HyperPro

Un environnement intégré de documentation pour la PLC

AbdelAli Ed-Dbali* — **Pierre Deransart**** — **Mariza A. S. Bigonha***** — **José de Siqueira***** — **Roberto da S. Bigonha*****

* *LIFO - Université d'Orléans BP 6759 - 45067 Orléans Cedex - France*
** *INRIA-Rocquencourt BP 105 - 78153 Le Chesnay Cedex - France*
*** *DCC-UFMG Belo Horizonte - MG - Brazil*

AbdelAli.ED-DBALI@lifo.unv-orleans.fr
Pierre.Deransart@inria.fr
{mariza,jose,bigonha}@dcc.ufmg.br

RÉSUMÉ. *Le but de cet article est de présenter certaines fonctionnalités du système* HYPERPRO. HYPERPRO *est un environnement de développement integré pour la PLC (programmation en logique et par contraintes) qui offre deux aspects du développement d'un programme : le codage en PLC et la documentation. Un document* HYPERPRO *est donc une combinaison de code et de sa documentation. La partie édition d'*HYPERPRO *est basé sur le système Thot.* HYPERPRO *est un éditeur WYSIWYG qui offre des facilités de navigation hypertexte, de manipulation de versions d'un même programme, d'extraction de vues et projections selon des critères spécifiques. Il offre également l'exportation de document vers plusieurs formats courants (ps, html, latex, ...) ainsi que la gestion des indexes.* HYPERPRO *a été conçu pour la PLC mais il peut aussi bien être adapté à d'autres paradigmes de programmation.*

ABSTRACT. *The purpose of this paper is to present some functionalities of the* HYPERPRO *System.* HYPERPRO *is a hypertext tool that offers a way to handle two basic aspects:* Constraint Logic Programming *(CLP) 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.* 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.* HYPERPRO *has been designed for logic programming but it can be adapted to other programming paradigms as well.*

MOTS-CLÉS : *PLC, programmation littéraire, gestion de versions, édition structurée, hypertexte, documentation, spécification*

KEYWORDS: *CLP, literate programming, version managment, structured editor, hypertext, documentation, specification*

1. Introduction

HYPERPRO est un outil de documentation et de développement pour la programmation en logique et par contrainte (PLC). HYPERPRO est un système d'aide à l'élaboration de programmes documentés. Il fait partie de la famille des environnements de "programmation littéraire". Il donne à l'utilisateur la possibilité d'éditer, dans un environnement homogène et intégré, des programmes et leurs différentes versions ainsi que leur documentation au sens large : commentaires libres et éléments formels dédiés à la validation et à la mise au point. Ainsi, une application PLC sera élaborée est maintenue (avec ses différentes versions) à travers un même et unique environnement hypertexte.

HYPERPRO est un outil adapté à la PLC grâce à deux caractéristiques originales : la gestion de versions et la documentation associée. Une des caractéristiques de la PLC est en effet l'écriture de programmes courts testés sur de multiples versions. Il n'est pas rare en effet que la manière de "poser" des contraintes influence drastiquement l'obtention plus ou moins rapide et effective de solutions. Il est donc essentiel que l'utilisateur puisse garder, documenter et combiner ces différents programmes (et manipuler ainsi de manière économique un nombre potentiellement exponentiel de versions). Dans cette approche du développement de programme, la documentation est au moins aussi utile que le texte du programme lui même. La démarche s'apparente à la manipulation de spécifications formelles (dont la PLC se rapproche remarquablement) où la documentation en langue naturelle (les "requirements" ou cahier des charges) joue un rôle au moins aussi important que le code.

HYPERPRO a été développé avec l'outil d'édition structurée Thot [QUI 86, QUI 95, QUI 96]. Thot est basé sur la structure logique d'un document et permet de définir des styles de documents, avec leur différentes présentations, à travers des langages dédiés. Thot est en fait très comparable à la technologie XML [W3C]. Son langage $\mathcal{S}$ de définition de la structure d'un document est semblable aux DTD (*Document Type Definiton*) de XML. Ses langages $\mathcal{P}$ et $\mathcal{T}$ de présentation et de translation de documents peuvent se comparer aux XSL (*eXtensible Stylesheet Language*) et XSLT (pour *Transformation*). Notre choix s'est porté sur Thot car il regroupe un certains nombre d'API qui permettent de créer son propre éditeur comme HYPERPRO.

