On the Uniform Representation of Mathematical Data Structures *

Carla Limongelli and Marco Temperini

Dipartimento Informatica e Sistemistica - Università "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
e-mail: {limongel,maxt e }@disco1.ing.uniroma1.it

Abstract. Topics about the integration of the numeric and symbolic computation paradigms are discussed. Mainly an approach through a uniform representation of numbers and symbols is presented, that allows for the application of algebraic algorithms to numeric problems. The \( p \)-adic construction is the basis of the unifying representation environment. An integrated version of the Hensel algorithm is presented, which is able to perform symbolic and numeric computations over instances of ground (concrete) and parametric structures, and symbolic computations over instances of abstract structures. Examples are provided to show how the approach outlined and the proposed implementation can treat both cases of symbolic and numeric computations. In the numeric case it is shown that the proposed extension of the Hensel Algorithm can allow for the exact manipulation of numbers. Moreover, such an extension avoids the use of simplification algorithms, since the computed results are already in simplified form.

1 Introduction

Systems for symbolic mathematics are based on the availability of powerful methods and techniques, which have been developed for numeric computation, symbolic and algebraic computation and automated deduction. But those different computing paradigms really work independently in such systems. Each of them represents an individual computing environment, while they are not integrated to support a uniform environment for computation. The problem of the integration of numeric and symbolic computation is still open [2, 8].

In this paper, a possible solution to this problem is considered, taking into account the well-known need for abstraction in the definition of mathematical data structures. Mainly we focus on the possibility of using a uniform representation of those structures and of extending the use of algebraic algorithms to numerical settings. Such a representation is based on Truncated Power Series. It allows, through the \( p \)-adic arithmetic tool, for the integration of the numeric and symbolic capabilities of algorithms defined at high level of abstraction. In

* This work has been partially supported by Progetto Finalizzato “Sistemi Informatici e Calcolo Parallelo” of CNR under grant n. 92.01604.69.
this work we describe how a uniform representation for functions and numbers, based on the $p$-adic construction, can be used in order to obtain a homogeneous approach to symbolic and numeric computation.

The well known Hensel algorithm is extended to cover both symbolic and numeric operations: polynomial factorization, finding of the $n$-th root of an analytical function, root finding of polynomial equations, and $p$-adic expansion of an algebraic number are examples of operations that can be performed by this extension. The extended algorithm works on a common $p$-adic representation of algebraic numbers and functions. In the numeric case the computations are performed by the $p$-adic arithmetic. Actually, numeric computations by the extended Hensel algorithm provide the exact representation of the resulting algebraic numbers, by means of their $p$-adic expansion.

2 A proposal of integration

The design of new generation symbolic computation systems [9, 13] has benefited from the use of abstraction [7] and, later, of object-oriented programming [1]. The use of abstraction and of hierarchical definition of mathematical data structures can improve the expressive power of symbolic computation systems and their characteristics of correctness. The abstraction supported by object-oriented techniques can be seen as a primary feature of a system able to support the integration of numeric and symbolic computation.

In [10] the approach through abstraction to the classification of algebraic structures has been described; an appropriate inheritance mechanism has been defined for their implementation by an object-oriented paradigm. Abstract structures (like ring or field algebraic structures), Parametric structures (like matrix of ring elements, or polynomials over ring coefficients) and Ground structures (like integer or real) can be distinguished by the different completions of their algebraic definition. Each parametric or ground structure can be derived from the appropriate abstract structure through inheritance of features. The mechanism of strict inheritance plus redefinition also allows one structure to be derived from another.

Actually, a uniform computing environment is obtained once each algebraic structure is implemented by its appropriate class template (namely abstract, parametric or ground classes). The behavioural compatibility of data structures is fixed by their inheritance relationships. The algorithms which have been included in a (class) structure $S$, designed to be functional attributes for the instances of $S$, will be available on every other structure connected to $S$ by inheritance. Hence, some kind of "implicit coercion" has the effect of polymorphism induced by inheritance.

But this can still be unsatisfactory. Indeed, it is possible that ground classes of the same abstract type may have incompatible representation. This would happen when two classes have no direct inheritance connection (one is not derived from the other), even if they are derived from the same abstract class. For example, numeric computations involving both real and complex numbers