The uniform representation of mathematical objects by truncated power series. *

Carla Limongelli and Marco Temperini

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 (Caviness, 1986; Limongelli & Miola, 1990; Mascari & Miola, 1986).

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.

With this aim in mind, we will present a classification of formal structures, defined in an algebraic context. In such a classification a fundamental role is played by the methods defined on algebraic structures: the algorithms for the manipulation of abstract structures can be included in the definition of the structures themselves. They acts as polymorphic operators and can be applied on every structure which is compatible with the structure in which the operator is defined, and which is defined at a lower level of abstraction.

This approach is based on abstraction techniques and on the Object-Oriented paradigm. Hence methods are considered from a twofold viewpoint: first they are interpreted as characterizing attributes of the entity on which they are defined; second they can provide the algorithms that may be not characterizing attributes, but enrich the definition of the structure in which they are given.

*This work has been partially supported by Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" of CNR under grant n. 92.01604.69.

Related to the former case, for example, we can cite the operators + and $*$, defined in a ring structure; these methods are necessary to the definition of this structure. Related to the latter case, we can consider for example the Hensel lifting algorithm (see Sec. 2); this algorithm is joined to the definition of the most abstract structure on which it acts. In such a way this algorithm can be applied on any instance of the abstract structure and on all descendant structures.

Moreover, this classification allows us to identify the algebraic structure from which it is possible to specialize numbers and functions. We define $\mathbb{D}[[x]]$ as the truncated power series with coefficients defined over a domain $\mathbb{D}$ and $\mathbb{D}[x]$ an univariate polynomial defined over $\mathbb{D}$. It is well known that every analytic function can be represented as a succession of truncated power series. Moreover we know that $\mathbb{D}[[x]]$ is isomorphic to $\mathbb{D}[x]$.

The proposed representation for functions and numbers is based on Truncated Power Series. It allows, through the p-adic arithmetic tool (Colagrossi, Limongelli, & Miola, 1994), for the integration of the numeric and symbolic capabilities of algorithms defined at high level of abstraction. In 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, 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 Classification of mathematical objects

The specification of an algebraic structure is defined through the notion of signature. A signature Σ consists of a triple $< S, O, P >$: S is a disjoint union of sets (sorts) on which the corresponding algebra is defined; O is a proper set of operators (function symbols) such that each operator is associated to a mapping type $s_1 \dots s_k \rightarrow s$ where $s_1 ... s_k$ and s belong to S; P is a set of properties defining the relationships among operators, expressed through a logic formalism, e.g. by formulae of FOL (as in (Wirsing & Broy, 1989)), or expressions of Algorithmic Logic (as in (Limongelli, Mele, Regio, & Temperini, 1990)). The choice of an axiomatic specification mechanism is taken in order to satisfy the design requirements of a symbolic computation system (Limongelli, Miola, & Temperini, 1992), as specified in the introduction.

In the following a classification of the algebraic structures is proposed, defining three planes for their static definition. This subdivision will point out the different specification requirements of structures laying on different planes, leading to a higher level of correctness in their treatment. In the sequel the term symbolic computation will be used for any computation involving the manipulation of properties of a structure, without worrying about the abstraction level of its definition. The term numeric computation will refer to computation that acts over structures that have completely specified sorts. Moreover it will be shown how OOP methodology naturally fits into the specification scheme exposed above.

2.1 Abstract structures

The definition of algebraic structures is carried out by a given scheme supporting the declaration of sorts, operators and their related axioms. The first step towards the definition of such a scheme should include mechanisms for the classification of the algebraic structures that are found at the highest level of abstraction as stated in classical algebra. In this way a hierarchy of structures (see Figure 1) is established. The elements of this hierarchy will be named *abstract structures*. The hierarchy specializes itself by adding new properties and operations, starting from a very general structure (e.g. semigroup). From now on the single arrows in the figures represent inheritance relations: the direction of an arrow is intended as starting from the "inherited" class (*parent*) towards the "inheriting" one (*successor*).
At a general specification level, as in classical algebra, sorts are not specified, while only operations and axioms, related to the presentation of an algebraic structure, are defined. Abstract structures cannot be employed for numeric computations: only symbolic computations can be carried out on them. In fact numeric computations are performed only when the sorts of the involved algebraic structures are completely specified. In Figure 2 the specification of the algebraic abstract structures of Figure 1 is shown. In this specification the possibility of obtaining the definition of a structure taking sorts, operators and properties **from** another definition, possibly by **redefining** some denotations, is exploited.

In (Limongelli & Temperini, 1992) it has been shown how this characteristic, which is classical in algebra, will match directly with inheritance OOP.
It should be noted how in this specification the use of the inheritance mechanism by means of "**from**" construction makes it possible to obtain shorter but equally expressive definitions.

2.2 Parametric structures

In order to actually use the specified abstract structures, the need arises for more specialized properties and operations. A first step towards this objective

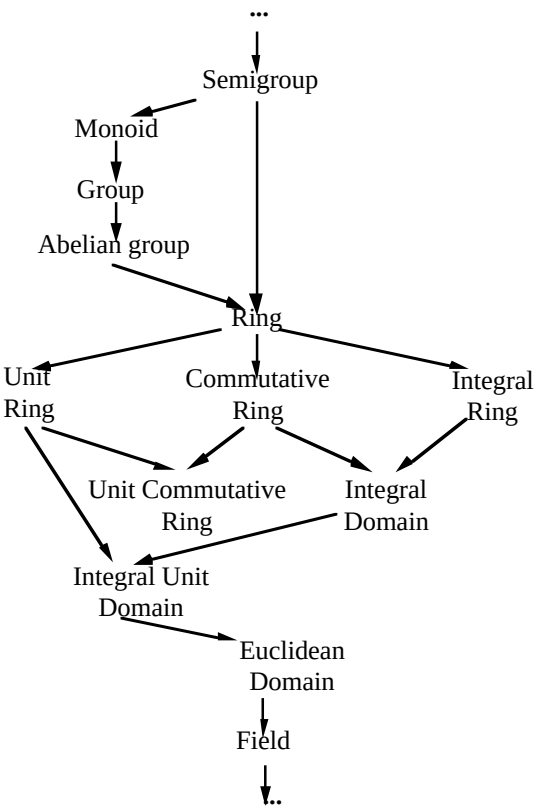

Figure 1: Classification of abstract algebraic structures.

