Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS – ENSL – INRIA – UCBL n° 5668

Univariate and Bivariate Integral Roots Certificates Based on Hensel's Lifting

Érik Martin-Dorel

École Normale Supérieure de Lyon, Arénaire,
LIP (UMR 5668 CNRS, ENSL, INRIA, UCBL)
46 allée d'Italie, 69364 Lyon Cedex 07, France
erik.martin-dorel@ens-lyon.org

March 2011

Research Report N° 2011-1

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

Univariate and Bivariate Integral Roots Certificates Based on Hensel's Lifting

Érik Martin-Dorel

École Normale Supérieure de Lyon, Arénaire,
LIP (UMR 5668 CNRS, ENSL, INRIA, UCBL)
46 allée d'Italie, 69364 Lyon Cedex 07, France
erik.martin-dorel@ens-lyon.org

March 2011

Abstract

If it is quite easy to check a given integer is a root of a given polynomial with integer coefficients, verifying we know all the integral roots of a polynomial requires a different approach. In both univariate and bivariate cases, we introduce a type of integral roots certificates and the corresponding checker specification, based on Hensel's lifting. We provide a formalization of this iterative algorithm from which we deduce a formal proof of the correctness of the checkers, with the help of the Coq proof assistant along with the SSReflect extension. The ultimate goal of this work is to provide a component that will be involved in a complete certification chain for solving the Table Maker's Dilemma in an exact way.

Keywords: Coq formal proofs, certifying algorithm, Hensel's lifting, integral roots, polynomials

Résumé

S'il est aisé de vérifier qu'un entier donné est racine d'un polynôme donné à coefficients entiers, vérifier que l'on connaît toutes les racines entières d'un polynôme requiert une approche différente. Dans le cas univarié comme bivarié, nous introduisons un type de certificats de racines entières et la spécification du vérifieur correspondant, dont le principe repose sur le relèvement de Hensel. Nous proposons une formalisation de cet algorithme itératif duquel nous déduisons une preuve formelle de correction des vérifieurs (univarié et bivarié), avec l'aide de l'assistant de preuves Coq muni de l'extension SSReflect. Le but ultime de ce travail est de fournir un composant qui sera partie intégrante d'une chaîne de certification complète pour la résolution exacte du dilemme du fabricant de tables.*

Mots-clés: Preuves formelles Coq, algorithme certifiant, relèvement de Hensel, racines entières, polynômes

*Ce travail est soutenu par le projet TaMaDi de l'ANR.

1 Introduction and Motivations

The newly revised IEEE 754–2008 standard for *floating-point* (FP) arithmetic recommends that some mathematical functions (exp, log, $x \mapsto 2^x$, ...) should be correctly rounded (roughly speaking, the system must always return the FP number nearest to the exact mathematical result of the operation). Requiring correctly rounded functions has a number of advantages: among them, it greatly improves the portability of numerical software and it allows one to design algorithms and formal proofs of software that use this requirement. To be able to design fast programs for correctly rounded functions, we must address a problem called the *Table Maker's Dilemma* (TMD) [15, chap. 12]. We need to locate, for each considered function f and for each considered FP format and rounding mode, the *hardest-to-round* (HR) points, that is, in rounding-to-nearest, what are the FP numbers x such that $f(x)$ is closest to the exact middle of two consecutive FP numbers. The naive method of finding these points (evaluating the function with large precision at each FP number) is far too impractical.

Two different algorithms have been designed to find these HR points:

- the Lefèvre algorithm — based on a variant of the Euclidean GCD algorithm — which made it possible to obtain HR cases in binary double-precision arithmetic [11,14];
- the Stehlé–Lefèvre–Zimmermann (SLZ) algorithm — based on the lattice basis reduction algorithm LLL [12] — which has a better asymptotic complexity [19,17] and might allow to get HR cases in double-extended-precision.

The processes that generate these HR points are based on complex and very long calculations (years of cumulated CPU time) that inevitably cast some doubt on the correctness of their results. In the French ANR project entitled TaMaDi, we thus undertake to fully reconsider the methods used to get HR points, with a special focus on their formal validation (by enabling our programs to generate certificates that guarantee the validity of their results).

As regards the SLZ algorithm, the LLL part is the most time consuming part of the overall algorithm so that we can treat the LLL calls as an oracle and log its results in a certificate, the verification of which no longer implies LLL calls. This has the clear benefit of avoiding to have to deal with LLL formally. Still, this approach has to be validated, we need to design "good" certificates that are of reasonable size and can be easily checked, with the goal to provide a fully verified checker for these certificates, in a similar way to what was done for primality certificates in [7,6,21].

The output of the LLL algorithm contains pairs of bivariate polynomials with integer coefficients, so that we must solve the systems

$$\begin{cases} P(x,y) = 0, \\ Q(x,y) = 0, \\ |x| \leqslant A \text{ and } |y| \leqslant B, \end{cases}$$

for each (P, Q, A, B) that is produced by LLL ($P, Q \in \mathbb{Z}[X, Y]$ and $x, y, A, B \in \mathbb{Z}$).

As mentioned in [18], there are several ways to solve this "root-finding step" of SLZ, including Hensel's lifting. But rather than just proving formally this root-finding algorithm, we introduce a type of integral roots certificates and the corresponding checker specification, based on Hensel's lifting, which thus leads to a certifying algorithm as defined in [13].

In this paper, we present a formalization within the Coq proof assistant [1,20,9] along with the SSReflect extension [4,5] of both univariate and bivariate Hensel's lifting with a uniqueness property from which we deduce a formal proof of the correctness of the integral-roots-certificates checkers.