Cet article décrit les fonctionnalités majeurs d'HYPERPRO. Parmis elles on peut citer : différentes vues statiques et synchrones du document, gestion des versions multiples d'un programme au sein du même document, test de ces versions et intéraction avec le système PLC sous-jacent, vues dynamiques (produites à la demande par projections), index, exportations du document sous différents format pour un traitement avec d'autres systèmes. Nous nous attacherons à développer plus précisement les projections et les index. Le mécanisme de projection permet d'extraire et présenter des morceaux appropriés de code selon des critères spécifiques. Les index sont de trois types : l'index classique de mots, l'index des références croisées des relations et l'index des versions de programme. Les deux derniers portent sur la partie code et mettent en évidence la structure hypertexte bi-dimentionnelle d'un document.

HYPERPRO décrit dans cet article correspond aux langages PLC ; mais avec peu de changements, il peut être adapté à d'autres langages comme pour le langage C (cf. [MON 98]).

Dans la section 2, nous présentons la gestion de versions d'un programme puis les différentes vues du document offertes à l'utilisateur et nous terminons par la fonctionnalité de test d'un programme et en suite les exportations vers différents formats.

La section 3 sera dédiée à l'étude des index. La section 4 sera consacrée aux vues dynamiques (ou projections). L'article sera cloturé par une comparaison entre HYPERPRO et d'autres systèmes de programmation littéraire.

2. Fonctionnalités de HyperPro

Cette section présente les fonctionnalités les plus importantes d'HYPERPRO qui sont : les versions d'un programme, leurs tests, les différentes vues statiques et dynamiques d'un document et son exportation vers d'autres formats.

Un document HYPERPRO est un fichier unique hypertexte dit "document principal". Celui-ci a la structure d'un rapport. Il doit avoir un titre, une suite d'au moins une section, une table des matières, un index de mots. Optionnellement, il peut contenir une date, les noms des auteurs et leurs affiliations, les mots-clés, les références bibliographiques, les annexes et d'autres index. HYPERPRO définit une structure spéciale pour les programmes PLC. Ainsi un paragraphe, peut être également une définition de *relation* (ou prédicat). Un document doit contenir au moins une définition de prédicat. Une relation est définie par la donnée d'un *titre* et d'une liste d'au moins une *définition* de la relation. Le titre est défini par un nom de prédicat avec son arité. L'arité est optionnelle dans le cas de buts ou directives qui sont considérés comme cas spéciaux de relations. Une relation est définie par un ensemble de paquets de clauses. Chaque paquet définit une version de cette relation. Le titre de la relation contient une référence à la *version actuelle* de la définition du prédicat suivie éventuellement d'une liste de références aux définitions qui composent les différentes *versions* du programme objet du document.

2.1. *Les vues statiques*

HYPERPRO offre la possibilité d'avoir plusieurs vues du même document présentées simultanément dans des fenêtres synchronisées différentes : la vue principale contenant le document intégral, la table des matières, les commentaires associés aux relations, les assertions associés aux relations, les relations avec les paquets de clauses associées correspondant aux différentes définitions possibles. Lorsqu'un document est ouvert, la vue principale et la table des matières sont ouvertes automatiquement. Les autres vues sont ouvertes à la demande.

En cliquant en un point du document principal, ainsi que dans l'une des vues ouvertes, toutes les vues se synchronisent simultanément pour présenter la partie correspondante du texte ainsi référée.

2.2. *Versions d'un programme*

La possibilité de pouvoir gérer différentes versions d'un programme pendant le développement et la documentation est un dispositif important offert par HYPERPRO. Cette possibilité trouve tout son intérêt dans le cas de la programmation en logique et par contraintes. En effet, il s'avère intéressant de pouvoir intégrer dans un même document des versions qui, par exemple, ne diffèrent que par une clause ou même fondamentalement différentes par les algorithmes mis en jeu.

Chaque relation est définie par au moins un paquet de clauses. Parmis ces paquets de clauses, on distingue un paquet particulier qui est le paquet courant définissant la relation (*current predicate definition* ou *c.p.d.*). Un lien, situé au niveau du titre de la relation, pointe sur ce paquet. Ce lien est dit "référence du prédicat courant" (*current predicate reference* ou *[c.p.r.]*). Le *[c.p.r.]* est mis à jour par l'utilisateur à chaque fois qu'il veut désigner un paquet de clauses comme la définition courante du prédicat. Par défaut, c'est le premier paquet saisit qui est référencé par le *[c.p.r.]*.

Les prédicats appelés dans le corps d'une clause sont munis d'un lien vers leur propres définitions (sauf pour les prédicats localement récursifs). Ces liens sont appelés "référence du prédicat" (*predicate reference* ou *[p.r.]*). Dans la version actuelle du prototype, ces références sont définies à la main par l'utilisateur.