Semigroup	
sorts	$\$$
operations	$op : \$ \times \$ \rightarrow \$$
properties	$\forall a, b, c \in \$,\ (a\ op\ b)\ op\ c = a\ op\ (b\ op\ c)$
Monoid	
from	*Semigroup*
operations	$1 :\ \rightarrow \$$
properties	$a \in \$,\ 1\ op\ a\ =\ a\ op\ 1\ =\ a$
Group	
from	*Monoid*
properties	$\forall a \in \$, \exists a^{-1} \in \$:\ a\ op\ a^{-1} = a^{-1}\ op\ a = 1$
Abelian Group	
from	*Group*
properties	$a, b \in \$,\ a\ op\ b\ =\ b\ op\ a$
Ring	
from	*Abelian Group* **redefining** *op* **as** $+$, 1 **as** 0, a^{-1}as $-a$
from	*Semigroup* **redefining** *op* **as** $*$
properties	$\forall a, b, c \in \$, a * (b + c) = a * b + a * c$
	$\forall a, b, c \in \$, (a + b) * c = a * c + b * c$
Unit Ring	
from	*Ring*
operations	$1 :\ \rightarrow \$$
properties	$\forall a \in \$,\ a * 1 = 1 * a = a$
Commutative Ring	
from	*Ring*
properties	$\forall a, b\ \in \$,\ a * b = b * a$
Comm. Unit Ring	
from	*Unit Ring*
from	*Commutative Ring*
Integral Ring	
from	*Ring*
properties	$\forall a, b\ \in \$,\ a * b = b * a = 0\ \iff\ a = 0 \vee b = 0$
Integral Domain	
from	*Commutative Unit Ring*
from	*Integral Ring*
Euclidean Domain	
from	*Integral Ring*
operations	$div : \$ \times \$ \rightarrow \$,\ mod : \$ \times \$ \rightarrow \$$
properties	$\forall a, b, c \in \$, \exists c, r \in \$:\ (a = c * b + r) \Rightarrow$
	$(a\ div\ b = c) \wedge (a\ mod\ b = r)$
Field	
from	*Integral Domain*
properties	$\forall a, \in \$,\ a \neq 0,\ \exists\ a^{-1} :\ a * a^{-1} = a^{-1} * a = 1$

Figure 2: Abstract structure specification.

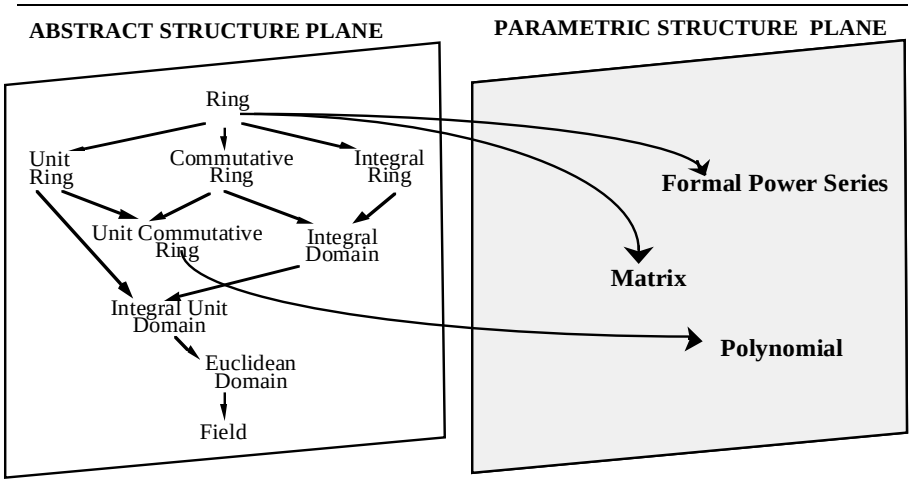

Figure 3: Example of parametric structure.

is the specification of an intermediate level of specialization, where there are essentially two requirements:

- the specialization of the sorts which is accomplished by the specification of sorts containing parameters, i.e. the complete specification of sorts depends on the specification of other sorts;
- the complete semantic definition of the operations, i.e. the methods which are characteristic of these structures.

These structures will be *named parametric structures*. Matrices, Polynomials and Formal Power Series are examples of parametric structures. The correspondence with the related abstract structures is shown in Figure 3. Here, for the sake of clarity, a distinction among abstract and parametric structures is outlined by placing these structures on different planes: an *abstract plane* and a *parametric plane* respectively.
Let us consider the case of the matrix structure. A general square matrix of order n (n is a natural number), is an element of the set

$$\mathrm{MAT} = \{\{a_{i,j}\}_{i,j \in I_n}, a_{i,j} \in \$\},$$

where:

$$I_n = i \in \mathrm{NAT} : 1 \leq i \leq n$$

and NAT is the sort of natural numbers.
The use of the **from** construction with the specification "**on** MAT" means that the operations (+ and *), inherited from the Ring structure, are applied to the

Matrix	
sorts	NAT, I_n, \$, *Ring*
operations	**from** *Ring* **on** MAT **redefining** $+$ as $+_{\mathrm{MAT}}$ and $*$ as $*_{\mathrm{MAT}}$
	PUT: MAT×NAT×NAT×\$ → MAT
	GET: MAT×NAT×NAT→ \$
properties	$\forall i,j \in I_n,\ \mathrm{GET}(0_{\mathrm{MAT}},i,j) = 0_{\$}$;
	$\forall M \in \mathrm{MAT},\ \forall i,j \in I_n,\ \forall e \in \$,\ \mathrm{GET}(\mathrm{PUT}(M,i,j,e),i,j) = e$;
	$\forall M_1, M_2 \in \mathrm{MAT}$,
	$M_1 +_{\mathrm{MAT}} M_2 = \{\mathrm{GET}(M_1,i,j) +_{\$} \mathrm{GET}(M_2,i,j)\}_{i,j,\in I_n}$;
	$\forall M_1, M_2 \in \mathrm{MAT}$,
	$M_1 *_{\mathrm{MAT}} M_2 = \{\mathrm{GET}(M_1,i,1) *_{\$} \mathrm{GET}(M_2,1,j) +_{\$} \ldots +_{\$}$
	$+_{\$}\ \mathrm{GET}(M_1,i,n) *_{\$} \mathrm{GET}(M_2,n,j)\}_{i,j,\in I_n}$;

Figure 4: Matrix structure specification.

parametric sort MAT: they will be denoted respectively as $+_{\mathrm{MAT}}$ and $*_{\mathrm{MAT}}$. In the definition of Figure 4.4, only a partial specification of matrix coefficients is given by the "parameter sort" \$ (so the coefficients are elements of a Ring structure in which 0_{MAT} is the null element for $+_{\mathrm{MAT}}$; $0_{\$}$ represents the null element of \$).

2.3 Ground structures

When sorts are completely specified, the resulting structure allows a direct variable instantiation (in this case the structure will be named *ground*). Therefore it can be used both for symbolic and numeric computations. Moreover it can be noted that new computational methods and properties can be added into the definition of a ground structure, in order to reach a higher level of efficiency, both for computation and deduction (see Figure 5).

