The Intellectual Training Environment for Prolog Programming Language

Serghei PELIN, Chişinău, Republica Moldova,
serghei.pelin@yahoo.com

In this work is described a new complex training system, named SPprolog, intended for training and self-training in logic programming language - Prolog. This system includes elements related to Prolog and logic programming, and the elements of independent, complex, self-sufficient training system which is capable considerably to increase the quality of self-training, and to be effective assistant in training. The most useful application of the system can be in distance education and self-training. The main elements of SPprolog system are:

- *Functionally expanded (in comparison with existing systems) Prolog development environment, with the multipurpose code editor, the automated organization system of the personal tools, automated advice mode "Expert Advice", based on the incorporated expert system for cultivated, effective and optimized programming;*
- *Link to foreign Prolog programs compiler which allow to compile the program to independent executable;*
- *Built in intellectual, interactive, multimedia Prolog interpreter integrated with expert system and the elements of the intellectuality, allowing to lead detailed program interpretation, with popular and evident, explanation of the theory and mechanisms used in it, applying audiovisual effects to increase the level of naturalness of process of explanation;*
- *Full digital training course of Prolog programming language presented in the form of the matrix of knowledge and supplied system of consecutive knowledge reproduction for self-training and evaluation;*
- *an intensive course of training to the Prolog language and SPprolog system, based on the programmed, consecutive set of actions, allowing using the previous two mechanisms of system for popular and evident explanation of the main principles of work of system and Prolog language.*

***Keywords**: training, prolog, environment, SPprolog.*

Introduction

The intellectual training environment for prolog programming language- SPprolog, is basically intended for training and self-training in Prolog programming language. It can be applied in any educational institutions (schools, high schools, specialized courses) by trainees - as environment for programming and evident understanding of results of the made program, as the experimental environment for studying some language mechanism; by trainer during an explanation of various language mechanisms or during an explanation of a principle of work of some program.

However it is supposed, that the basic success of the given system should be during self-training of Prolog language, because the system itself will tell and will explain a new theme under the incorporated plan; will offer necessary examples, in detail and evidently will explain a principle of their work. During the independent programming will substitute the teacher with valuable and duly advice from the incorporated expert system.

System Functionalities:

1. Development environment
   a. Multifunctional editor with Prolog syntax highlighting, code completion for standard and user predicates, code snippets, code outlining, bookmarks, split view, etc…
   b. Automated user tools organizer which permit quick saving of selected code as independent tool; quick insert of user tools; "on fly" verification for duplicated predicates.

c. Automated advice system "Expert advice". This system permit "on fly" syntax verification, advice for cultivated programming, code analyzing and giving advice for more "optimized" code (more quickly, more evidently, more shortly), notification for senselessly code.

2. Foreign prolog compiler which allow code error verification, and compiling the program to separate executable.

3. Intellectual, interactive, multimedia Prolog interpreter integrated with expert system and the elements of the intellectuality, allowing leading the detailed program interpretation, with popular and evident, explanation of the theory and mechanisms used in it.

a. Automated code scrolling while interpretation
b. Visual selection of the part of the program which is interpreted
c. Visualization of all input and output parameters
d. Animated demonstration of unification process
e. Animated transitions of executed actions
f. Animated backtracking demonstration
g. Scoring the executed actions and transitions
h. Automated building and visualization of program solution tree
i. Detailed explanation of executed action
j. Hot access to digital manual information
k. Expert Advice

4. Digital training course of Prolog programming language presented in the form of the matrix of knowledge and supplied system of consecutive knowledge reproduction for self-training and evaluation.

The development environment. We understand all those tools, as development environment which are applied direct or indirect for writing the program. As one of the basic specified tools the editor of a code acts. The idea of own cod editor creation has arisen during using various editors of existing realizations of Prolog language, during teaching the Prolog to students, and also during personal professional work. Existing realizations Prolog languages a prologue as the most powerful, and less widespread give few attention to process of coding (writing the program), and concentrate mostly on internal language mechanisms and its final opportunities. Actually the process of writing the program is poorly automated, and is poorly distinct from entering text in the simple editor, which allows only typing various symbols.