La figure 1(a) montre quelques relations définies pour un programme, avec les *[c.p.r.]* placés dans les titres des relations qui composent le programme et les *[p.r.]* placés au niveau des prédicats appelés dans les corps des clauses des *c.p.d.*. Le document a deux sections : la section 1.1, qui définit les relations `a/1`, `b/2`, et `c/1`, et la section 1.2 qui définit la relation `a/1`. Seule la relation `a/1` de la section 1.1 a deux définitions distinctes. Son *[c.p.r.]* pointe sur la première définition. Les *[c.p.r.]* sont représentés dans la figure par des lignes continues. Les *[p.r.]* sont représentées par des lignes discontinues. Comme s'est indiqué, les *[p.r.]* pointent sur les titres des relations correspondantes.

Les *[p.r.]*, grâce à la possibilité de construire et nommer des versions qui ne diffèrent que par la définition des relations, donnent la possibilité à l'utilisateur d'identifier et documenter différentes versions d'un même programme dans le même document.

Pour nommer une version, une fois celle-ci définie grâce aux *[p.r.]* et *[c.p.r.]*, l'utilisateur sélectionne l'item *name a version* dans le menu *Tools*. Une fenêtre est activée dans laquelle l'utilisateur peut donner un nom de version. Il lui reste alors à pointer une première relation et la version est automatiquement créée en suivant récursivement les pointeurs vers les versions courantes des prédicats utilisés.

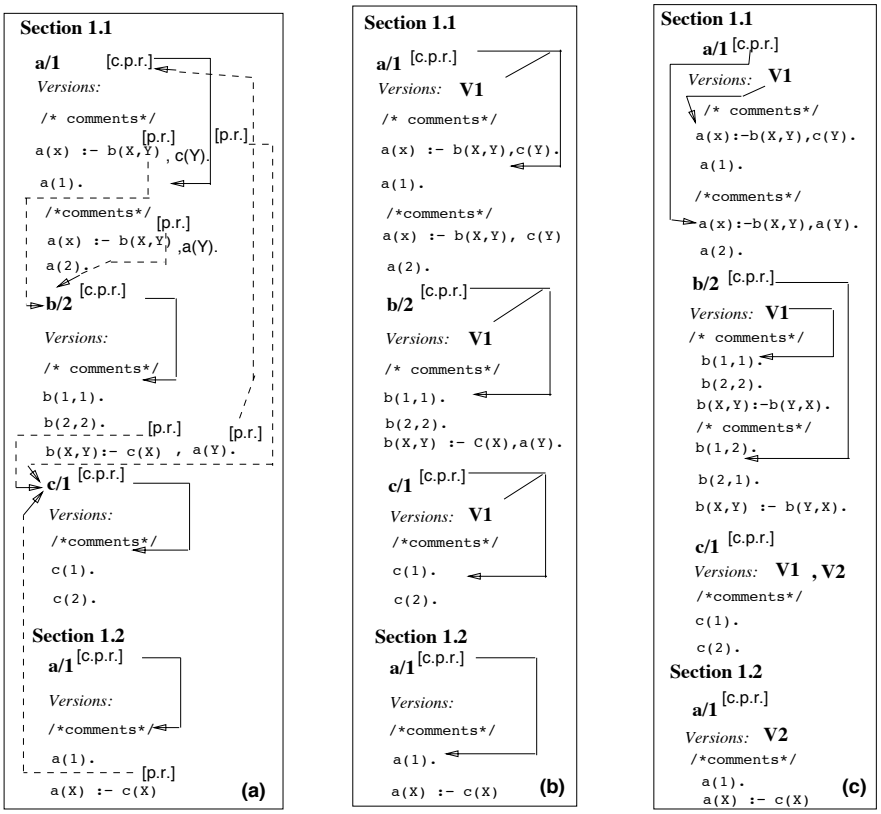

Figure 1. *(a) Quelques relations et des références qui les lient (b) Définition de la version* `V1` *(c) Définition de la version* `V2`

La figure 1(c) montre deux versions différentes d'une même relation. Celles-ci ont été définies en deux étapes. Dans un premier temps la version V1 a été créée (figure 1(b)) en pointant sur la relation a/1 de la section 1.1. Les pointeurs pour V1 pointent sur le c.p.d., à savoir sur la définition du prédicat identifiée par le *[c.p.r.]*.

Puis l'utilisateur a défini une nouvelle section numérotée 1.2, où il a redéfini la relation a/1. Dans cette nouvelle définition du prédicat la deuxième clause de a/1 réfert à la relation c/1, déjà définie dans la section 1.1. Il a donc inséré un *[p.r.]* après la prédication `c(X)`, qui pointe vers la relation c/1 dan s la section 1.1. Il nomme alors cette version "V2". Après avoir choisi la relation a/1 dans la section 1.2, le nom V2 apparaît dans la barre des versions de la relation à côté de son titre, qui pointe vers le même *[c.p.d.]* pointé par le *[c.p.r.]*. Celui-ci apparaît également dans le titre de la relation c/1, puisqu'elle appartient aussi à la version V2.