The characteristics of the structures defined above are summarized as follows:

- *Abstract structures*: classical algebraic structures described by inherited properties. Sorts are not specified, only symbolic computations can be performed;
- *Parametric structures*: (e.g: matrix structure) enrich the definition of abstract structures by partial (parameterized) sorts, and by additional operations and properties;
- *Ground structures*: (e.g.: integers) completely specified; both symbolic and numeric computations are allowed.

The *definition space* is defined by the three previously described planes. Here two different levels of hierarchy can be imagined: a first level is limited to the plane of abstract structures, and defines their hierarchy; a second interplanar

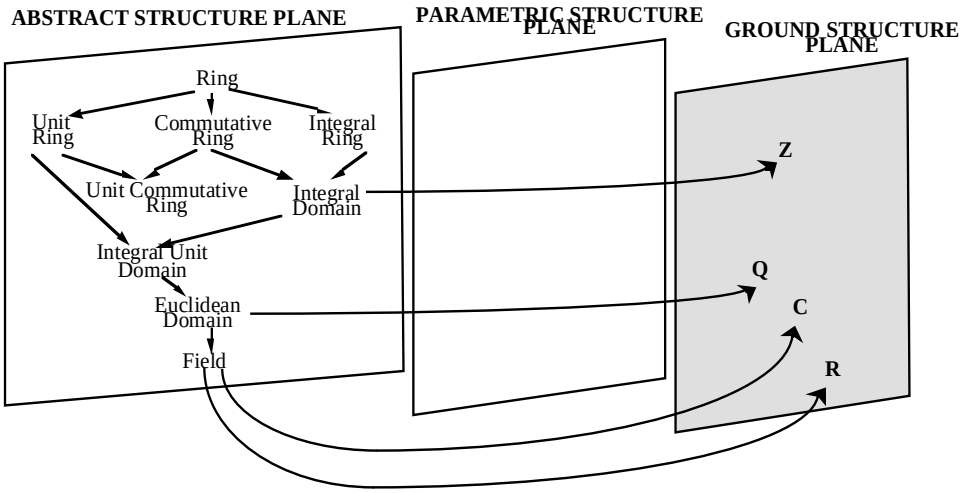

Figure 5: Plane of ground structures.

level acts respectively between the abstract and parametric structure planes and between abstract and ground structure planes. The *execution space* takes place out of the definition planes; that is the space where the structures are instantiated into algebraic objects. This space represents the connection between a single object and its generating structure (see Figure 6). In this figure the double arrows stand for object instanciation.

For example an integer number is joined with its ground structure $\mathbb{Z}$, a matrix of integers is joined both with a Matrix parametric structure and $\mathbb{Z}$ ground structure. A matrix of integer polynomials is joined with the structure $\mathbb{M}(\mathbb{P}(\mathbb{Z}))$.

It must be noted that an abstract structure is never instantiated. The chain of inheritance acts only on the abstract structure plane. On the other hand the elements that belong respectively to the parametric structure plane or to the ground structure plane, are not themselves related. Their correlations are established by transversal inheritance from the abstract plane to the parametric and ground planes.

3 Method abstractions

In the previous section it has been seen that algebraic structures present some method definitions. In general these methods specify the semantics of fundamental operations in the structure (e.g. the matrix addition method in Matrix parametric structure, or the *div* and *mod* methods defined in Euclidean Domain

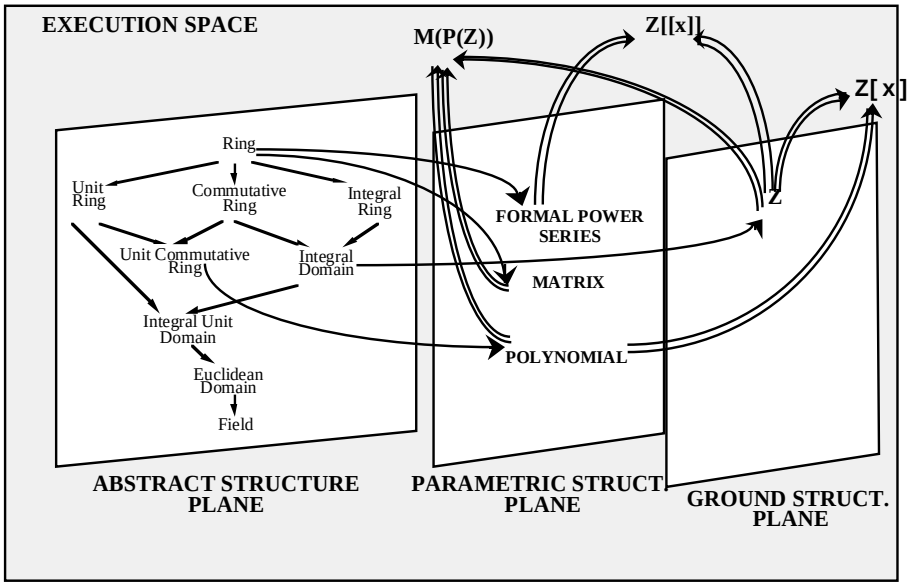

Figure 6: The execution space

abstract structure).

Now it will be shown how the enrichment of the definition of algebraic structures is possible, thus giving rise to abstract structures in which computing methods become available components. In fact as far as method abstraction is concerned, the inheritance allows us to use the code provided for a higher structure in all its subdomains, without any redefinition, as (Limongelli et al., 1990) and (Limongelli & Temperini, 1992) show.

The approach to numeric computation through truncated p-adic arithmetic (Hensel, 1908) has been widely analyzed (Gregory & Krishnamurthy, 1984; Colagrossi et al., 1994). As a matter of fact, the truncated p-adic representation can be viewed as the truncation of the p-adic series representing the rational number. The degree r, chosen for the approximation, depends on the characteristics of the problem. The same approach can be used to treat problems in abstract structures. In particular it is possible to exploit the uniformity between the truncated p-adic expansion of a rational number (Hensel code) and the p-adic polynomial expansion by power series. The theory and construction of Hensel Code for arithmetic of a rational function over a finite field is analogous to the truncated p-adic arithmetic. So a rational polynomial function can be represented in the form of Hensel code like the Example 3.1 shows.

