CLP(\texttt{R}) and Some Electrical Engineering Problems*

NEVIN HEINTZE and SPIRO MICHAyLOV  
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

PETER STUCKEY  
Department of Computer Science, University of Melbourne, Parkville 3052, Victoria, Australia

(Accepted: 2 August 1990)

Abstract. The Constraint Logic Programming Scheme defines a class of languages designed for programming with constraints using a logic programming approach. These languages are soundly based on a unified framework of formal semantics. In particular, as an instance of this scheme with real arithmetic constraints, the CLP(\texttt{R}) language facilitates and encourages a concise and declarative style of programming for problems involving a mix of numeric and non-numeric computation.

In this paper we illustrate the practical applicability of CLP(\texttt{R}) with examples of programs to solve electrical engineering problems. This field is particularly rich in problems that are complex and largely numeric, enabling us to demonstrate a number of the unique features of CLP(\texttt{R}). A detailed look at some of the more important programming techniques highlights the ability of CLP(\texttt{R}) to support well-known, powerful techniques from constraint programming. Our thesis is that CLP(\texttt{R}) is an embodiment of these techniques in a language that is more general, elegant and versatile than the earlier languages, and yet is practical.

Keywords. Constraints, logic programming, circuit analysis, circuit synthesis, field analysis, signal flow analysis.

1. Introduction

The Constraint Logic Programming Scheme \cite{8} defines a family of declarative and formally-based languages for reasoning about constraints. An instance of this scheme, the CLP(\texttt{R}) language \cite{6, 9}, deals particularly with arithmetic constraints. In this paper we demonstrate the features of CLP(\texttt{R}) with some problems from electrical engineering. This important problem domain, requiring a variety of different solving techniques, provided the driving examples for the pioneering work on constraint programming \cite{3, 15, 18}. The complexity of the systems arising in electrical engineering problems, together with the large number of arithmetic constraints typically involved, made this problem domain an obvious candidate for the application of constraint programming languages.

It is not our intention in this paper to propose new techniques for solving these problems, but rather to show how effectively the well-known techniques developed by

* An earlier version of this paper appeared in the proceedings of the 4th International Conference on Logic Programming, Melbourne, May 1987. Much of this work was carried out while the authors were at Monash University, Melbourne, Australia.
researchers in constraint programming can be used in conjunction with the CLP(91) language. In fact, while these techniques could previously be applied only in special purpose hand-crafted code, here we incorporate them in simple programs within a cleaner and more general programming framework. Furthermore, while CLP(91) cannot claim to solve all the problems of constraint programming, it is more easily applicable to a much larger class of problems than any of its predecessors. Significantly, our experience has also indicated that the resulting programs are usually practical – often surprisingly so.

The remainder of the paper is organized as follows. In Section 2 we give a brief description of the CLP(91) language and system. Then in Section 3 we examine some approaches to programming with constraints, comparing the CLP(91) approach with that of other constraint languages. In Section 4 we discuss software for the analysis of circuits. The two major examples are steady-state analysis of RLC circuits, and synthesis and analysis of transistor amplifier circuits. Section 5 describes software for the simulation of digital filtering circuits. Finally, in Section 6 we discuss the analysis of electromagnetic fields.

2. The CLP(91) Language and System

We give a very brief description of the CLP(91) language. For more details, see [6, 7, 9]. Arithmetic terms are constructed from real constants, variables, +, −, *, /, sin, cos, tan, pow where all of these symbols have the usual meanings and parentheses may be used in the usual way to resolve ambiguity. Constraints are built up from arithmetic terms using the binary relation symbols =, ≥, ≤, >, <. For example 1.234 + X < Y and X + Y*(sin(T) − 1) = tan(Z) are constraints. Any variable that appears in an arithmetic term is said to be an arithmetic variable, and cannot take a non-arithmetic value. Terms are constructed from variables, arithmetic terms and uninterpreted functors. For example, (X + Y)/4 and g(22, h(4 *(Y + X))) are terms, while f(X) + g(X) and a − 3 are not. Atoms are of the form p(t1, . . . , tn) where p is an n-ary predicate symbol and the t, are terms. Finally a program is a finite collection of rules, each of the form:

\[ H :- B_1, \ldots, B_m \]

where H is an atom and each Bi is either a constraint or an atom.

For example, the following single-rule program models the relationship between two complex numbers and their product, where each complex number, X + iY, is represented as c(X, Y).

\[
c_{\text{mult}}(c(R1, I1), \ c(R2, I2), \ c(R3, I3)) :- \\
\text{R3} = R1 \* R2 \ - \ I1 \* I2, \\
\text{I3} = R1 \* I2 \ + \ R2 \* I1.
\]

The operational model for CLP(91) is an obvious generalization of that for PROLOG. Briefly, a goal consists of a number of solvable constraints and a number of atoms. A derivation step consists of first matching a selected atom with the head of an input rule (whose variables have been suitably renamed). This matching in