Proving an Execution of an Algorithm Correct?

Alex J. Best, Mario Carneiro, Edgar Costa, and James Harold Davenport
edgarc@mit.edu masjhd@bath.ac.uk

[1] Imperial College
[2] ??
[3] M.I.T.
[4] University of Bath, Bath BA2 7AY, UK

Abstract. Many algorithms in computer algebra and beyond produce answers. For some of these, we have formal proofs of the correctness of the algorithm, and for others it is easy to verify that the answer is correct. Other algorithms produce either an answer or a proof that no such answer exists. It may still be easy to verify that the answer is correct, but what about the "no such answer" case. A slight variant on "no such answer" is given by polynomial factoring, where we say "f factors as $g \cdot h$" but imply "and no more", i.e. that g and h don't factor. How might an algebra system help produce such a proof?

1 Introduction

[Dav23] asked whether various computer algebra algorithms could produce evidence that a theorem-prover could use to prove correctness of that specific execution of the algorithm. It turns out that the question has been asked, in the context of graph algorithms by Kurt Mehlhorn [Meh11] (in 1999 according to Tobias Nikpow), and there's some useful theory in [MMNS11].

[Dav23] looked at three examples.

Polynomial Factorisation $f = f_1 f_2 \cdots f_k$ *and* the f_i is irreducible.
Integration The assertion "unintegrable" is correct.
Real Algebraic Geometry The assertion that the semi-algebraic variety is empty (UNSAT) is correct.

The last is the most important question, but factorisation is easy to explain and a good case study in its own right.

2 Polynomial Factorisation: the theory

For simplicity we will consider factorisation over the integers of polynomials with integer coefficients. Algebraic number fields add complications, but not, we believe, fundamental ones. The problem of factorisation is normally stated as follows.

Problem 1 (Factorisation). Given $f \in \mathbf{Z}[x_1, \ldots, x_n]$, write $f = \prod f_i$ where the f_i are *irreducible* elements of $\mathbf{Z}[x_1, \ldots, x_n]$.

Verifying that $f = \prod f_i$ is, at least relatively, easy. The hard part is verifying that the f_i are *irreducible*. The author knows of no implementation of polynomial factorisation that produces any evidence, let alone a proof, of this.

2.1 Univariate Polynomials

We may as well assume f is square-free (else factor each square-free factor separately). Then the basic algorithm goes back to [Zas69]: step M is a later addition [Mus75], and the H' variants are also later.

1. Choose a prime p (not dividing the leading coefficient of f) such that $f \pmod p$ is also square-free. For technical reasons we tend to avoid $p = 2$.
2. Factor f modulo p as $\prod f_i^{(1)} \pmod p$. There are essentially two options here: Berlekamp "small prime" [Ber67] and the Cantor–Zassenhaus Las Vegas algorithm [CZ81].

M Take five such p and compare the factorisations.

3. If f can be shown to be irreducible from modulo p factorisations, return f.
4. Let B be such that any factor of f has coefficients less than B in magnitude, and n such that $p^n \geq 2B$. Generally the Landau–Mignotte bound [Lan05,Mig74].
5. Use Hensel's Lemma to lift the factorisation to $f = \prod f_i^{(n)} \pmod{p^n}$

H Starting with singletons and working up (pairs, triples, ... r-tuples [Col79]), take subsets of the $f_i^{(n)}$, multiply them together and check whether, regarded as polynomials over $\mathbf{Z}$ with coefficients in $[-B, B]$, they divide f — if they do, declare that they are irreducible factors of f.

H' Use some alternative technique, originally [LLL82], but now e.g. [ASZ00] to find the true factor corresponding to $f_1^{(n)}$, remove $f_1^{(n)}$ and the other $f_i^{(n)}$ corresponding to this factor, and repeat.

In practice, there are a lot of optimisations, which would greatly complicate a proof of correctness of an implementation of this algorithm.

> We found that, although the Hensel construction is basically neat and simple in theory, the fully optimised version we finally used was as nasty a piece of code to write and debug as any we have come across [MN81].

Since if f is irreducible modulo p, it is irreducible over the integers, the factors produced from singletons in step 5 are easily proved to be irreducible. Unfortunately, the chance that an irreducible polynomial of degree n is irreducible modulo p is $1/n$. Hence the factorisation in step 2 is very likely to be an overestimate, in that we have more factors modulo p than over the integers.

Musser introduced step M, saying we should take five[5] primes p_i and compare the factorisations. This is more than just taking the best (where the chance of

[5] Subsequently [LP97] showed that asymptotically the correct number is seven, not Musser's experimentally-derived five.

irreducibility would then be roughly $5/n$). For example, if f factors as 3×1 (i.e. a factor of degree 3 times a linear factor) modulo p_1 and 2×2 modulo p_2, then it must in fact be irreducible. For a generic polynomial (Galois group S_n) this is very likely to prove f irreducible.