L'utilisateur peut supprimer une version simplement en utilisant le menu qui permet d'effacer un nom de version. Le système supprime alors automatiquement tous les pointeurs la définissant ainsi que le nom de celle-ci accompagnant les titres des relations qui en faisaient partie.

Dans la figure 1(c), le document HYPERPRO présenté est le même que celui de la figure 1(b). Dans ce cas cependant l'utilisateur a modifié le *[c.p.r.]* de la relation a/1 de la section 1.1. et a ajouté une nouvelle définition pour la relation b/2, en modifiant son *[c.p.r.]* de telle manière qu'il pointe vers sa seconde définition. Pour autant la version V1, déjà définie, pointe toujours vers l'ancienne définition. Pour ne pas surcharger la figure, les pointeurs *[p.r.]* ne sont pas montrés. L'utilisateur peut alors donner, s'il le souhaite, l'ancien nom V1 à ce nouveau programme, et une nouvelle version sera définie automatiquement en suivant récursivement la chaine des pointeurs *[c.p.r.]*/*[p.r.]*.

2.3. *Test d'un programme*

Une fois qu'une version est définie, il est possible d'en vérifier la syntaxe, d'en faire une vue ou de l'exécuter. Afin d'effectuer l'un de ces tests, il est nécessaire de préciser à l'aide d'un menu quel est le langage utilisé. Les modes de test suivants peuvent alors être invoqués : `programme/version` (sélection en cliquant sur le nom de version), `clause/relation` (sélection manuelle des relations constituant le programme), ou `automated recursive mode` (sélection par enchaînement récursif à partir d'une relation initiale).

Une fois le programme (ou une portion) sélectionné, HYPERPRO ouvre une fenêtre d'exécution du test dans laquelle celui-ci est affiché puis testé.

Il est donc possible de tester un programme complet ou seulement une partie analysable. Il est également possible de réimporter le résultat d'un test dans le document initial.

2.4. *Exportations d'un document*

HYPERPRO permet d'exporter le document complet dans les formats ASCII, LaTeX, PostScript et HTML.

L'exportation ASCII est un "dump" des codes ASCII du document et ne sert que lors de la mise au point du système pour récupérer des exemples lors de "crash" de HYPERPRO.

L'exportation LaTeX reflète fidèlement le document, excepté pour les liens hypertextes évidemment. Les index se retrouvent dans le document exporté ainsi que la table des matières.

L'exportation HTML permet de lire le document exporté avec n'importe quel web "browser" en utilisant les hyperliens comme dans le document original. Il est également possible de n'exporter qu'un programme ou une partie à l'aide du menu approprié.

3. Les index

Lorsque le document hypertexte complet est imprimé sur papier en deux dimensions les liens hypertextes sont sans effet et des index deviennent indispensables. Mais même avec le document hypertexte ils sont utiles : ceux-ci peuvent être présentés dans une vue séparée et, grâce aux références sous forme de liens hypertextes qu'ils contiennent, permettre un accès rapide à l'information utile.

Un document HYPERPRO a deux structures hypertexte : les liens entre la relation et sa définition et les liens de versions. Il y a donc deux index utiles : l'*index des références croisées* et l'*index des versions*. Chaque index est construit indépendamment grâce aux menus de l'éditeur. Dés qu'un index est construit une fenêtre correspondante est ouverte.

3.1. *Index des références croisées*

L'index des références croisées contient les relations définies dans le document dans l'ordre où elles apparaissent. Celles-ci sont listées dans une fenêtre avec les numéros des pages où elles sont définies. Chaque numéro correspond également à un hyperlien pointant vers la définition correspondante dans le document original. La figure 2 montre un exemple d'index de références croisées.

En cliquant sur une référence, le document principal, ainsi que toutes les vues ouvertes, se synchronisent simultanément pour présenter la partie correspondante du texte ainsi référée.

3.2. *L'index des versions*

L'index des versions contient les noms des versions (donc des programmes qui ont été définis) dans l'ordre où elles apparaissent dans le document original. Celles-ci sont listées dans une fenêtre avec les relations utilisées dans la version et leur numéro de page. Chaque relation ou numéro correspond également à un hyperlien pointant vers la définition ou l'emplacement correspondant dans le document original. La première relation correspond à la première définition associée à cette version. La figure 3 montre un exemple d'index des versions.

En cliquant sur une référence, le document principal, ainsi que toutes les vues ouvertes, se synchronisent simultanément pour présenter la partie correspondante du texte ainsi référée.