Organization of the paper. In the upcoming Section 2 we briefly present the use of Hensel's lifting to find the integral roots of a univariate polynomial. In Section 3 we give pen-and-paper sketches of the main proofs related to Hensel's lifting in both univariate and bivariate cases, in order to highlight the required concepts for the formalization. Section 4 is devoted to the Coq formalization itself, namely a description of the Coq mechanized formal background that is common to both univariate and bivariate cases, followed by the presentation of our univariate, then bivariate integral-roots certificates with the corresponding certificate checkers. In Section 5 we discuss the advantages of the approach to design certifying algorithms we have followed, then we draw some conclusions in Section 6.

Notations. We summarize below the mathematical notations used in the sequel.

- $\mathbb{Z}$ denotes the set of signed integers;
- $\mathbb{N}$ denotes the set of nonnegative integers;
- $\mathbb{P}$ denotes the set of prime numbers;
- $\mathbb{B}$ denotes the set of booleans, that is $\mathbb{B} := \{\mathsf{true}, \mathsf{false}\}$;
- $[\![a, b]\!]$ (for $a \leqslant b$ in $\mathbb{Z}$) denotes the set of all integers k such that $a \leqslant k \leqslant b$; in other words, $[\![a, b]\!] = \mathbb{Z} \cap [a, b]$;
- $[\![a, b[\![$ (for $a \leqslant b$ in $\mathbb{Z}$) denotes the set $\mathbb{Z} \cap [a, b[= \{k \in \mathbb{Z} \mid a \leqslant k < b\}$;
- $a \bmod n$ denotes the remainder of a modulo n, taken in $[\![0, n[\![$;
- $a \equiv b \pmod{n}$ states the modular equality between a and b; in other words, $a \equiv b \pmod{n} \iff a \bmod n = b \bmod n$.
- $\mathbb{Z}[X]$ denotes the ring of univariate polynomials with coefficients in $\mathbb{Z}$;
- $\mathbb{Z}[X, Y]$ denotes the ring of bivariate polynomials on $\mathbb{Z}$, which can be viewed as the ring $(\mathbb{Z}[Y])[X]$.

2 An overview of Hensel's lifting in the univariate case

The origin of Hensel's lifting is deeply linked to the introduction of the p-adic numbers by Kurt Hensel [8], so that it is also called the p-adic Newton iteration [2], though in this work we will focus on statements that are fully expressed in (modular arithmetic on) $\mathbb{Z}$.

2.1 A uniqueness property on the modular roots of $P \in \mathbb{Z}[X]$

We briefly present the functioning of Hensel's lifting through the following lemma, which is actually a key result of the formalization at stake.

Lemma 1. *Let $P \in \mathbb{Z}[X]$ and $p \in \mathbb{P}$ that satisfies*

$$\forall z \in \mathbb{Z}, \quad P(z) \equiv 0 \pmod{p} \implies P'(z) \not\equiv 0 \pmod{p}, \tag{1}$$

where P' is the derivative of the polynomial P. If $x \in \mathbb{Z}$ is such that

$$P(x) \equiv 0 \pmod{p^{2^m}} \tag{2}$$

for a given $m \in \mathbb{N}$, then for

$$u_0 := x \bmod p, \tag{3}$$

the sequence (u_k) defined by the recurrence relation

$$\forall k \in [\![0, m[\![\quad u_{k+1} := u_k - \frac{P(u_k)}{P'(u_k)} \bmod p^{2^{k+1}} \tag{4}$$

satisfies:

$$\forall k \in [\![0, m]\!], \quad u_k = x \bmod p^{2^k}. \tag{5}$$

Remark 1. Note that Lemma 1 gives a necessary condition on each root of $P \in \mathbb{Z}[X]$ modulo p^{2^k} depending on the value of the considered root modulo p. It is somehow a uniqueness result, whereas usual results about Hensel's lifting such as the correctness theorem we can find in [2, p. 264] are existence results. We will see in Sections 4.2 and 4.3 that we specifically need this uniqueness property to prove our main theorems that deal with integral roots.

2.2 A simple bound on the univariate integral roots

We can prove the following lemma that provides more information on a possible choice of bound.

Lemma 2. *For any $P \in \mathbb{Z}[X]$, if we write P in the form $\sum_{i=\nu}^{d} a_i X^i \in \mathbb{Z}[X]$ with $a_\nu \neq 0$, we have $\forall z \in \mathbb{Z},\ P(z) = 0 \Rightarrow |z| \leqslant |a_\nu|$. Therefore $B = |a_\nu| = \left|a_{\min\{k\,:\,a_k \neq 0\}}\right|$ is a bound on the integral roots of P.*

Consequently, we can either take the generic bound given by the previous simple lemma, or choose a different bound, notably when we just want to deal with the subset of integral roots that are bounded by a specific $B > 0$.

2.3 The final doubling trick

If we can iterate Hensel's lifting (4) as many times as desired, in practice we can stop as soon as the considered power of p becomes greater than *twice* the chosen bound B on the integral roots. We explain below why this reasoning is valid.

Assuming we stop the iteration at rank k, we have the following state:

$$\exists k \in \mathbb{N}, \begin{cases} M := p^{2^k} \\ M > 2 \cdot B \\ u_k = x \pmod{M} \\ 0 \leqslant u_k < M \\ |x| \leqslant B \end{cases}$$

Thanks to the presence of the coefficient 2 above, the intervals $[0, B]$ and $[M-B, M[$ are disjoint, which facilitate the computation of x from u_k, as shown by Figure 1. More precisely, we can prove the following disjunction:

$$x \geqslant 0 \ \wedge \ x = u_k \ \wedge \ u_k \leqslant B \ \bigvee \ x < 0 \ \wedge \ x = u_k - M \ \wedge \ u_k \geqslant M - B,$$

which enables us to decide which is the sought value of x from a single test on u_k. For instance, we can just do

$$x = \begin{cases} u_k & \text{if } u_k \leqslant B \\ u_k - M & \text{if } u_k > B \end{cases}$$

to compute x.

