A System to Visualize and Interact with Prolog Programs

José Paulo Leal

Laboratório de Inteligência Artificial e Ciência de Computadores
R. do Campo Alegre, 823 / 4100 Porto
Portugal
fax: +351 2 6003654
email:zp@ncc.up.pt

Abstract. We present a system for visualization and interaction with Prolog programs using a structural editing approach, that may be extended to other logic languages since it is based in concepts from the Logic Programming (LP) paradigm.

We present a system to visualize and to interact with the execution of Prolog programs. The clauses of the predicates to be visualized or interacted with are **annotated** with terms that specify their visual syntax and those predicates are **interactively solved** using a graphical interface, where the user specifies the goal and the clause for the next derivation step.

This approach is motivated by the analogy between resolution of a logic program and syntactic analysis of a context free grammar [9]. It allows us to use the knowledge of attribute grammars in interactive editing [1, 3], particularly in the incremental evaluation of attributes [5] and integrates the concepts of interaction and visualization LP in the paradigm.

The system is implemented in Prolog and uses a portable interface package [8] to create interfaces for X Windows. It is composed of tree main modules to solve annotated predicates, evaluate incrementally the attributes, and manage the interface. The system has also a module to specialize by partial evaluation the incremental attribute evaluator for a set of annotated predicates.

The system works with standard Prolog program files with some annotated clauses that are transformed during compilation. The annotations are terms rewritable to a set of assignments, where the left hand side refers the attribute of a goal and the right hand side is an expression involving arithmetic operators, functions, constants and other attributes. It does not intend to be an interactive Prolog engine. In fact, only the predicates where user selection is pertinent should be annotated, and only those clauses would be interactively solved.

We intend to use this system for several tasks in the development of Prolog programs, in particular for:

- visualizing/editing data structures
- debugging and tracing of programs
- management of user interaction

The main characteristics of this system are a result of the following decisions:

Execution oriented versus code oriented: The adopted model is execution oriented although for some purposes a code oriented visualization could be preferable. In the latter case it is possible to use some annotated meta program to manipulate the program whose code we are interested in visualizing.

Restricted input versus input-compilation-execution cycle: In this system the user inputs are restricted to valid ones: the selection of pending goals or unifiable clauses. It prevents input errors but it is applicable only when the input is a tree, failing if its a D.A.G. (general graphs are even more problematic) as those resulting from the unification of logic variables. We are still looking for the best way of merging this two types of interaction to solve this problem.

Expressiveness versus efficiency: We were more concerned with expressiveness rather than efficiency in the definition of the annotation language. Nevertheless, attribute evaluation is more efficient than, for instance, constraint solving (another common approach for describing a layout) and using partial evaluation optimization the interaction is rather fast.

References

1. R. Balk, G. Snelting, *The PSG System: ¿From Formal language Definitions to Interactive Programming Environments*, ACM Transactions on Programming languages and Systems, Vol. 8, In 4, (1986:10), 557-608.
2. A. Jorge, *Using EDIPO*, Centro de Informática, Universidade do Porto, 1990.
3. S. Horwitz, T. Teitelbaum, *Generating Editing Environments Based on Relations And Attributes* ACM Transactions on Programming languages and Systems, Vol. 8, In 4, (1986:10), 557-608.
4. C. Hogger, *Essentials of Logic Programming*, Graduate Texts in Computer Science, Oxford University Press, 1990.
5. S. E Hudson, *Incremental Attribute Evaluation: The Flexible Algorithm for Lazy Update* ACM Transactions on Programming languages and Systems, Vol. 13, No3, July 1991, 315-341.
6. J. P. Leal, *The Ytoolkit: the Prolog approach to an users interface*, ICLP Preconference Workshop on Logic Programming Environments, Eilat, Israel, 1990.
7. J. P. Leal, *An History Based Interface*, ICLP Preconference Workshop on Logic Programming Environments, Paris, 1991.
8. J. P. Leal, J. P. Santos, *Towards the portable interface for Prolog applications - Reference Manual - X-Windows Interface*, Centro de Informática da Universidade do Porto.
9. J. Maluszyńky, *Attribute Grammars and Logic Programs: A Comparison of Concepts* Lecture Notes in Computer Science, 545, Springer Verlag.
10. E.Yardeni, E. Shapiro *The Type system for Logic Programs* Journal of Logic Programming, (1991:10), 125-153.