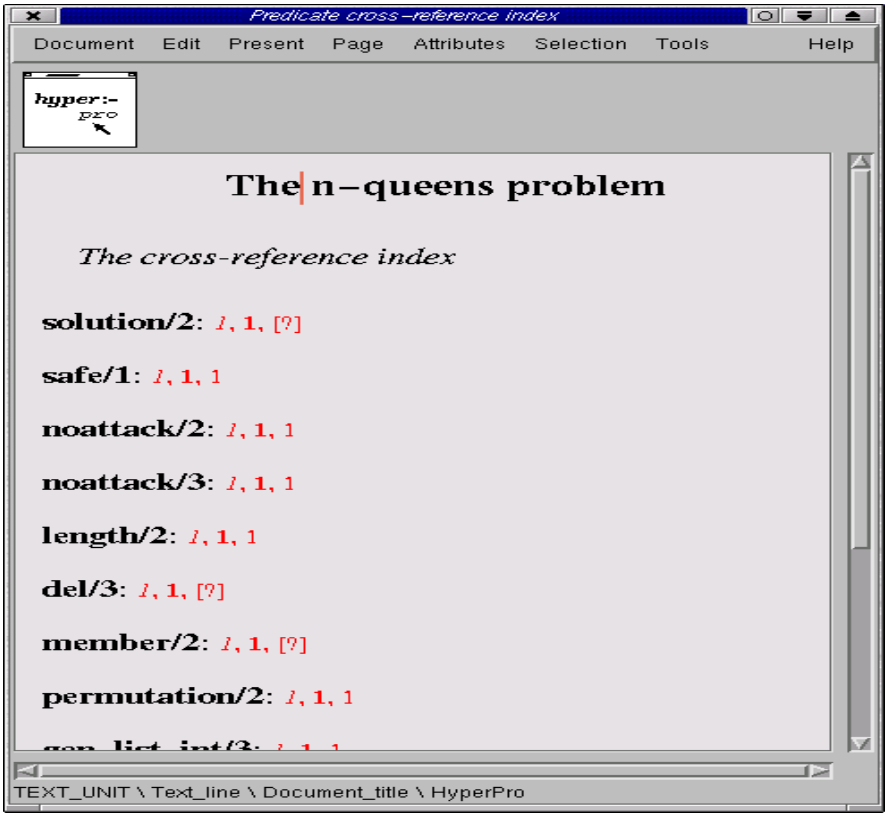

Figure 2. *Vue de l'index des références croisées*

4. Les vues dynamiques : projections

En plus des vues statiques décrites à la section 2.1, et c'est une des originalités de HYPERPRO, il est possible de définir des "vues projections" qui contiennent des morceaux de choix définis par l'utilisateur. HYPERPRO offre les projections suivantes définies selon un *mode* : *manuel*, *automatique*, *version*, *récursif*, *basé index*.

Une "vue projection" permet d'afficher des unités sélectionnées du document dans l'ordre où elles apparaissent dans celui-ci. Les vues dépendent du processus de sélection ainsi que des unités de texte choisies.

La "vue projection mode manuel" permet de mettre dans une seule vue synchronisée tous les paragraphes sélectionnés manuellement par l'utilisateur en cliquant sur les paragraphes choisis dans le document (ou une de ses vues). Ce type de vue est surtout utile lorsque les paragraphes sont sélectionnés automatiquement selon un critère choisi. C'est le but des vues suivantes.