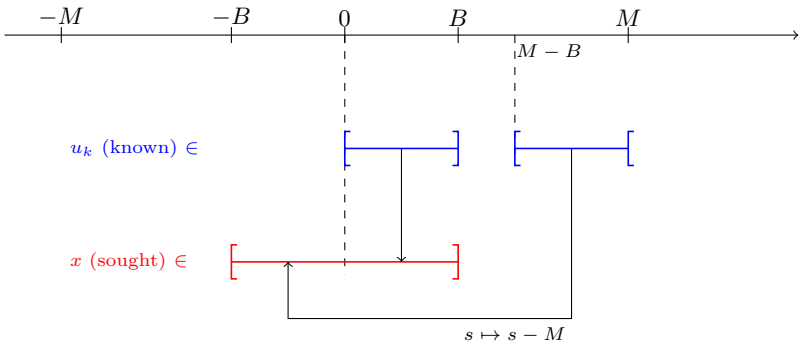

Fig. 1. How to deduce x from u_k

2.4 Example of use

Let us consider the polynomial $P(X) := X^2 - X - 42 \in \mathbb{Z}[X]$, whose roots are -6 and 7. Its derivative is $P'(X) = 2 \cdot X - 1$. Using Lemma 2, let us choose $B = 42$ as a bound. If we consider the small prime number $p = 2$, we can see at once that the roots of P modulo 2 are 0 and 1, and that the hypothesis (1) is fulfilled for both roots modulo $p = 2$. From $u_0 = 0$, we compute:

$$\begin{cases} u_1 = u_0 - P(u_0)/P'(u_0) \bmod 2^{2^1} = 0 + 42/(-1) \bmod 4 = 2; \\ u_2 = u_1 - P(u_1)/P'(u_1) \bmod 2^{2^2} = 2 + 40/3 \bmod 16 = 10; \\ u_3 = u_2 - P(u_2)/P'(u_2) \bmod 2^{2^3} = 10 - 48/19 \bmod 256 = 250, \end{cases}$$

then we stop since $256 > 2 \times B$ and the obtained root is $250 - 256 = -6$, following the reasoning of Section 2.3. Likewise, from $u_0 = 1$, we compute three iterations:

$$\begin{cases} u_1 = u_0 - P(u_0)/P'(u_0) \bmod 2^{2^1} = 1 + 42/1 \bmod 4 = 3; \\ u_2 = u_1 - P(u_1)/P'(u_1) \bmod 2^{2^2} = 1 + 36/5 \bmod 16 = 7; \\ u_3 = u_2 - P(u_2)/P'(u_2) \bmod 2^{2^3} = 7 - 0/13 \bmod 256 = 7, \end{cases}$$

and the obtained root is 7.

We notice we have found *all the roots in* $\mathbb{Z}$ of the considered univariate polynomial P, namely -6 and 7. In the sequel, we will thus focus on the formal certification of results of this kind, for both univariate and bivariate polynomials.

3 Pen-and-paper proofs

Before focusing on aspects strongly related to Coq, we present the pen-and-paper proofs that helped us to carry out the formalization at stake.

3.1 Pen-and-paper sketch of the proof of Lemma 1

Let us assume all the hypotheses of the theorem hold for $P \in \mathbb{Z}[X]$, $p \in \mathbb{P}$, $x \in \mathbb{Z}$, and (u_k) which is defined by (3) and (4). To shorten most of the following formulas, we will denote $x \bmod p^{2^k}$ by x_k for all integer k.

We want to show (5), that is to say, $\forall k \in \mathbb{N},\ k \leqslant m \Rightarrow u_k = x_k$. We prove it by induction on k:

• First, we can notice that $p = p^{2^0}$, therefore (3) means that $u_0 = x_0$, which proves the base case.

• The inductive case amounts to showing that for a given integer $k < m$ satisfying $u_k = x_k$, we have $u_{k+1} = x_{k+1}$.

To start with, we can write $x_k = x \bmod p^{2^k} = \left[x \bmod p^{2^{k+1}}\right] \bmod p^{2^k} = x_{k+1} \bmod p^{2^k}$, which implies $x_{k+1} \equiv x_k \pmod{p^{2^k}}$, hence

$$\exists \lambda \in \mathbb{Z}, \quad x_{k+1} = x_k + \lambda p^{2^k}.$$

We can now apply Taylor's theorem for the polynomial P at point x_{k+1} :

$$P(x_{k+1}) = P(x_k) + \lambda p^{2^k} P'(x_k) + \sum_{j=2}^{\deg P} \frac{\left(\lambda p^{2^k}\right)^j}{j!} P^{(j)}(x_k).$$

The first member of this equality is zero modulo $p^{2^{k+1}}$, since

$$P(x_{k+1}) = P(x \bmod p^{2^{k+1}}) \equiv P(x) = 0 \pmod{p^{2^{k+1}}}.$$

As for the last member, we can notice that

$$\forall j \geqslant 2,\ \left(\lambda p^{2^k}\right)^j = \left[\lambda^j \cdot \left(p^{2^k}\right)^{j-2}\right] \cdot p^{2^{k+1}} \text{ and } \frac{P^{(j)}(x_k)}{j!} \in \mathbb{Z}.$$

