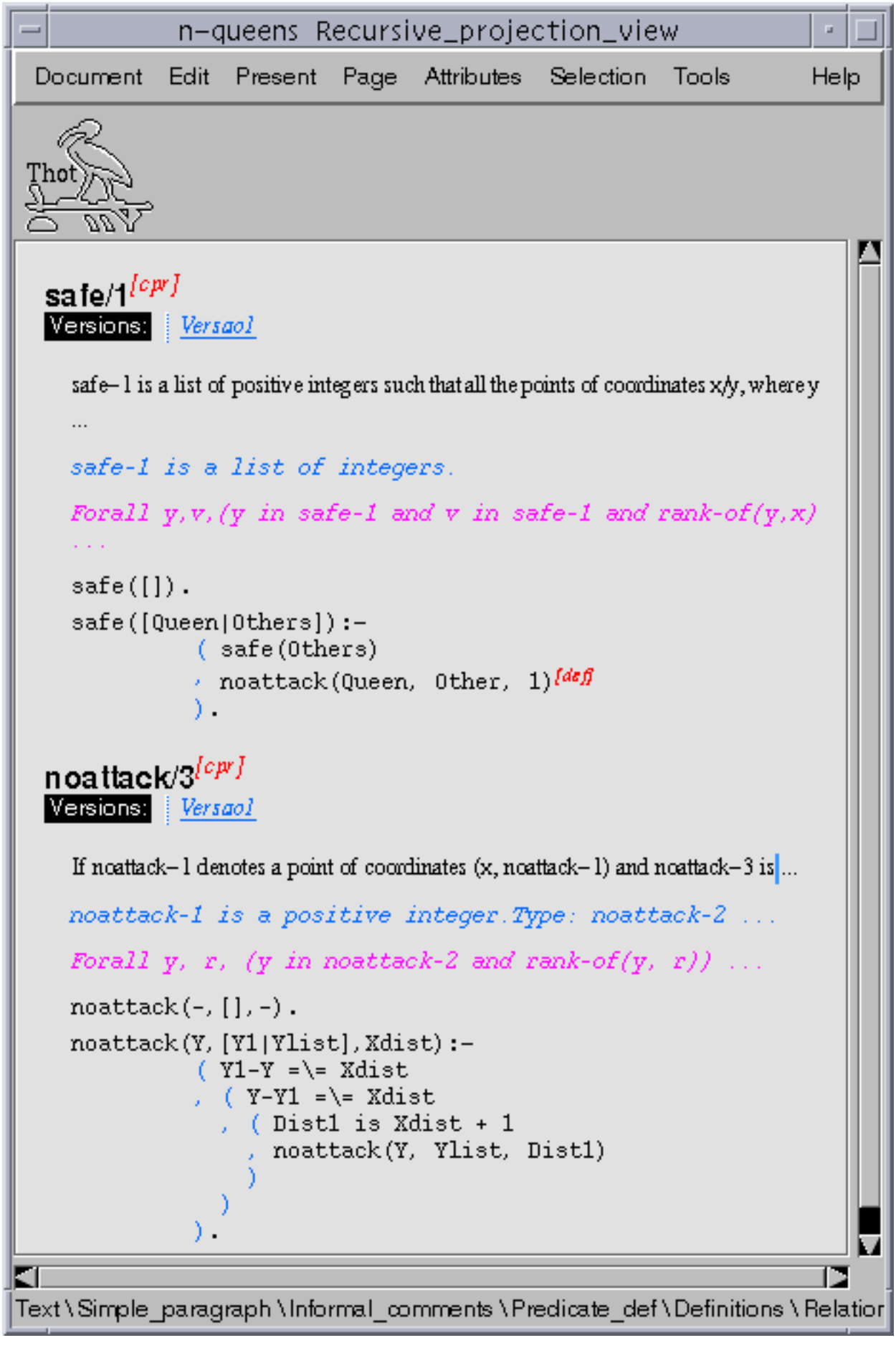

Figure 9: Output of Recursive Projection View