Example 3.1 *Given the univariate polynomial* $P(x) = x^3 + 3x^2 - 2x + 1$

we fix the base x *and the approximation* $r = 4$*, the related Hensel code is:* $H(x, 4, p(x)) = (1\ \ -2\ 3\ 1,\ 0)$

Given the bivariate polynomial $P_1(x, y) = yx^2 - 2yx + 2y + y^2x - y^2 - 3$ *we fix the base* $(x - 1)$ *and the approximation* $r = 4$*, the related Hensel Code is:* $H(x - 1, 4, P_1(x, y)) = (y^2\ y\ y - 3,\ 0)$ □

The means to operate on such uniformly represented structures are provided by a variety of algebraic techniques, based on the application of the Extended Euclidean Algorithm (EEA henceforth), as shown in (Limongelli & Miola, 1990). The possibility of a unifying strategy used to treat problems by approximated p-adic construction methods and truncated p-adic arithmetic is confirmed, since both are based on the same algebraic properties of the EEA. As it is well known, the approximated p-adic construction method is founded on the following computational steps:

- starting from an appropriate initial approximation;
- computes the first order Taylor series expansion;
- solve the obtained equation;
- find an update of the solution.

The following theorem states the convergence of linear p-adic liftingOB.

Theorem 3.1 (Abstract Linear Lifting) *Let* I *be a finitely generated ideal in* R*,* $f_1, \ldots, f_n \in R[x_1, \ldots, x_r]$*,* $r \geq 1$*,* $a_1, \ldots, a_r \in R$ *with*

$$f_i(a_1, \ldots, a_r) \equiv 0 \ mod\ I, \text{ with } i = 1, \ldots, n.$$

Moreover let

$$U = (u_{i,j}),\ i = 1, \ldots, n, \quad j = 1, \ldots, r, \quad \text{with } u_{i,j} = \frac{\partial f_i}{\partial x_j}(a_1, \ldots, a_r) \in R$$

(U *is the Jacobian matrix of* $f_1, \ldots, f_n$*, evaluated at* $a_1, \ldots, a_r$*). Assume that* U *is invertible mod* I*.*
Then, for any positive integer t*, there exist* $a_1^{(t)}, \ldots, a_r^{(t)} \in R$*, such that*

$$f_i(a_1^{(t)}, \ldots, a_r^{(t)}) \equiv 0 \ mod\ I^t, \quad i = 1, \ldots, n$$

and

$$a_j^{(t)} \equiv a_j \ mod\ I, \quad j = 1, \ldots, r.$$

PROOF: The proof is given by induction on t. See (Lauer, 1983). □

The Hensel lemma is a specialization of the above-cited theorem. The lemma is specialized by assuming $n = 1$, $r = 2$, $R = \mathbb{Z}[x]$ and $p = I$, p being a prime number. Thus, let us assume $A_1, A_2, C \in \mathbb{Z}[x]$. Let us rename A_1 and A_2, G and H respectively and consider $C = F$. Let us also suppose that

$$G \cdot H \equiv F \ mod\ p,$$

and that G and F are relatively prime *mod p*. Then, for any positive integer t there exist $A_1^{(t)}, A_2^{(t)} \in \mathbb{Z}_{p^t}[x]$, such that

$$G^{(t)} \equiv G \ mod\ p \ \text{ and } \ H^{(t)} \equiv H \ mod\ p.$$

Moreover, since $n = 1$, we can rename f_1, Φ. Thus we consider

$$\Phi(G, H) = G \cdot H \equiv \ 0 \ mod\ p.$$

In such case the partial derivatives are:

$$u_{1,1} = \frac{\partial \Phi}{\partial G}(G, H) = H,$$

$$u_{1,2} = \frac{\partial \Phi}{\partial H}(G, H) = G$$

and the matrix U (1×2) is equal to (H, G) and is invertible *mod p*, if and only if G and H are relatively prime *mod p*.

If $G^{(t)}, H^{(t)}$ satisfies

$$G^{(t)} H^{(t)} - F \equiv 0 \ mod\ p^t,$$

and if we consider

$$G^{(t+1)} = G^{(t)} + Ap^t, \quad H^{(t+1)} = H^{(t)} + Bp^t,$$

by solving the two equation in the indeterminates A and B, we obtain:

$$A \cdot H + B \cdot G \equiv 0 \ mod\ p.$$

The specialization described above can be reformulated by the following lemma. This lemma results from THEOREM 3.1 renaming t as k, $G^{(t)} and H^{(t)}$ as G_k, H_k respectively.

Lemma 3.2 (HENSEL LEMMA)*: Given a prime number $p \in \mathbb{Z}$, and given $F(x) \in \mathbb{Z}[x]$, if $G_1(x)$ and $H_1(x)$ are two polynomials relatively prime over $\mathbb{Z}_p[x]$, such that*

$$F(x) \equiv G_1(x) \cdot H_1(x) \ mod\ p,$$

then, for any integer $k \geq 1$ there exist two polynomials $G_k(x), H_k(x) \in \mathbb{Z}_{p^k}[x]$ such that

$$F(x) \equiv G_k(x) \cdot H_k(x) \ mod \ p^k$$

where

$$G_k \equiv G_1(x) \ mod \ p, \quad H_k \equiv H_1(x) \ mod \ p.$$

PROOF: See (Yun, 1974). □

This lemma ensures that, starting from an appropriate initial approximation, we can obtain higher order approximations.

Following Newton's idea (Lipson, 1976), the Hensel method gives an iterative mechanism to solve equations of the following type: $\Phi(G, H) = 0$ where is

$$\Phi : \mathbb{D}[x] \times \mathbb{D}[x] \rightarrow \mathbb{D}[x]$$

with $\mathbb{D}$ an abstract Euclidean Domain.

The generality of this method makes it an apt tool to deal with approximated representations in a uniform way and in the same algebraic context. The Hensel algorithm can perform different computational methods according to different actual specializations of the parameters G and H (i.e. different forms of the equation and of the initial approximations). The following scheme provides some examples:

(i)	Factorization:	$\Phi(G, H) = F - GH = 0;$
(ii)	n-th root of a function G:	$\Phi(G) = F - G^n = 0;$
(iii)	Legendre polynomial:	$\Phi(G, t) = (1 - 2x + t^2)G^2 - 1 = 0;$
(iv)	Newton's method:	$\Phi(x) = F = 0.$

Starting from the initial approximations, $G_1(x)$ and $H_1(x)$ such that $\Phi(G_1, H_1) \equiv 0 \ mod \ I$, at each step k the approximate solution G_k, H_k can be obtained, such that $\Phi(G_k, H_k) \equiv 0 \ mod \ I^k$, where I is an ideal in $\mathbb{D}[x]$. Then the bivariate Taylor series expansion of Φ at the point (G_k, H_k), is computed as:

$$\Phi(G, H) = \Phi(G_k, H_k) + (G - G_k)\frac{\partial \Phi(G_k, H_k)}{\partial G} + (H - H_k)\frac{\partial \Phi(G_k, H_k)}{\partial H}.$$