Thus all terms in the summation (for $j \geqslant 2$) are zero modulo $p^{2^{k+1}}$. Consequently, Taylor's formula becomes:

$$0 \equiv P(x_k) + \lambda p^{2^k} P'(x_k) + 0 \pmod{p^{2^{k+1}}}.$$

Furthermore, we have $P(x_k) \equiv 0 \pmod p$, hence by (1), $P'(x_k) \not\equiv 0 \pmod p$, and consequently $P'(x_k) \not\equiv 0 \pmod{p^{2^{k+1}}}$, which allows us to write:

$$\lambda p^{2^k} \equiv -\frac{P(x_k)}{P'(x_k)} \pmod{p^{2^{k+1}}},$$

Now replacing λp^{2^k} with $x_{k+1} - x_k$ and using the induction hypothesis leads to

$$x_{k+1} \equiv u_k - \frac{P(u_k)}{P'(u_k)} \pmod{p^{2^{k+1}}}.$$

Then we recognize the definition (4), which means that we have proved that

$$x_{k+1} \bmod p^{2^{k+1}} = u_{k+1},$$

that is, thanks to the idempotence of modulo, $x_{k+1} = u_{k+1}$. □

3.2 Statement of the uniqueness lemma in the bivariate case

Lemma 3. *Let P_1, P_2 be two bivariate polynomials with integer coefficients, and let p be a prime that satisfies:*

$$\forall z, t \in \mathbb{Z},\quad P_1(z,t) \equiv 0 \equiv P_2(z,t) \pmod p \implies J_{P_1,P_2}(z,t) \not\equiv 0 \pmod p. \tag{6}$$

If $(x, y) \in \mathbb{Z}^2$ is such that

$$P_1(x,y) \equiv P_2(x,y) \equiv 0 \pmod{p^{2^m}} \tag{7}$$

for a given $m \in \mathbb{N}$, then for

$$\begin{pmatrix} u_0 \\ v_0 \end{pmatrix} := \begin{pmatrix} x \bmod p \\ y \bmod p \end{pmatrix}, \tag{8}$$

the sequence $(u_k, v_k)_k$ defined by the recurrence relation

$$\forall k \in [\![0, m[\![\,, \quad \begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} := \begin{pmatrix} u_k \\ v_k \end{pmatrix} - \left[J_{P_1,P_2}(u_k, v_k) \right]^{-1} \begin{pmatrix} P_1(u_k, v_k) \\ P_2(u_k, v_k) \end{pmatrix} \bmod p^{2^{k+1}} \tag{9}$$

satisfies:

$$\forall k \in [\![0, m]\!]\,, \quad \begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} x \bmod p^{2^k} \\ y \bmod p^{2^k} \end{pmatrix}. \tag{10}$$

3.3 Pen-and-paper sketch of the proof of Lemma 3

Let us assume all the hypotheses of the theorem hold for $P_1, P_2 \in \mathbb{Z}[X, Y]$, $p \in \mathbb{P}$, $x, y \in \mathbb{Z}$, and $(u_k), (v_k)$ which are defined in (8) and (9) by mutual recurrence.

We consider the sequences (x_k) and (y_k) of the modular residues of the root (x, y):

$$\forall k \in \mathbb{N}, \quad \begin{pmatrix} x_k \\ y_k \end{pmatrix} := \begin{pmatrix} x \bmod p^{2^k} \\ y \bmod p^{2^k} \end{pmatrix}.$$

We want to show (10), that is to say, $\forall k \in \mathbb{N},\ k \leqslant m \Rightarrow (u_k, v_k) = (x_k, y_k)$. We prove it by induction on k:

• First, we notice that $p = p^{2^0}$, therefore (8) means that $(u_0, v_0) = (x_0, y_0)$, which proves the base case.

• The inductive case amounts to showing that for a given integer $k < m$ satisfying $(u_k, v_k) = (x_k, y_k)$, we have $(u_{k+1}, v_{k+1}) = (x_{k+1}, y_{k+1})$.

We write again $x_k = x \bmod p^{2^k} = [x \bmod p^{2^{k+1}}] \bmod p^{2^k} = x_{k+1} \bmod p^{2^k}$, which implies $x_{k+1} \equiv x_k \pmod{p^{2^k}}$, hence

$$\exists \lambda \in \mathbb{Z}, \quad x_{k+1} = x_k + \lambda p^{2^k}.$$

Likewise, we obtain:

$$\exists \mu \in \mathbb{Z}, \quad y_{k+1} = y_k + \mu p^{2^k}.$$

Now we can apply Taylor's theorem to each bivariate polynomial P_l $(l = 1, 2)$:

$$P_l(x_{k+1}, y_{k+1}) = \sum_{i,j \in \mathbb{N}} \frac{1}{i!j!} \left[\frac{\partial^{i+j}}{\partial X^i \partial Y^j} P_l \right] (x_k, y_k) \cdot \left(\lambda p^{2^k} \right)^i \left(\mu p^{2^k} \right)^j.$$

The first member of this equality is zero modulo $p^{2^{k+1}}$, since

$$P_l(x_{k+1}, y_{k+1}) = P_l(x \bmod p^{2^{k+1}}, y \bmod p^{2^{k+1}}) \equiv P_l(x, y) = 0 \pmod{p^{2^{k+1}}}.$$

Concerning the second member, we can notice that

$$\forall i,j \in \mathbb{N}, \quad i+j \geqslant 2 \implies \left(\lambda p^{2^k}\right)^i \left(\mu p^{2^k}\right)^j = \left[\lambda^i \mu^j \left(p^{2^k}\right)^{i+j-2}\right] \cdot p^{2^{k+1}}$$

and

$$\frac{1}{i!j!}\left[\frac{\partial^{i+j}}{\partial X^i \partial Y^j} P_l\right](x_k, y_k) \in \mathbb{Z},$$

therefore all terms in the summation such that $i+j \geqslant 2$ are zero modulo $p^{2^{k+1}}$. As a result, Taylor's formula becomes

$$0 \equiv P_l(x_k, y_k) + \lambda p^{2^k} \partial_X P_l(x_k, y_k) + \mu p^{2^k} \partial_Y P_l(x_k, y_k) + 0 \pmod{p^{2^{k+1}}}. \quad (11)$$

Note that combining both formulas for $l = 1, 2$, we obtain the following matrix equation:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} P_1(x_k, y_k) \\ P_2(x_k, y_k) \end{pmatrix} + \left[J_{P_1,P_2}(x_k, y_k)\right] \begin{pmatrix} \lambda p^{2^k} \\ \mu p^{2^k} \end{pmatrix} \bmod p^{2^{k+1}}.$$

where the modulo operation is applied coordinatewise.