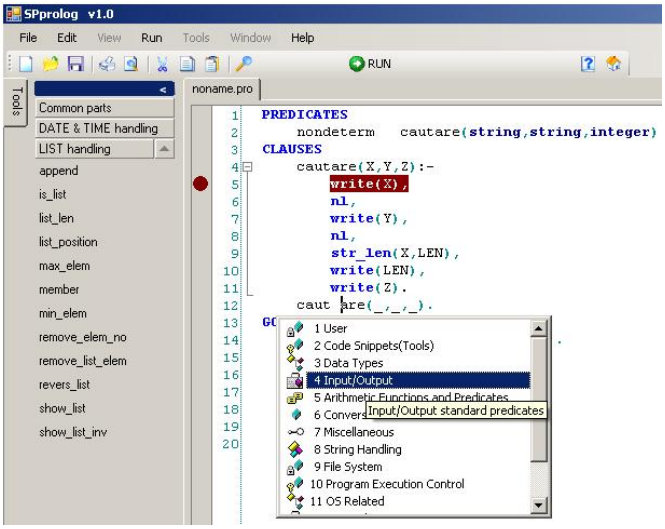

Fig.1. Prolog development environment

This became one of principal causes of difficult understanding of this language by beginners, as before it can check up result of the written program, it faces a problem – the necessity of writing a correct program. The absence of syntax highlighting , absence of the unequivocal code organization, absence of possibility to auto-insert the predicates

names, together with weak syntax knowledge and a small experience strongly complicate and "extend" the process of writing the program. At the same time for more experienced users in Prolog, the absence of elementary possibilities of the organization, and code navigation also is not positive property of the editor if the code size is rather big. Only navigation on a code needs a lot of time, the strengthened attention and good memorizing of the program text.

It has been decided to introduce in SPprolog system the multipurpose code editor, which whenever possible would not have the listed lacks, and would promote faster, easy and correct typing of the program. Besides this it has been decided to add a little, not dependent from the editor tools which also would optimize the work of the user while writing the program. It is possible to declare with confidence that the given development environment is currently one of the most advanced Prolog tool, among existent Prolog development environments (Fig 1).

The intellectual, interactive, multimedia Prolog Interpreter (further the Interpreter). The interpreter is a set of tools providing, training effect in the given system. The basic purpose of the interpreter is, first of all, to interpret the user program, however the feature of the given interpreter is not only naked interpretation of some program, but also an audio-visual explanation of the process of execution (interpretation) of the given program by the Prolog language, with comments, explanations, references to the theory. The given interpreter can be compared to the teacher which not only displays on a board (or orally) the result of program execution, but step by step explains to a trainee how the given process is executed, what language mechanisms are involved, what operation and why it is carried out at present time, and if necessary can give more simple similar example, quote an extract from a theoretical material or to stop at some stage, to explain the theory of this or that mechanism, and then to continue. Exactly this function is incorporated in the interpreter of the given system as the basic function for the sake of which it and has been created. With minor reservations the given interpreter can be named the artificial teacher of Prolog language.

The interpreter should show *how* the Prolog language works from within. After theoretical material this system will help on the one hand to fix the studied material on the other hand to understand the practical mechanism of its application.

Special highlight of the given interpreter is the opportunity of full scoring by a "human" voice of all explanations and occurring operations. Appear the effect of explanation of interpretation process by the person not by the artificial system. It becomes possible because of developed interpretation algorithm which contains final number of possible actions and operation during interpretation of any Prolog program, at the same time infinite number of their combinations arising at interpretation of any program, does not make impression of the "simple" sounding robot. All explanation operation, besides a voice, are also displayed in the text mode which can be logged, and then printed to be studied later on. At the same time, the unique system of visual support and display of interpretation process, has been developed for more accessible and evident explanation (Fig.2), which on strictly determined algorithm displays directly in the program code: all used parameters, process of transition from one action to another, special marks and designations for evident display of internal Prolog language mechanisms (such as backtracking, recursive rules, etc.). For more evident demonstration are applied not static, but the animated image. It not only draws attention to a place of interpretation, but also enables trained "to track" carried out action, to have time to realize what has occurred. The given technology is much more effective than simple appearing on the screen of some information, because gives the possibility to track some information sequence that occurred, and understanding whence there is this or that element of data.

