Towards a glass-box typed CLP language

Antonio Fernández
Dpto.Lenguajes y Ciencias de la Computación,
Universidad de Málaga,
29071 Teatinos,
Málaga, Spain
e-mail: afdez@lcc.uma.es
Fax-number: +34-5-2131397

Abstract

Constraint Logic Programming (CLP) merges two declarative paradigms: constraint solving and logic programming. This combination increases the efficiency of logical programming languages and, in some cases, makes programs more expressive than other kinds of programs. CLP has progressed in several quite different directions such as parallelism, concurrency, development of satisfiable algorithms in various domains, program verification, etc, and constraints have also been embedded in a number of other languages and system, having been proven to be useful for a wide variety of applications [WAL 96] in different disciplines such as combinatorial algorithms, operations research, databases, symbolic computation, artificial intelligence, concurrent computation, engineering, numerical analysis, etc.

Recently a large amount of research has been done in the search of declarative and efficient CLP languages. CLP languages like *CHIP* [vHE 89] or other instances of *CLP(X)* language family (such as *CLP(R)* [JAF 92]) allow the user to write Prolog-style clauses only containing, often in some cases, predefined constraints which will be sent to a built-in constraint solver. This solver is a *black-box* from the user's point of view. A new approach, called *glass-box* approach, is being developed. Languages such as *clp(FD)* [COD 96] or *CHR* [FRU 94] provide primitive constraints from which to build more complex constraints. These approaches allow the user to control the constraint solver at a more detailed level. However, the *clp(FD)* and *CHR* are two very different *glass-box* languages where one (*CHR*) has the advantage of expressiveness and the other (*clp(FD)*), efficiency[FER 97]. An ongoing direction is to investigate the combination of these two *glass-box* languages capturing the *clp(FD)* efficiency and the *CHR* expressiveness.

At the same time and over the last decades, the traditional logic programming has experimented with a wide variety of extensions. The need for an improved environment for logic programming has encouraged research into other forms of abstraction such as types ([PFE 92]) and modules ([HIL 93]). Logic languages that combine types and modules with other extensions include *Gödel* [HIL 94] and $\lambda Prolog$ [NAD 92]. Such languages have shown the advantages of such an integration. It is important to clarify what advantages, if any, there may be in the integration of extensions such as (user-defined) types with constraints in a logic language. Since several domains may be combined in a single constraint language,

it would be desirable to have some general means to test whether a given combination of domains and predicates should or not should be allowed in a program. This direction of research was suggested before in [COH 90].The logic and functional paradigms offer a wide set of strongly typed languages which could be the beginning for the integration of (user-defined) types in CLP.

From the two directions shown above, depending on the outcome of a study on the integration of types in CLP (*black-box* and *glass-box* constraints), we intend to design a typed CLP language that maximises both expressiveness and efficiency.

References

[COD 96] CODOGNET P. and DIAZ D., *Compiling Constraints in clp(FD)*. In The journal of Logic Programming,27:1-199,1996.

[COH 90] COHEN J., *Constraint Logic Programming Languages*, In Communications of the ACM, Vol.33 (7), pp:52-68, July, 1990.

[FER 97] FERNÁNDEZ A. and HILL P., *Finite Domain solvers compared using Self-Referential Quizzes*, Technical Report ref.97.03, School of Computer Studies, University of Leeds, January, 1997.

[FRU 94] FRÜHWIRTH T., *Constraint handling rules*. In A. Podelski, editor, Constraint Programming: Basics and Trends, pp:90-107, LNCS 910, May 1994.

[HIL 93] HILL P., *A Parameterised Module System for Constructing Types Logic Programs*, In Proceedings of IJCAI93 Thirteenth International Conference on Artificial Intelligence, vol.2, pp:874-880, Morgan Kaufmann, 1993.

[HIL 94] HILL P. and LLOYD J., *The Gödel Programming Language*, The MIT Press, Cambridge, Massachussetts, London, England, 1994.

[JAF 92] JAFFAR J. MICHAYLOV S. STUCKEY P.J. and YAP R.H.C., *The CLP(R) Language and System*, ACM transactions on Programming Languages and Systems, Vol.14, No.3, July 1992. Pages 339-395.

[NAD 92] NADATHUR G. and PFENNING F., *The Type System of a Higher-Order Logic Programming Language*, in [PFE 92], pp.245-282, 1992.

[PFE 92] PFENNING F., editor, *Types in Logic Programming*, Logic Programming Series, the MIT Press, Cambridge, Massachusetts, London, England, 1992.

[vHE 89] Van HENTENRYCK P., *Constraint Satisfaction in Logic Programming*, Logic programming Series, The MIT Press, 1989.

[WAL 96] WALLACE M.,editor, *Proceedings PACT96, Practical Application of Constraint Technology*, publisher Prolog Management Group, 1996.