Dividing by p^k and assuming

$$C = \Phi(G_k, H_k), \quad A_k = -\frac{\partial \Phi(G_k, H_k)}{\partial G}, \quad B_k = -\frac{\partial \Phi(G_k, H_k)}{\partial H},$$

$$\frac{G - G_k}{p^k} = \Delta G_k, \ \frac{H - H_k}{p^k} = \Delta H_k,$$

we solve the following equation

$$A_k \Delta G_k + B_k \Delta H_k \equiv \frac{C}{p^k} \; mod \; p \tag{1}$$

in order to find the $(k+1)$-th updates

$$G_{k+1} = G_k + p^k B_k, \quad H_{k+1} = H_k + p^k A_k.$$

Here G_i (respectively H_i) stands for the sum of the first i terms in the p-adic series expansion of G (respectively H).
A more detailed description of the Hensel algorithm is reported in (Limongelli & Temperini, 1992). At each step we rebuild the p-adic coefficients related to series expansion of the solution. We must note that the hypothesis of the relative primality of G_1 and H_1 in the previous lemma is necessary only to solve the above Diophantine equation (1). When F has a specialized form, like in (ii), (iii) and (iv), the equation to be solved is just a modular univariate equation. In these cases the hypothesis of relative primality is no longer needed. A unifying view of different data structures and algebraic methods comes out by their treatment through a generalized p-adic representation. The Figure 7 shows a set of application areas of the Hensel algorithm, distinguishing the numeric from the symbolic cases.

3.1 Placing Hensel method at the right specification level

Once the treatment of numeric data structures is founded on the p-adic representation and arithmetic, it is possible to devise a hierarchy of data structures, which represent the computational domain of symbolic (and numeric) computations. An approach, based on abstraction, to the classification of algebraic structures has been described in (Limongelli & Temperini, 1992). 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 completion of their algebraic definition. Among all these structures, the mechanism of strict inheritance plus redefinition allows to derive each one by another. Following this approach, we can locate in the appropriate planes the algebraic methods and structures, which have been cited previously as main devices in the unified view of symbolic and numeric computations. The first structure is the truncated power series. It is straightforwardly a parametric structure whose coefficients are defined over an Euclidean domain. In Fig. 8 a specification of the Hensel algorithm is given. Notice that $+_{\mathbb{ID}}$, $=_{\mathbb{ID}}$, $0_{\mathbb{ID}}$, represent, respectively, the addition operation, the equality relation and the zero element in the domain $\mathbb{ID}$. The function call $EVAL(x = xk, H = Hk, G = Gk, F)$, evaluates Φ in the point (G_k, H_k, x_k). The function $SOLVE$ computes the expansion in Taylor series of a function F, and the related zeros. In particular these zeros are computed

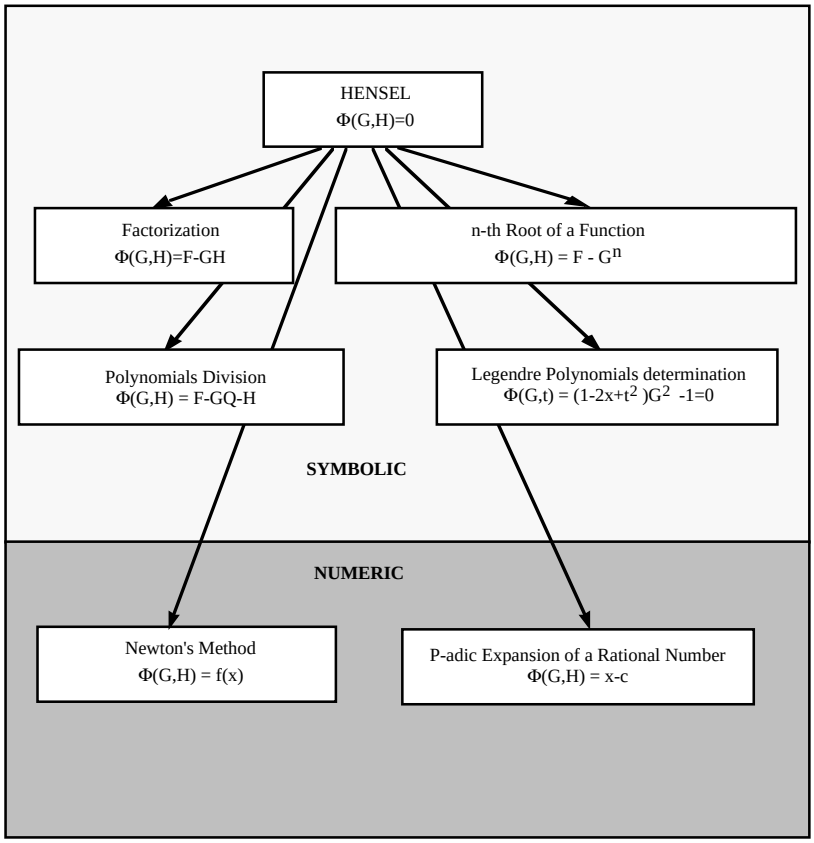

Figure 7: Possible specializations of the Hensel method.

w.r.t. either the symbolic variables (G, H) or the numeric variable (x) depending on the configuration of the given input parameters. Hence the algorithm is able to compute either symbolic or numeric solutions of Φ.

From the previous description we can see that the Hensel method takes polynomials as input and returns either a truncated power series or a Hensel code. It applies the EEA (in $SOLVE$, in order to solve either the Diophantine equation, or the modular equation). It should be located where all the properties of an Euclidean Domain are available (see Fig. 9). Hence the Euclidean Domain is the structure of highest level of abstraction, where the Hensel method can be defined. Let us note that, in this case, the algorithm is not a characterizing attribute of the abstract structure of Euclidean Domain; in fact it is added to the definition, just enriching it.

The p-adic arithmetic is represented by the Hensel-Code which is conveniently located in the plane of ground structures. Hensel codes are particular instances of truncated power series whose coefficients are defined over $\mathbb{Z}_p$ (p a prime number).

4 The integration of symbolic and numeric computation: a case study.

In this section we propose a first implementation of the "integrated" algorithm of Fig. 8. The programming language included in the system MAPLE (Char, Fee, Geddes, Gonnet, Monagan, & Watt, 1986) has been used. The procedure `hgen` implements the plain Hensel algorithm. In it the procedure `hsolve`, which we do not show here, performs the selection between numeric and symbolic case. Actually, as in every other existing system for symbolic computation, the specification of data structure is not allowed as described in Sect. 3. So we had to design special structures in order to simulate organization of abstract parametric and ground planes. In the rest of this section we will provide examples. Firstly we show the specialization of the Hensel algorithm in the symbolic case.

Let us show some examples of applications.