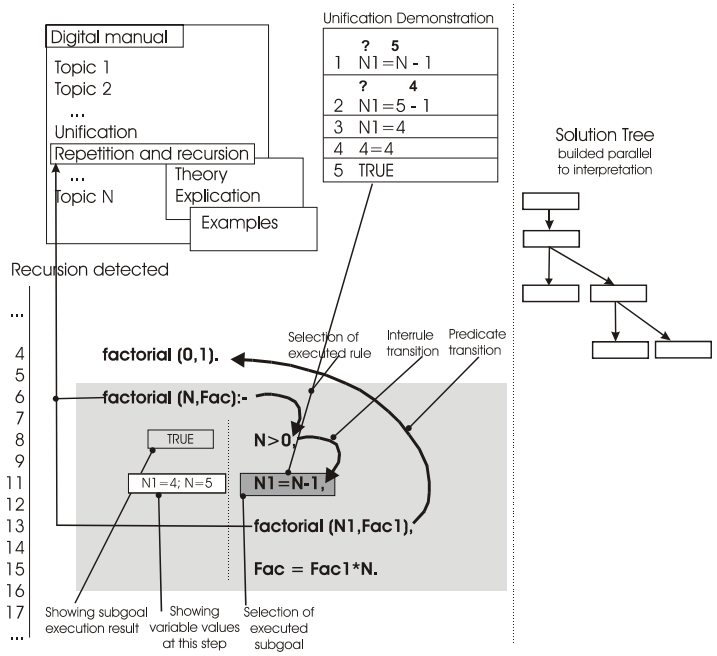

Fig.2. System of visual support

For example, if at a stage of transition from one subgoal to another subgoal there is some unification process, and all this to display by instantly appeared arrow of transition and showing the result of unification process in some cases even such simple action can cause difficulty in understanding. The technology applied in given system, will make following actions: From allocated proceeding subgoal the arrow will start to proceed aside entering subgoal, then entering subgoal will be selected, will be displayed all the subgoal parameters. With animation and sound the unification process will be shown. Then interpreter will finally display that subgoal is processed and executed, and entering subgoal should be processed. It resembles some animated film which even without sound support should accessible to explain that has occurred. In relation to statically appearing pictures the given method should become clearer. In this system there is a threefold effect promoting perception and understanding of the information: the voice explanation, animated graphic demonstration, and directly interpretation of the program or in other words educational tracing of the program.

Intellectuality of the interpreter consists in an opportunity of a finding in the extensive knowledge base of such set of explanations which would correspond to a current interpreted part of the program, and in case of the request of the programmer to give them. Also the interpreter has an opportunity to pick up from base of well explained simple examples, such example which would correspond to an interpreted program part, and can be useful to detailed understanding of some moment in the program or actually process of interpretation of the given program part.

The Interpreter is not the whole and indivisible mechanism in system, it more likely a set of small functions and mechanisms which together carry out one general function - evident interpretation. It enables both infinite expansion of the given mechanism, reduction of functionality if necessary. So each user can adjust the interpreter thus which will correspond as much as possible to necessity of the concrete user.

Full digital training course. It is a high-grade training course of Prolog programming language which consists of 12 theoretical sections, a set of the described and explained examples, and also comments of independent experts. The digital training course in SPprolog is structured with the Matrix of Knowledge Elements (MKE) [2,6]. The MKE permit the data in digital training course to be not only the static theoretical information, but it also is a part of the knowl-

edge base used by expert system.

In general, MKE is presented as $\mathbf{A}=\langle a_{ij}\rangle$. In logic presentation MKE can be presented thought the multitude of predicates:

matrix (a11,a12, …, a1N).
matrix (a21,a22, …, a2N).
…
matrix (aM1,aM2,…,aMN).

Every of the mentioned predicates with the arity N, at its row, can be transformed in production of binary predicates. From binary to unary. The multitude $A= \langle a_{ij}\rangle$ will form a language **L** of some theory **T**, which will describe the active knowledge. Elements **i** of MKE will consist from:

<concretization, definition, demonstration, other opinion, glossary, bibliography>.

It is permitted the modification of element **i** conformity with determined by user concept of this MKE . For the element ***j*** will be determined Aristotle's categories, as follows:

<essense, quantity, quality, in comparation with ..., …>.

If it is necessary the ***j*** elements of the matrix also can be modificated.