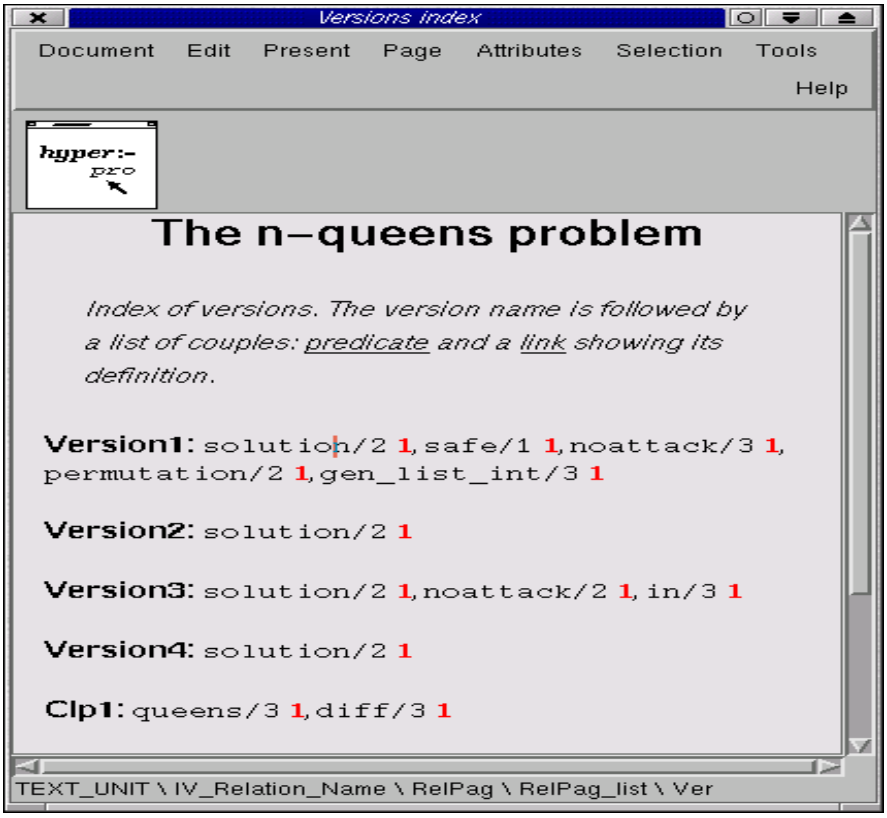

Figure 3. *Vue de l'index des versions*

La "vue projection mode automatique" est obtenue avec un critère d'expression régulière fournie par l'utilisateur. La granularité minimale est le mot. Ceci permet de mettre dans une seule vue toutes les portions de texte ou par exemple un même mot est utilisé. Ceci est particulièrement utile lorsque des modifications de terminologie sont nécessaires, en permettant l'examen de tous les contextes dans lesquels une notion est utilisée.

La "vue projection mode récursif" est obtenue en sélectionnant une relation. Tous les paquets de clauses correspondant à sa définition sont alors mis dans une seule vue ainsi que ceux obtenus en suivant la chaine *[c.p.r.]/[p.r.]*. Ceci est particulièrement utile pour détecter les prédications qui n'ont pas encore de pointeur *[p.r.]* ou pour trouver la définition à laquelle une telle relation correspond.

La "vue projection mode version" est semblable, à ceci près que les paquets sélectionnés correspondent à la seule version indiquée par l'utilisateur.

La "vue projection mode basé index" enfin permet de définir de telles projections à partir d'un index. La figure 4 montre une vue projection correspondant à une version nommée dans l'index des versions de la figure 3.

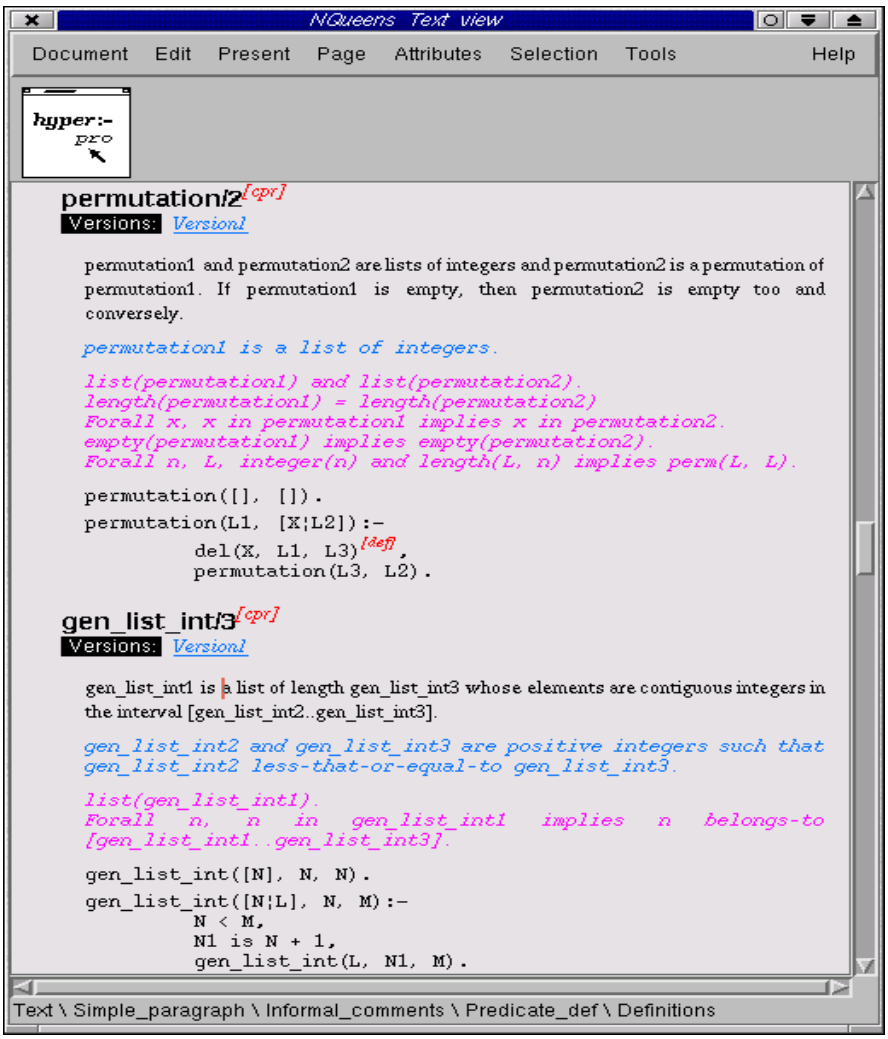

Figure 4. *Vue projection, mode version*

5. HyperPro et d'autres sytèmes de programmation et documentation

Le développement de programmes logiques a été considéré par différents auteurs avec différents points de vues. La plupart se concentrent sur le processus de déve-