Furthermore, we have $P_1(x_k, y_k) \equiv 0 \equiv P_2(x_k, y_k) \pmod{p}$, hence by (6), $J_{P_1,P_2}(x_k, y_k) \not\equiv 0 \pmod{p}$, and consequently $J_{P_1,P_2}(x_k, y_k) \not\equiv 0 \pmod{p^{2^{k+1}}}$, which allows us to write

$$-\left[J_{P_1,P_2}(x_k, y_k)\right]^{-1}_{p^{2^{k+1}}} \begin{pmatrix} P_1(x_k, y_k) \\ P_2(x_k, y_k) \end{pmatrix} \equiv \begin{pmatrix} \lambda p^{2^k} \\ \mu p^{2^k} \end{pmatrix} \pmod{p^{2^{k+1}}}.$$

Then, we use the induction hypothesis $(x_k, y_k) = (u_k, v_k)$ after replacing λp^{2^k} with $x_{k+1} - x_k$ (resp. μp^{2^k} with $y_{k+1} - y_k$), and we obtain

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} - \left[J_{P_1,P_2}(u_k, v_k)\right]^{-1} \begin{pmatrix} P_1(u_k, v_k) \\ P_2(u_k, v_k) \end{pmatrix} \equiv \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} \pmod{p^{2^{k+1}}}.$$

We eventually recognize the definition (9), which means we have proved that

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} \bmod p^{2^{k+1}},$$

that is, thanks to the idempotence of the modulo, $(u_{k+1}, v_{k+1}) = (x_{k+1}, y_{k+1})$. □

4 Coq formalization

4.1 Formal background for Hensel's lifting

In this section we present the various choices we have made to undertake this formalization, along with some new support results that are common to both univariate and bivariate cases.

The SSReflect extension. The Small-scale-reflection (SSReflect) extension of Coq was first used to carry out the formalization of the Four-Colour Theorem [3]. It consists of an extension of the Coq proof language as well as a set of Coq libraries developed upon this extension.

We chose to use the SSReflect extension for both its tactic facilities and the Coq libraries it provides. In particular, we extensively used the following libraries that supply most of the key concepts involved in our formalization:

`ssrnat` for boolean comparison predicates on the type `nat` of natural numbers, with the usual operations including exponentiation;
`prime` for the primality predicate `prime : nat -> bool`;
`div` for the divisibility predicate `dvdn : nat -> nat -> bool` and the modulo `modn : nat -> nat -> nat` on which it is based;
`zmodp` for the ExtendedGCD-based modular inversion in $\mathbb{Z}/q\mathbb{Z}$ (where q will be instantiated by p^{2^k});
`poly` for the ring of univariate polynomials, with the polynomial evaluation, formal derivatives, and Taylor's theorem for univariate polynomials;

We also needed the bivariate version of Taylor's theorem that Laurent Théry proved in the theory `bipoly` mainly based on `poly`.

The library on binary integers from the Coq standard library. But we also needed to handle signed integers since we focus on providing certificates of integral roots (in $\mathbb{Z}$). We chose to use the library `ZArith` on signed, binary integers from the Coq standard library whose type `Z` is presented below, essentially for the presence of the modulo operation `Zmod : Z -> Z -> Z` and for efficiency reasons, since the arithmetic operations on `Z` are much more efficient that those defined on the type `nat` of Peano, *unary* integers.

```
Inductive Z : Set :=
  | Z0 : Z
  | Zpos : positive -> Z
  | Zneg : positive -> Z.
```

```
Inductive positive : Set :=
  | xI : positive -> positive
  | xO : positive -> positive
  | xH : positive.
```

For example, the Coq term `Zneg (xO (xO (xI xH)))` will represent the negative number -12, whose absolute value is $\overline{1100}$ in radix 2.

Polynomials with integer coefficients. As regards the definition of polynomials, both `poly` and `bipoly` theories deal with polynomials with coefficients in a type that have to be equipped with a SSReflect decidable-equality ring structure.

Therefore, to be able to talk about polynomials on `Z`, we first needed to prove that `Z` satisfies the required axioms of the SSReflect algebraic hierarchy. This is done in our theory `ssrz` whose first part acts as a wrapper of `ZArith` to be used within SSReflect.

Now the terms `{poly Z}` and `{bipoly Z}` typecheck so that we can use them in the sequel to designate univariate and bivariate polynomials on $\mathbb{Z}$.

Handling different definitions of the modular reduction. In this section, we will summarize why we need to use several definitions of the modular reduction in our formalization. Then we will present some lemmas that allow to move from one definition of to another.

So far, we have mentioned two different definitions of the "modular reduction" available in the considered libraries, namely `modn` on the type `nat`, and `Zmod` on `Z`. The link between `modn` and `Zmod` is provided by a key lemma `Z_of_nat_moduli`:

```
Lemma Z_of_nat_moduli :
  forall n m : nat, m > 0 ->
  Z_of_nat (modn n m) = Zmod (Z_of_nat n) (Z_of_nat m).
```

Note that these two functions differ if the second argument is zero: `(modn 2 0) = 2` whereas `(Zmod 2 0) = 0`, But it is not that distracting since we just need modular reduction modulo $q = p^{2^k} \geqslant p \geqslant 2 > 0$.

Furthermore, for any fixed second argument $q \geqslant 2$, none of these functions are homomorphisms, since $(m + n) \bmod q \neq (m \bmod q) + (n \bmod q)$ in the general case (unless we add a outermost modulo operation in the right-hand-side). However, to prove some key results such as the compatibility between the modulo operation and the polynomial evaluation on `{poly Z}` as well as `{bipoly Z}`, we need to apply lemmas that expect a morphism as argument.