It is considered, that every of element $\mathbf{a_{ij}}$ can be interpretated only as **TRUE.** To determine the boundary of any two neibourn elements of MKE, for example $a_{i,j}$ and $a_{i+1,j}$ of MKE will use the notion of distance of Hemming **d** wich is presented as:

$$d(a_{i,j}, a_{i+1,j}) = \Sigma \left/ \mu_{ai,j}(x_k) - \mu_{ai+1,j}(x_k) \right/,$$

where $_j(x_k)$ and $\mu_{ai+1,j}(x_k)$, - linguistic variable, wich define the level how x belong to $a_{i,j}$ or $a_{i+1,j}$.

Conclusion. The offered complex elements training system to language logic a prologue, harmoniously supplement each other and together make effective and productive training system. The given system can become the irreplaceable tool during training and especially during self-training and distance learning. Due to elements of intellectuality and specific effects the system can raise considerably the quality of training, and reduce the time spent for training and human resources.

Bibliography

1. S.Pelin. Work optimizer and intellectual assistant for Prolog Development Environment. International Conference: Knowledge Management, Bucharest, November 9-10,2006 .- p.185-190
2. N. Pelin, S.Pelin, A.Pelin "The matrix of Knowledge Elements" – effective technology for knowledge management. International Conference: Knowledge Management, Bucharest, November 9-10,2006 .- p.63-71
3. N. Pelin, S.Pelin. A pattern of a modern expert and outlooks of using Prolog, the language of logic programming. In book: "Information Society". The proceedings of the fourth International Symposium on economic informatics, mai 1999. Inforec Printing House. Editura economica.Bucuresti- p.777-784.
4. N.Pelin, S.Pelin, I. Orlovsky, A.Miron, Y.Grinev, "Problems of Creation of Prolog Operating System" CLPSE 2002: (Constraint) Logic Programming and Software Engineering. Proceedings of ICLP'02 workshop, International Conference on Logic Programming, Copenhagen, July-August 2002
5. N.Pelin, S.Pelin, I.Orlovsky "Problems of knowledge formalization and their usage at distance learning." The 20th ICDE World Conference on Open Learning and Distance Education Dusseldorf, Germany, 01 - 05 April 2001
6. Pelin N., Pelin S., Pelin A. Structurizarea si sistematizarea cunostintelor: teoria si aplicarea "matricei elementelor de cunostinte". Informatica in aplicatii. Culegeri a materialelor conf. a VIII-a, IX-a si a X–a stiintifico-metodice si practice a profesorilor si studentilor "Stiinte aplicative in perioada de tranzitie". USAM, 4-5 decembrie 2002, 4-5 decembrie 2003, 25-26 mai 2004, Chisinau, - Ch.: USAM, 2004.
7. I. Bratko. Programming for Artificial Intelligence. Addison Wesley 2001
8. Robinson J. A. Logic programming – past, present and future. - New Generation Computing, 1, 1983, p. 107-121.
9. Andre Thayse, Pascal Gribomont, Georges Louis etc. Apprche Logique de l'intelligence artificielle 1 de la logique classique a la programmation logique.Bordas, Paris, 1998.
10. Sterling L., Shapiro E. The art of PROLOG. – London: Massachusetts Institute of Technology, 1986 – 333 p.
11. Doores J., Reiblein A.R., Vadera S. PROLOG – programming for tomorrow. – Wilmslow: Sigma Press, 1987 – 142 p.
12. Hogger C. J. Introduction to logic programming – Academic Press, Inc. 1984 – 348 p.
13. K. Maung Yin, D. Solomon. Using Turbo Prolog, que Corporation Indianapolis, Indiana, 1987
14. Gerbert Shildt. Turbo – Prolog. Version 1.1. Borland – OSBORNE/ MsGRAW –HILL.
15. A. Janson. Turbo-Prolog Kompakt, Moscova, Editura "Mir", 1991
16. PDC Prolog User's Guide, Prolog Development Center A/S 1992
17. Visual Prolog, Language Tutorial, Prolog Development Center A/S, 1996
18. Meszaros. TURBO PROLOG 2.0 ghid de utilizare, Cluj-Napoca ,1996
19. V.Cotelea. Programare în logică. Chişinău, Editura Nestor, 2000
20. http://en.wikipedia.com
21. http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html
22. http://www.amzi.com/AdventureInProlog/advfrtop.htm
23. http://kti.ms.mff.cuni.cz/~bartak/prolog/index.html
24. http://www.coli.uni-saarland.de/~kris/learn-prolog-now/