loppement par raffinements successifs à partir d'une spécification. Deville [DEV 90] part de formules logiques du premier ordre avec déclaration de modes. De manière générale, une déclaration de mode peut être vue comme un typage. La plupart des systèmes visent à aider le développeur à obtenir d'emblée des programmes correctes ou à vérifier après coup leur correction. En programmation en logique différents systèmes ont été définis. Dans [LOV 91] l'un de ces systèmes est décrit. Dans [DER 93] une approche systématique de la correction est présentée. Quelques unes de ces idées ont été implantées dans le système LDS2 décrit dans [REN 94] et utilisé pour définir une méthodologie d'écriture de spécifications dans le style de la programmation en logique [EDD 92]. Ces approches méthodologiques manquent d'outils pour en permettre une utilisation satisfaisante qui permette l'inclusion d'explications des choix de spécification ou d'implantation.

A l'heure actuelle il n'y a pas non plus d'outil satisfaisant qui soit largement accepté pour réaliser et documenter les programmes PROLOG.

Donald Knuth a introduit *literate programming* sous la forme de Web, un outil pour écrire *literate Pascal* et des programmes **C** [KNU 83, KNU 92]. Cette approche du développement de programmes peut être appliquée à la PLC, mais elle reste incomplète.

Dans cet esprit, de nombreux systèmes ont été développés :

- Web de Knuth pour Pascal et **C** [KNU 83];
- Cweb de Thimbleby [THI 86], une variante de Web;
- Spider de Ramsey [RAM 89] qui est un Web générateur;
- Noweb de Ramsey [RAM 92].

Ces systèmes reposent sur l'idée que le programmeur utilise trois langages : un traitement de texte (ex. LaTeX), un langage de programmation (ex. Pascal) et un (meta)langage qui permet facilement la combinaison des deux dans un seul document. Un "literate program" contient donc des parties de programme combinées avec des portions de texte. Un système de "programmation littéraire" fournit des outils permettant d'extraire du document unique le programme et de générer un document avec la documentation complète du programme, des index, des références croisées, etc.

Cweb [THI 86] est un outil pour produire une documentation de programme sous forme d'une combinaison de **C** et *troff*. Cweb diffère de Web essentiellement par le choix du langage (**C**) et du traitement de texte (*troff* ou *nroff* au lieu de TeX).

L'objectif de Spider [RAM 89] est de développer de programmes **Ada** certifiés. L'utilisation de Web directement est rendue difficile du fait que le langage objet (langage utilisé pour spécifier l'éditeur structuré) est **SSL**. De plus les seuls langages pour lesquels il existe un système Web sont Pascal et **C**. Spider est un générateur Web semblable à un générateur d'analyseur syntaxique. L'utilisation de Spider permet de construire un Web sans avoir à comprendre les détails de l'implantation et il est facile d'adapter le système à un nouveau langage.

Norman Ramsey a proposé un nouveau système appelé Noweb [RAM 92], et qui veut être un outil simple et extensible. Il a été développé pour Unix et peut être porté sur d'autres systèmes qui peuvent supporter les "pipelines", ANSI-C ainsi que awk ou icon. Noweb peut également travailler en HTML. Un fichier Noweb est une séquence de *chunks*. Chaque "chunk" contient une portion de programme ou documentation. Leur ordre est quelconque. Ceux qui contiennent du code ont aussi des références vers d'autres "chunks". Ramsey considère que Noweb est plus simple que Web, vu qu'il est indépendant du langage de programmation; mais cela signifie qu'il est aussi moins puissant. Noweb utilise TEX et LATEX. Noweb est extensible : de nouveaux outils peuvent être intégrés sans avoir à reprogrammer. Son exportation (*Weave*) préserve les espaces et indentations lors de l'expansion des "chunks". Ceci est nécessaire pour documenter correctement des langages comme Miranda ou Haskell.

Il ressort de cette étude qu'aucun système n'a pu aller au delà du prototype traitant des programmes relativement petits. Programmer à une large échelle est encore un objectif lointain pour de tels systèmes. La question de développer en parallèle la documentation et le programme reste encore marginale.