Consequently, we need to specify the surjective morphism from `Z` onto $\mathbb{Z}/q\mathbb{Z}$, even though it is just for proving purposes. For this we use the SSReflect theory `zmodp` that provides a finite type `'I_q` representing $[\![0, q-1]\!]$ for $q \geqslant 1$, which becomes a (nontrivial) ring and is denoted by `'Z_q` when $q \geqslant 2$. More precisely, using the existing surjection `inZp` from `nat` onto `'I_q` we define `ZtoI : forall q : nat, q >= 2 -> Z -> 'Z_q` and we prove it is a ring homomorphism for any fixed $q \geqslant 2$.

For this latter proof we use extensively the following simple lemma that relate `ZtoI` and `Zmod`, and whose proof actually relies on `Z_of_nat_moduli`:

```
Lemma ZtoI_Zmod : forall (q : nat) (q_gt1 : q >= 2) (z : Z),
  Z_of_nat (ZtoI q_gt1 z) = Zmod z (Z_of_nat q).
```

4.2 Certified integral roots for a univariate polynomial

Insights into the Coq formalization of Lemma 1. Thanks to the material presented in 4.1, we can define the univariate Hensel's lifting as a `Fixpoint`, i.e. a recursive function, whose decreasing argument is the integer k. Its type can be displayed after an invocation of the `Check` command:

```
univ_hensel_iter :
  {poly Z} -> forall p : nat, prime p -> Z -> nat -> Z
```

Now we can state Lemma 1 in Coq:

```
Variable P : {poly Z}.
Variable p : nat.
Hypothesis p_prime : prime p.
```

```
Hypothesis P_roots_mod_p :
  forall (z : Z), 0 <= z < Z_of_nat p ->
  P.[z] = 0 %[Zmod p] -> (deriv P).[z] <> 0 %[Zmod p].
Variable x : Z.
Variable m : nat.
Hypothesis x_root_m : P.[x] = 0 %[ZmodN p^2^m].
Let xk := fun k : nat => x modN p^2^k.
Let u0 := x modN p.
Let uk := univ_hensel_iter P p_prime u0.
Lemma univ_hensel_lemma :
  forall k : nat, k <= m -> uk k = xk k.
```

Then we have proved this result with the help of a number of arithmetic lemmas that are gathered in `ssrz`. Among these auxiliary results there is a key lemma that asserts the "compatibility" between polynomial evaluation and modular reduction:

```
Lemma ZmodN_horner_compat :
  forall (q : nat) (z : Z) (P : {poly Z}),
  P.[z modN q] = P.[z] %[ZmodN q].
```

where the notation `P.[z]` represents the polynomial evaluation `(horner P z)`, the notation `m = n %[ZmodN q]` means `m modN q = n modN q`, and `n modN q` (with a big `N`) is the same thing as `n mod (Z_of_nat q)` or `(Zmod n (Z_of_nat q))`.

Specification of our univariate certificates and their checker. If it would be quite easy to verify the output of an algorithm designed for "finding *an* integral root x of $P \in \mathbb{Z}[X]$" (by verifying the equality "$P(x) = 0$" holds), the problem we consider in this paper, namely "finding *all* the integral roots of $P \in \mathbb{Z}[X]$," requires a different approach.

We follow the certificate-based approach described in [13]. The idea is to design a so-called *certifying algorithm*: for a given input x, one such algorithm produces the same output (say y) as a traditional algorithm, plus a witness w that allows us to verify the result. Roughly speaking, the tuple (x, y, w) forms what we call the certificate. For the particular problem which is at stake (find all the integral roots of $P \in \mathbb{Z}[X]$), we thus need to choose an appropriate type of certificate C, then to specify a certificate verifier $v : C \to \mathbb{B}$ and to formally prove that if $v(x, y, w) = \mathsf{true}$ holds, then y is indeed the correct output corresponding to x.

As regards the kind of certifying algorithm that can be used to solve the problem at stake, it suffices to have a Hensel's lifting program that outputs p and k in addition to the list L of the computed roots corresponding to the given polynomial P and bound B.

Then, we can store these data in a form of a certificate, which will be easily checked with our verifier without the need to replay all the iterations of Hensel's lifting.

We consider the following Coq `Record` as the type of our univariate certificates:

```
Record uniCertif := UniCertif {
  uc_P : {poly Z};
  uc_B : Z;
  uc_p : nat;
  uc_k : nat;
  uc_L : seq (Z * bool)
}.
```

In other words, a univariate certificate will be a 5-tuple (P, B, p, k, L). It will be called "valid" if the following conditions are satisfied:

- $P \in \mathbb{Z}[X]$;
- $p \in \mathbb{P}$;
- replacing the binary integer $B \in \mathbb{Z}$ with $|B| \geqslant 0$ for the sequel:
- $k \in \mathbb{N}$ such that $p^{2^k} > 2 \cdot B$;
- $L \in (\mathbb{Z} \times \mathbb{B})^\ell$, a list of ℓ pairs ($\ell \leqslant p$) such that,
  denoting $L_p = \{u \bmod p \ : \ \exists b \in \mathbb{B}, (u, b) \in L\}$, we have:
  - $\forall s \in [\![0, p[\![,\ s \in L_p \iff P(s) \equiv 0 \pmod p$;
  - the ℓ elements of L_p are pairwise distinct;
  - $\forall (u, b) \in L$, $\begin{cases} P'(u) \not\equiv 0 \pmod p, \\ |2 \cdot u| \leqslant p^{2^k}, \\ P(u) \equiv 0 \pmod{p^{2^k}}, \\ b = \mathsf{true} \iff u \in [\![-B, B]\!] \ \wedge \ P(u) = 0. \end{cases}$

Remarks on the formal verification of our univariate checker. From a formal point of view, we have derived our correctness proof of the certificate verifier specified above by rewriting twice `univ_hensel_lemma` (corresponding to Lemma 1). Note that the algorithm `univ_hensel_iter` has been specified in Coq above all for proof purposes, as a "witness" of the uniqueness of modular roots.