Example 4.1 (FACTORIZATION) Given the following univariate polynomial;

$$F(x) = x^5 + 12x^4 - 22x^3 - 163x^2 + 309x - 119$$

we want to compute its factorization in $\mathbb{Z}_5[x]$. Following the steps previously described we obtain:

1: starting from the initial approximations

$$G_1(x) = x^3 + 2;\ H_1(x) = x^2 + 2x - 2.$$

EXTENDED HENSEL ALGORITHM

Input: a: a function $\Phi(G(x), H(x))$, where $G(x),\ H(x) \in \mathbb{ID}[x]$, and x a variable;
r: order of the solution;
n: possible exponent of G;
$I \in \mathbb{ID}[x]$: base;
G_0, H_O, x_0: initial approximations;

Output: $G_k,\ H_k,\ x_k$, such that $\Phi(G_k(x_k), H_k(x_k)) \equiv 0\ mod\ I^k$;

Begin
$k := 1$;
$\Phi := F -_{\mathbb{ID}} G^n \cdot H$;
$sol := [x_0, G_0, H_0]$;
$a[1] := sol$;
$x_k := x_0;\ H_k := H_0;\ G_k := G_0$;
$C := EVAL(x = x_k,\ H = H_k,\ G = G_k, \Phi)$;
$delta := [0_{\mathbb{ID}}, 0_{\mathbb{ID}}, 0_{\mathbb{ID}}]$;
While$(k \leq r) \wedge (C \neq_{\mathbb{ID}} 0_{\mathbb{ID}})$
do
$SOLVE(F, G, x, x_0, G_0, H_0, \Phi, C, I)$;
$a[k+1] := delta$;
$sol := [delta[1] \cdot I^k +_{\mathbb{ID}} sol[1], delta[2] \cdot I^k +_{\mathbb{ID}}$;
$+ sol[2], [delta[3] \cdot I^k +_{\mathbb{ID}} sol[3]$;
$x_k := sol[1];\ G_k := sol[2];\ H_k := sol[3]$;
$k := k + 1$;
$C := EVAL(x = x_k,\ H = H_k,\ G = G_k, \Phi)$;
od
End;

Figure 8: Specification of the extended Hensel algorithm.

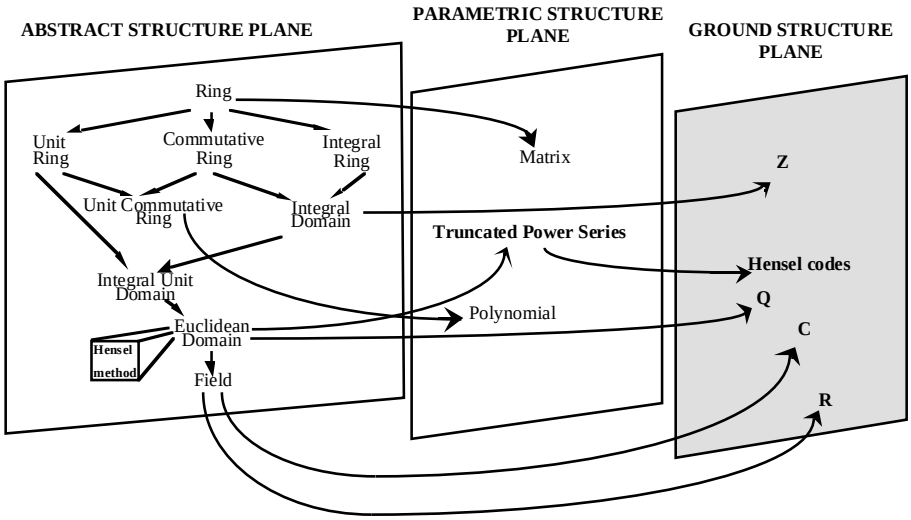

Figure 9: Specification levels.

2: we compute the first order bivariate Taylor series expansion:

$$\frac{F - G_1 \cdot H_1}{p} - (\frac{G - G_1}{p}) \cdot H_1 - (\frac{H - H_1}{p}) \cdot G_1 \equiv 0 \ mod \ p$$

3: then we solve the Diophantine equation:

$$\frac{F - G_1 \cdot H_1}{5} \equiv B_1(x^2 + 2x - 2) + A_1(2x - 1) \ mod \ 5$$

finding the values

$$B_1 = 2x - 2; \ A_1 = 2x - 1; \ A_1, B_1 \in \mathbb{Z}_5[x]$$

4: then we update the initial solutions:

$$G_2 = (x^3 + 2) + 5B_1 = x^3 + 10x - 8;$$
$$H_2 = (x^2 + 2x - 2) + 5A_1 = x^2 + 12x - 7.$$

We use this partial result to find further updates of the initial solution

3: we solve the Diophantine equation:

$$\frac{F - G_2 \cdot H_2}{5^2} \equiv B_2 \cdot H_2 + A_2 \cdot G_2 \ mod \ 5$$

```
hgen := proc(F,G,H,x,x0,g0,h0,n,p,r);
        a:=array(1..r+1);
        k:=1; FI:=F-G^ n*H;
        sol:= [x0,g0,h0]; a1:=sol;
        xk:=x0; gk:=g0; hk:=h0;
        C:=expand(subs(x=xk, G=gk, H=hk,FI));
        while (k <= r) and C<>0
              do
                 hsolve(F,G,H,x,x0,g0,h0,FI,C,p);
                 delta:=[op(1,delta) mod p, mods(op(2,delta),p);
                 mods(op(3,delta),p)];
                 a[k+1]:=delta;
                 sol:=[op(1,delta)*p^k+op(1,sol);
                 op(2,delta)*p^k+op(2,sol),op(3,delta)*p^k+op(3,sol)];
                 xk:=op(1,sol); gh:=op(2,sol); hk:=op(3,sol);
                 k:=k+1;
                 C:=expand(subs(x=xk,G=gk,H=hk,FI));
              od
        for i to k do print(a[i]); od;
        if C=0 then print('exact result'); print(sol);
               else print('sum of first',k,'coefficients:',sol);
        fi;
end;
```

Figure 10: Implementation of extended Hensel algorithm.

finding the values

$$B_2 = -x - 1;\ A_2 = 0;\ A_2, B_2 \in \mathbb{Z}_5[x]$$

4: then we update the solutions

$$G_3 = G_2 + 5^2 B_2 = x^3 + 10x - 8 + 5^2(-x - 1) = x^3 - 15x + 17 :$$

$$H_3 = H_2 \cdot 5^2 \cdot A_2 = x^2 + 12x - 7 + 5^2 \cdot 0.$$

After two steps of iteration we have found G_2 and H_2 which satisfy the following identity:

$$F(x) = (x^2 + 12x - 7) \cdot (x^3 - 15x + 17).$$

Hence G_2 e H_2 represent the exact factorization of $F(x)$.

$$G = (x^3 + 2) \cdot 5^0 + (2x - 2) \cdot 5^1 + (-x - 1) \cdot 5^2 = \sum_{i=0}^{2} B_i 5^i$$

$$H = (x^2 + 2x - 2) \cdot 5^0 + (2x - 1) \cdot 5^1 + 0 \cdot 5^2 = \sum_{i=0}^{2} A_i 5^i$$

□

Note that at each step we rebuild the p-adic coefficients related to the series expansion of the solution.