Notre objectif avec HYPERPRO est de réaliser un outil permettant d'emmagasiner l'ensemble de l'expérience accumulée lors du développement d'une application de taille a priori non limitée, tout en permettant l'accomplissement de l'ensemble des tâches de mise au point. Ceci est réalisé par l'utilisation et l'adaptation de l'outil Thot qui permet l'édition de documents structurés hypertexte avec des vues synchronisées, ainsi que la définition d'applications (API) répondant à ces différentes tâches de mise au point.

6. Conclusion

Nous avons présenté un système intégré de rédaction/documentation de programmes PLC basé sur le système Thot [QUI 95], appelé HYPERPRO, dont l'objectif est de faciliter considérablement la mise au point de ces programmes.

Il permet de manipuler le document complet à plusieurs niveaux d'abstraction simultanément, avec différents points de vue. Les programmes peuvent être modifiés et documentés simultanément tout en gardant les différentes versions utiles. Toute l'expérience acquise au cours du développement d'une application peut donc être clairement collectée dans un même document structuré hypertexte et facilement retrouvée. Le résultat de l'ensemble des expérimentations peut y être rassemblé, permettant d'y inclure aussi bien les versions définitives ou simplement intéressantes, que l'historique du développement.

Ce papier décrit la version de base du système HYPERPRO dont le but est de permettre une expérimentation pour en tester les fonctionnalités. Une utilisation effective nécessiterait d'avoir quelques fonctionnalités supplémentaires comme la possibilité d'importer des programmes existants. Toute application utilise en général des bibliothèques ou certains de leurs composants qui peuvent nécessiter quelques adaptations

qui doivent être documentées. De plus, dans la version actuelle, le nombre de vues et projections est limité. Ceci est dû à des restrictions de Thot. Notre expérience de réalisation de spécifications nous a montré que dans le processus de développement de programme avec sa documentation, il peut être extrèmement utile de pouvoir travailler avec plusieurs vues et projections correspondants à différents concepts mis au point simultanément.

HYPERPRO peut aider à mettre au point des applications, et il semble qu'il puisse jouer un rôle significatif dans l'écriture de spécifications logiques et leur implantation. Dans ce cas en effet le code formel et son explication en langage naturel sont particulièrement enchevêtrés et des facilités de lecture deviennent alors essentielles.

7. Bibliographie

[DER 93] DERANSART P., MAŁUSZYŃSKI J., *A Grammatical View of Logic Programming*, The MIT Press, 11 1993.

[DEV 90] DEVILLE Y., *Logic Programming: Systematic Program Development*, Addison Wesley, 1990.

[EDD 92] ED-DBALI A., DERANSART P., « Software Formal Specification by Logic programming », FUCHS N. E., COMYN G., Eds., *LPSS'92*, n° 636 LNAI, Zurich, 1992, Springer Verlag, p. 278–289.

[KNU 83] KNUTH D., « The `web` System of Structured Documentation », rapport n° 980, 9 1983, Stanford Computer Science.

[KNU 92] KNUTH D., *Literate Programming*, N° 27 CSLI lecture notes, Center for the Study of Language and Information, 1992, pages 349–358.

[LOV 91] LOVELAND D., « Near-Horn Prolog and Beyond », *Journal of automated Reasoning*, vol. 1, 1991, p. 1–26.

[MON 98] MONTAGNE S., « HyperPro(C) : un environnement pour la programmation en C », Rapport de stage de fin d'études, 1998.

[QUI 86] QUINT V., VATTON I., « Grif : an interactive System for structured Document Manipulation », *International Conference on Text Processing and document Manipulation*, Cambridge University Press, 11 1986, p. 200-213.

[QUI 95] QUINT V., « The Thot User Manual », rapport, 1995, INRIA.

[QUI 96] QUINT V., « The Languages of Thot », rapport, 6 1996, INRIA, translated by Ethan Munson.

[RAM 89] RAMSEY N., « Literate Programming: Weaving a language-independent `web` », *Communications of the ACM*, vol. 32, n° 9, 1989, p. 1051-1055.

[RAM 92] RAMSEY N., « The noweb Hacker's Guide », Princeton University, 9 1992, Revised 08/1994.

[REN 94] RENAULT S., DERANSART P., « Design of Redundant Formal Specifications by Logic Programming: Merging Formal Text and Good Comments », *International Journal of Software Engineering and Knowledge Engineering*, vol. 4, n° 3, 1994.

[THI 86] THIMBLEBY H., « Experiences of 'Literate Programming' using Cweb (a variant of Knuth's `web`) », *The Computer Journal*, vol. 29, n° 3, 1986, p. 201-211.

[W3C] W3C, « XML W3C Recommendation », http://www.w3c.org/XML.