4.3 Certified integral roots for a pair of bivariate polynomials

Insights into the Coq formalization of Lemma 3. As mentioned in Section 3.3, we needed Taylor's theorem for bivariate polynomials, which is provided in `bipoly`. Concerning the definition of the type of bivariate polynomials, `{bipoly Z}` is actually a shortcut for `{poly {poly Z}}`.

We start our "bivariate uniqueness" proof by induction on the integer k. The base case is solved trivially, while in the inductive case we need to invoke the bivariate Taylor's theorem.

Note that to truncate the Taylor expansion in order to retrieve the "first three terms" involved in (11), we needed to prove an appropriate lemma `trunc_biv_sum` that relies on some "sum bookkeeping", highly facilitated by the `bigops` SSReflect library.

Finally, we follow the arguments that were presented in a "matrix fashion" at the end of Section 3.3, working coefficient by coefficient. This is accomplished with the help of half a dozen lemmas that are linked to Cramer's rule for 2-by-2 matrices.

Specification of our bivariate certificates and their checker. We consider the following Coq `Record` as the type of our bivariate certificates:

```
Record bivCertif := BivCertif {
bc_P : {bipoly Z};
bc_Q : {bipoly Z};
bc_A : Z;
bc_B : Z;
bc_p : nat;
bc_k : nat;
bc_L : seq (Z * Z * bool)
}.
```

One such certificate (P, Q, A, B, p, k, L) will be valid if the following conditions are satisfied:

- $P \in \mathbb{Z}[X, Y]$ and $Q \in \mathbb{Z}[X, Y]$;
- $p \in \mathbb{P}$;
- $A \in \mathbb{N}$ and $B \in \mathbb{N}$ (same remark than in the univariate case);
- $k \in \mathbb{N}$ such that $p^{2^k} > 2 \cdot A$ and $p^{2^k} > 2 \cdot B$;
- $L \in (\mathbb{Z} \times \mathbb{Z} \times \mathbb{B})^\ell$ s.t., for $L_p = \{(u \bmod p, v \bmod p) : \exists b \in \mathbb{B}, (u, v, b) \in L\}$:
  - the ℓ elements of L_p are pairwise distinct;
  - $\forall (s, t) \in [\![0, p[\![^2$, $(s, t) \in L_p \iff P(s, t) \equiv Q(s, t) \equiv 0 \pmod p$;
  - $\forall (u, v, b) \in L$, $\begin{cases} J_{P_1,P_2}(u, v) \not\equiv 0 \pmod p, \\ |2 \cdot u| \leqslant p^{2^k}, \\ |2 \cdot v| \leqslant p^{2^k}, \\ P(u, v) \equiv Q(u, v) \equiv 0 \pmod{p^{2^k}}, \\ b = \mathsf{true} \iff \begin{cases} |u| \leqslant A, \\ |v| \leqslant B, \\ P(u, v) = Q(u, v) = 0. \end{cases} \end{cases}$

These boolean conditions are formalized in the form of a Coq boolean function

```
biv_check : bivCertif -> bool
```

Remarks on the formal verification of our bivariate checker. Among the various fields involved in our bivariate certificate `bivCertif`, P, Q, A, B represents the data of the problem, whereas $L' := \{(u, v) : (u, v, \mathsf{true}) \in L\}$ represents the output of the problem, namely finding the integral roots $(x, y) \in [\![-A, A]\!] \times [\![-B, B]\!]$ of (P, Q).

As regards the correctness proof of the bivariate checker, we need to prove that for all certificate that is accepted by the checker, the output stored in it is valid.

To be more precise, the list $\{(u, v) : (u, v, \mathsf{true}) \in L\}$ can be defined in Coq in the following manner:

```
Definition bc_roots (bc : bivCertif) := map fst (filter snd (bc_L bc)).
```

Then, we derive the correctness proof of `biv_check` that consists of proving that for all `bc : bivCertif` such that `(biv_check bc)` holds, for all $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ we have the equivalence

$$|x| \leqslant A \ \wedge \ |y| \leqslant B \ \wedge \ P(x, y) = Q(x, y) = 0 \iff (x, y) \in \texttt{(bc\_roots bc)}.$$

5 Discussion on the certificate-based approach

First of all, the SLZ algorithm that is the main application of our work is a very complex algorithm whose future implementations will certainly have recourse to approximated calculations as well as nasty code optimizations to save computation time. It is thereby totally excluded to formally prove the algorithm or its implementation. Hence the need to follow the certificate-based approach.

Compared to the formal verification of a traditional algorithm (for providing what we can call a *certified* algorithm), the certificate-based approach that relies on a *certifying* algorithm [13] ensures that the computed result has not been compromised by any bug. Moreover, with this approach we do not need to verify the implemented program nor the algorithm itself. This means that we could even use a "fast-and-dirty program" to do the job, since the result can be easily checked by the certificate verifier that has been formally proved in Coq. In compensation the checker itself has to be somewhat efficient since the approach rely on the individual verification of each result.

6 Conclusion and Future Work

The algorithm of Hensel's lifting itself has been successfully used in the area of Computer Arithmetic, in particular [10] and [16] rely on the geometric variant of Hensel's lifting (i.e., with the moduli increasing in a geometric way), for $p = 2$. In this work, we have introduced some integral roots certificates whose specification is closely related to the semantics of Hensel's lifting. Then we have formally proved in the Coq system the correctness of our certificate checkers: if a certificate is declared valid by our verifier, we are sure that it describes all the integral roots of the considered polynomial(s) below the considered bound. Furthermore, the formal proof of these results relies on a *uniqueness* property on the modular roots produced by Hensel's lifting that is, up to our knowledge, not very discussed in the literature. Note that the use of Hensel's lifting is two-fold: it can be first used to generate "p", "k" and the corresponding roots, and it is also used in the form of a simple, non-optimized algorithm defined within Coq to reason on the uniqueness of the modular roots it produces.