We must note that the hypotheses of primality between G_0 and H_0 in the LEMMA 3.2 are necessary only to solve the Diophantine equation (1). When F has a specialized form, as in (ii), (iii) and (iv), the equation to be solved is only a modular univariate equation. In these cases the hypothesis of primality between G_0 and H_0 are no longer needed. In particular, in the numeric case (iv) we need simpler hypotheses, while the algorithm can stay unchanged. Let us show some examples:

Example 4.2 (THE n-TH ROOT OF A POLYNOMIAL) Given the following univariate polynomial:

$$F(x) = x^4 + 6x^3 - x^2 + 30x + 25$$

we want to compute the square root of $F(x)$ as in (ii). We will solve the equation $\Phi(G) = F - G^2$. If the exact root does not exist we want to compute its approximation. Following the steps previously described we obtain:

1: starting from a suitable initial approximation:

$$\Delta_0 = G_0 = x^2 + 3x \quad e \quad H_0 = 1,$$

2: we compute the first order Taylor series expansion of $\Phi(G)$:

$$\Phi(G_0) - 2(G - G_0)\frac{\partial \Phi(G_0)}{\partial G} =$$

$$-10x^2 + 30x + 25 - 2(G - G_0)(x^2 + 3x)$$

3: we solve the equation by letting $(G - G_0) = \Delta_0$:

$$-10x^2 + 30x + 25 - 2\Delta_1(x^2 + 3x) \equiv 0 \ mod \ 5^2$$

and we find the solution $\Delta_1 = -1$

4: we update the initial approximation:

$$\Delta_0 + \Delta_1 \cdot p = x^2 + 3x + (-1)5 = x^2 + 3x - 5$$

In this case we verify that the solution obtained is the exact one:

$$(x^2 + 3x - 5)^2 = F(x)$$

□

Now we show how the specialization of the Hensel algorithm in the numeric case can provide error-free results.

Example 4.3 Finding the root of $x^2 - 2$, means to obtain the exact representation of the number $\sqrt{2}$. Let be $F(x) = x^2 - 2$.

Input:

$$F(x);\ G = 0;\ H = 0;\ p = 7; \quad G_0 = 0;\ H_0 = 0;\ x_1 = 3; \quad r = 3;$$
$$(F(3) \equiv 0 \ mod \ 7)$$
$$\Delta x_0 = 3; \quad x_1 = 3;$$

We want to compute its roots in $\mathbb{Z}_7[x]$. Following the computational steps described in the extended Hensel algorithm we obtain:

1^{st} iteration

$$F(x) = F(x_1) + (x - x_1) \cdot F'(x_1)$$
$$\frac{F(3)}{7} + \frac{x-3}{7} \cdot F'(3) \equiv 0 \ mod \ 7$$
$$1 + \Delta x_1 \cdot 6 \equiv 0 \ mod \ 7 \quad \Delta x_1 = 1, \quad x_2 = 3 + 1 \cdot 7^1 = 10$$

2^{nd} iteration

$$\frac{F(10)}{7^2} + \frac{x-10}{7^2} \cdot F'(10) \equiv 0 \ mod \ 7$$
$$2 + \Delta x_2 \cdot 6 \equiv 0 \ mod \ 7 \quad \Delta x_2 = 2, \quad x_3 = 10 + 2 \cdot 7^2 = 108$$

3^{rd} iteration

$$\frac{F(108)}{7^3} + \frac{x-108}{7^3} \cdot F'(108) \equiv 0 \ mod \ 7$$
$$6 + \Delta x_2 \cdot 6 \equiv 0 \ mod \ 7 \quad \Delta x_3 = 6, \quad x_3 = 108 + 6 \cdot 7^3 \in \mathbb{Z}_{7^4}[x].$$

At the end of the iterations the algorithm gave :

$$x_4 = \sum_{i=0}^{3} \Delta x_i \cdot p^i = 3 + 1 \cdot 7 + 2 \cdot 7^2 + 6 \cdot 7^3$$

and the coefficients of this p-adic expansion (3 1 2 6) represent the mantissa of the Hensel code related to the irrational number $\sqrt{2}$ in the p-adic arithmetic. In fact, in this arithmetic, $(3\ 1\ 2\ 6,\ 0) \times (3\ 1\ 2\ 6,\ 0) = (2\ 0\ 0\ 0,\ 0)$, which is the representation of the number 2 in the integer arithmetic. □

Example 4.4 Following a procedure like in the previous example the computation of $F(x) = x^2 - 5$ is briefly shown:
Input:

$$F(x);\ G = 0;\ H = 0;\ p = 11;\quad G_0 = 0;\ H_0 = 0;\ x_1 = 7;\quad r = 3;$$

Output:

$$x_3 = \Delta x_0 \cdot p^0 + \Delta x_1 \cdot p^1 + \Delta x_2 \cdot p^2 + \Delta x_3 \cdot p^3 = 7 \cdot 11^0 + 6 \cdot 11^1 + 0 \cdot 11^2 + 6 \cdot 11^3 = 8059.$$

The coefficients of this 11-adic expansion, (7 6 0 6), represent the number $\sqrt{5}$ in the p-adic arithmetic. It is simple to verify that, in this arithmetic, is $(7\ 6\ 0\ 6,\ 0) \times (7\ 6\ 0\ 6,\ 0) = (5\ 0\ 0\ 0,\ 0)$, which is the representation of the integer number 5. □

The importance of dealing with exact results is well known: an example is the following case, involving the analytical expression of the n-th number:

$$F_n = \frac{1}{\sqrt{5}} \cdot \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right]$$

By using a p-adic representation, we are guaranteed to obtain the exact result, say for F_3, with $p = 11,\ r = 4$ and $\sqrt{5}$ represented by $(7\ 6\ 0\ 6,\ 0)$.