If we naturally started our work with the univariate case before generalizing to pairs of bivariate polynomials, in the latest stages of the formalization some global improvement ideas actually came from the bivariate case where the efficiency of the approach becomes a central concern. For instance, at first it

seemed possible to store (with a boolean) the status of all the numbers modulo p (as in practice $p \lessapprox 17$) in the univariate case, however in the bivariate case it would have forced us to systematically store p^2 values, with *hundreds* of superfluous values, for a single certificate. Thus in both univariate and bivariate cases, we provide an integral-roots certificate that is somewhat compact and can be quickly checked by a verifier that does not recompute the prime p nor replays all the iterations of Hensel's lifting. The code of our library will be available along with this research report as soon as it is stable.

During the formalization process, we noticed that most of the semantics of our certificate is carried by the specification of the verifier, and that some slight changes in this specification can lead to "false-negative," namely a given certificate could be wrongly denied by the verifier. We thus envisage to develop some extra proofs that demonstrate the absence of such situations, even though in the certificate-approach the most important thing is not to have "false-positive," which is asserted by our main correctness lemmas.

As said before, this work is within the scope of "certifying algorithms" as introduced by [13]. The certifying algorithm we have presented will lead to an important component that can be plugged into the "root-finding step" of SLZ [18] to solve the Table Maker's Dilemma in an exact way. The safety of the result is ensured at the same time by the formal verification performed by Coq and by the certificate-based approach that makes it possible to separate the verification of the result from its (possibly intensive) computation.

Acknowledgements

The author wishes to thank Guillaume Hanrot, Micaela Mayero, Jean-Michel Muller, Ioana Pasca and Laurent Théry for their precious advice and help.

References

1. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science, Springer-Verlag (2004), http://www.labri.fr/publications/l3a/2004/BC04
2. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press (2003)
3. Gonthier, G.: Formal Proof—The Four-Color Theorem. Notices of the American Mathematical Society 55(11), 1382–1393 (2008), http://www.ams.org/notices/200811/tx081101382p.pdf
4. Gonthier, G., Mahboubi, A.: A Small Scale Reflection Extension for the Coq system. Research Report RR-6455, INRIA (2009), http://hal.inria.fr/inria-00258384
5. Gonthier, G., Roux, S.L.: An Ssreflect Tutorial. Technical Report RT-0367, INRIA (2009), http://hal.inria.fr/inria-00407778/en/
6. Grégoire, B., Théry, L.: A purely functional library for modular arithmetic and its application to certifying large prime numbers. In: Furbach, U., Shankar, N. (eds.)

Automated Reasoning, Lecture Notes in Computer Science, vol. 4130, pp. 423–437. Springer (2006)
7. Grégoire, B., Théry, L., Werner, B.: A computational approach to Pocklington certificates in type theory. In: Hagiya, M., Wadler, P. (eds.) Functional and Logic Programming, Lecture Notes in Computer Science, vol. 3945, pp. 97–113. Springer (2006)
8. Hensel, K.: Neue Grundlagen der Arithmetik. Journal für die reine und angewandte Mathematik (Crelle's Journal) 1904(127), 51–84 (1904), 10.1515/crll.1904.127.51
9. Huet, G., Kahn, G., Paulin-Mohring, C.: The Coq proof assistant: a tutorial: version 8.2 (2010), `http://coq.inria.fr/distrib/V8.2pl2/files/Tutorial.pdf`
10. Kahan, W.: A Test for Correctly Rounded SQRT (May 1996), `http://www.cs.berkeley.edu/~wkahan/SQRTest.ps`, Lecture note
11. Lefèvre, V., Muller, J.M.: Worst Cases for Correct Rounding of the Elementary Functions in Double Precision. In: Burgess, N., Ciminiera, L. (eds.) Proceedings of the 15th IEEE Symposium on Computer Arithmetic. pp. 111–118. Vail, CO (Jun 2001)
12. Lenstra, A.K., Lenstra, Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 515–534 (1982)
13. McConnell, R.M., Mehlhorn, K., Näher, S., Schweitzer, P.: Certifying Algorithms (June 2010), `http://www.mpi-inf.mpg.de/~mehlhorn/ftp/CertifyingAlgorithms.pdf`, to appear in Computer Science Review
14. Muller, J.M.: Elementary Functions, Algorithms and Implementation. Birkhäuser Boston, MA, 2nd edn. (2006)
15. Muller, J.M., Brisebarre, N., de Dinechin, F., Jeannerod, C.P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of Floating-Point Arithmetic. Birkhäuser (2009)
16. Parks, M.: Number-Theoretic Test Generation for Directed Rounding. In: 14th IEEE Symposium on Computer Arithmetic. pp. 241–248. Adelaide, Australia (1999)
17. Stehlé, D.: Algorithmique de la réduction des réseaux et application à la recherche de pires cas pour l'arrondi des fonctions mathématiques. Ph.D. thesis, Université Nancy 1 Henri Poincaré (Dec 2005)
18. Stehlé, D.: On the Randomness of Bits Generated by Sufficiently Smooth Functions. In: Hess, F., Pauli, S., Pohst, M.E. (eds.) Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4076, pp. 257–274. Springer-Verlag (2006), `http://dx.doi.org/10.1007/11792086_19`
19. Stehlé, D., Lefèvre, V., Zimmermann, P.: Searching Worst Cases of a One-Variable Function Using Lattice Reduction. IEEE Transactions on Computers 54(3), 340–346 (Mar 2005)
20. The Coq Development Team: The Coq Proof Assistant: Reference Manual: version 8.2 (2010), `http://coq.inria.fr/distrib/V8.2pl2/files/Reference-Manual.pdf`
21. Théry, L., Hanrot, G.: Primality proving with elliptic curves. In: Schneider, K., Brandt, J. (eds.) Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science, vol. 4732, pp. 319–333. Springer (2007)