$$F_3 = \frac{(1\ 0\ 0\ 0,\ 0)}{(7\ 6\ 0\ 6,\ 0)} \cdot$$

$$\cdot \left[\left(\frac{(1\ 0\ 0\ 0,\ 0) + (7\ 6\ 0\ 6,\ 0)}{(2\ 0\ 0\ 0,\ 0)} \right)^3 - \left(\frac{(1\ 0\ 0\ 0,\ 0) - (7\ 6\ 0\ 6,\ 0)}{(2\ 0\ 0\ 0,\ 0)} \right)^3 \right],$$

$$F_3 = \frac{(1\ 0\ 0\ 0,\ 0)}{(7\ 6\ 0\ 6,\ 0)} \cdot \left[\left(\frac{(8\ 6\ 0\ 6,\ 0)}{(2\ 0\ 0\ 0,\ 0)} \right)^3 + \left(\frac{(6\ 6\ 0\ 6,\ 0)}{(2\ 0\ 0\ 0,\ 0)} \right)^3 \ \right],$$

$$F_3 = \frac{(1\ 0\ 0\ 0,\ 0)}{(7\ 6\ 0\ 6,\ 0)} \cdot \left[(4\ 3\ 0\ 3,\ 0)^3 + (3\ 3\ 0\ 3,\ 0)^3 \ \right]$$

where $(4\ 3\ 0\ 3,\ 0)^3 = (9\ 6\ 0\ 6,\ 0)$ and $(3\ 3\ 0\ 3,\ 0)^3 = (5\ 6\ 0\ 6,\ 0)$

$$F_3 = \frac{(1\ 0\ 0\ 0,\ 0)}{(7\ 6\ 0\ 6,\ 0)} \cdot [(9\ 6\ 0\ 6,\ 0) + (5\ 6\ 0\ 6,\ 0)\ ],$$

$$F_3 = \frac{(3\ 2\ 1\ 1,\ 0)}{(7\ 6\ 0\ 6,\ 0)} = (2\ 0\ 0\ 0,\ 0)$$

It is worthwhile to stress that the previous computation cannot be performed by the existing systems, without applications of some simplification algorithm. In fact this is not needed once the proposed p-adic approach is followed.

Other examples can be shown here, about the application of the "integrated" Hensel algorithm to numeric computation problems.

Example 4.5 In this example we find, starting from an appropriate initial approximation, a real root of the following polynomial: $f(x) = 6x^2 - 11x + 3$, the zeros of which are $3/2$ and $1/3$. Starting from $x_0 = 2$, $p = 5$, and choosing three iterations for the algorithm, at each iteration we can observe the following output:

$$\Delta x_0 = 2,\ \Delta x_1 = 3,\ \Delta x_2 = 1,\ \Delta x_3 = 3,$$

which are the first four coefficients of the 5-adic expansion of the rational number $1/3$. The other solution, $3/2$, can be found starting from $x_0 = 4$. □

5 Conclusion

By means of a uniform representation of mathematical data structures, it is possible to import into a numeric setting the precision guaranteed by the algebraic one. On this basis the integration of numeric and symbolic computation is defined also from a computational point of view.

This integration is founded on the possibility of establishing a precise isomorphism guaranteeing the validity of the approximation method for objects that can be uniformly represented through truncated power series (i.e. numbers and polynomials).

The examples of Sect. 4 show the effectiveness of extending the use of algebraic algorithms to numeric algorithms. Moreover complex numbers can be represented and treated, as the following example shows.

Example 5.1 Applying the "integrated" algorithm to the polynomial $x^2 + 1$ with $x_wild_life0 = 2$, $p = 5$, and $r = 3$, we obtain the following experimental results:

`hgen(`x^2`+1,G,H,x,2,0,0,1,5,3);`

$$[2, 0, 0]$$

$$[1, 0, 0]$$

$$[2, 0, 0]$$

$$[1, 0, 0]$$

`summation of the first, 4, coefficients with base, 5, [182, 0, 0]`
Let us note that $(2\ 1\ 2\ 1,\ 0) \times (2\ 1\ 2\ 1,\ 0) = (4\ 4\ 4\ 4,\ 0)$ which represents the rational number -1. □

Finally let us discuss some remarks about the choice of initial approximation in the algorithm that we have presented. It must be noted that some problems do still exist. Presently the initial assignments G_0, H_0 or x_0 are chosen by following a heuristic approach. For instance, in each one of the "numeric" examples which have been presented, we have chosen always the smallest base such that an initial approximation does exist.

The analysis of the computational complexity of the algorithm will complete the study of this algorithm, as soon as the authors will finish the implementation of the very first step of this algorithm, that is the choice of the suitable base p.

References

Caviness, B. (1986). Computer algebra: Past and future. *Journal of Symbolic computation*, *Vol. 2*, 217–236.

Char, B., Fee, G., Geddes, K., Gonnet, G., Monagan, B., & Watt, S. (1986). A tutorial introduction to MAPLE. *Journal of Symbolic Computation*, *Vol.2, n.2*.

Colagrossi, A., Limongelli, C., & Miola, A. (1994). p-adic arithmetic: a tool for error free computations. This Volume.

Gregory, R., & Krishnamurthy, E. (1984). *Methods and Applications of Error-Free Computation*. Springer Verlag.

Hensel, K. (1908). *Theorie der Algebraischen Zahlen*. Teubner, Leipzig-Stuttgart.

Lauer, M. (1983). Computing by homomorphic images. In Buchberger B., Collins E.G., L. R. (Ed.), *Computer Algebra - Symbolic and Algebraic Computation*.

Limongelli, C., Mele, M. B., Regio, M., & Temperini, M. (1990). Abstract specification of mathematical structures and methods. In *First International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO '90)*, Vol. 429 of *LNCS*, pp. 61–70. Springer Verlag.

Limongelli, C., & Miola, A. (1990). Abstract specification of numeric and algebraic computation methods. *Journal of information Science and Technology*.

Limongelli, C., Miola, A., & Temperini, M. (1992). Design and implementation of symbolic computation systems. In P.W.Gaffney, E. (Ed.), *IFIP TC2/WG2.5 Working Conference on Programming Environments for High Level Scientific Problem Solving*. Elsevier Science Publishers B.V. (North-Holland).

Limongelli, C., & Temperini, M. (1992). Abstract specification of structures and methods in symbolic mathematical computation. *Theoretical Computer Science*, *104*, 89–107. Elsevier Science Publisher.

Lipson, J. D. (1976). Newton's method: a great algebraic algorithm. In *1976 ACM Symposium on Symbolic and Algebraic Computation*.

Mascari, G., & Miola, A. (1986). On the integration on numeric and algebraic computations. In in Computer Science, L. N. (Ed.), *Proc. AAECC-5*, Vol. n. 307. Karlsrhue, Germany.

Wirsing, M., & Broy, M. (1989). A modular framework for specification and implementation. In *TAPSOFT '89, LNCS 351*. Springer Verlag.

Yun, D. Y. Y. (1974). *The Hensel Lemma in Algebraic manipulation*. Ph.D. thesis, Massachusetts Institute of Technology.