However, [SD69] showed that there are irreducible polynomials which factor *compatibly* modulo every prime. The easiest example is $x^4 + 1$, which factors as 2×2 (or 2×1^2 or 1^4) modulo every prime, which is also compatible with a 2×2 factorisation over the integers, and the recombination part of step 5 may be required.

Hence we can see that a factorisation algorithm could, even though no known implementation does, relatively easily produce the required information for a proof of irreducibility unless the recombination step is required. Note that *verifying* the Hensel lifting, the "nasty piece" from [MN81] is easy: the factors just have to have the right degrees from the factorisation of $f \pmod{p}$ and multiply to give $f \pmod{p^n}$.

2.2 Multivariate Polynomials

The algorithm is basically similar, replacing primes by evaluations $x_i \to v_i$. The difference is that, if $f(x_1, \ldots, x_n)$ is irreducible, then with probability 1, $f(x_1, v_2, \ldots, v_n) \in \mathbf{Z}[x_1]$ is also irreducible. Hence this is probably not significantly easier than the univariate case in terms of proving, unlike implementation [MN81].

3 The practice: FLINT

We chose FLINT [tea23] as an open-source implementation with which the third author was familiar. The basic implementation of polynomial factorisation largely follows the scheme of §2, but there are some differences.

1. Step M uses three primes.
2. Step H checks the degree of the product to be verified against the list of possible degrees from M, and can therefore rule out a potential product.

4 Worked example

This polynomial is given in the Zassenhaus tests of FLINT.

$$\begin{aligned}
&x^{62} + x^{61} + x^{60} - 4x^{59} - 7x^{58} - 2x^{57} - 6x^{56} - 3x^{55} - 7x^{54} + 18x^{53} + 7x^{52} \\
&+25x^{51} - 11x^{50} + 95x^{49} + 36x^{48} + 21x^{47} + 16x^{46} + 69x^{45} + 56x^{44} + 35x^{43} \\
&+36x^{42} + 32x^{41} + 33x^{40} + 26x^{39} - 26x^{38} - 15x^{37} - 14x^{36} - 53x^{35} - 96x^{34} \\
&+67x^{33} + 72x^{32} - 67x^{31} + 40x^{30} - 79x^{29} - 116x^{28} - 452x^{27} - 312x^{26} \\
&-260x^{25} - 29x^{24} - 1393x^{23} + 327x^{22} + 69x^{21} - 28x^{20} - 241x^{19} + 230x^{18} \\
&-54x^{17} - 309x^{16} - 125x^{15} - 74x^{14} - 450x^{13} - 69x^{12} - 3x^{11} + 66x^{10} \\
&-27x^9 + 73x^8 + 68x^7 + 50x^6 - 63x^5 - 1290x^4 + 372x^3 + 31x^2 - 16x + 2
\end{aligned} \tag{1}$$

[Kon23] quotes a computation by Max Horn in MAGMA that the Galois group)over $\mathbf{Z}$ is $S_{31} \times S_{31}$.

Table 1. Factorisation shapes

Prime p	Factor shape modulo p	Possible degrees
5	2,2,7,12,19,20	2,4,7,9,11,12,14,16,19,20,21,22,23,24,26,27,28,29,30,31...
7	1,1,1,3,4,5,6,7,7,13,14	1,2,3,4,5,6,7,8,9,10,11,12,...
5 and 7		no improvement
11	2,2,2,10,19,27	2,4,6,10,12,14,16,19,21,23,25,27,29,31,...
5,7,11		2,4,12,14,16,19,21,23,27,29,31...
13	1,2,3,10,17,29	$1, \cdots, 6, 10, \cdots, 23, 27, \cdots, 35, \ldots$
5,7,11,13		no improvement
17	4,8,9,18,23	4,8,9,12,13,17,18,21,22,23,26,27,31,...
5,7,11,13,17		4,12,21,23,27,31,...
19	1,4,4,10,12,13,18	1,4,5,8,9,10,11,12,13,14,15,16,17,18,19,20,21
5,7,11,13,17,19		no improvement
23	1,1,1,4,10,20,25	$1, \cdots, 7, 10, \cdots, 17, 20, \cdots, 27, 30, 31, \ldots$
5,7,11,13,17,19,23		no improvement

M (adjusted) This polynomial is square-free, but not square-free modulo 3. Hence step M, as adjusted, takes the first three primes in table 1.

Choice We take $p = 11$, hence factors of degree 2,2,2,10,19,27.

JHD: should we explain this?

Bound We use 11^{23}.

Try $r = 1$ We see if each single factor modulo 11^{23}, regarded over $\mathbf{Z}$ and made primitive, is a factor, *except* the degree 10 factor, which is ruled out by Table 1. Note that, if we had used 5 primes, as recommended by [Mus75], the extended version of Table 1 would also have ruled out the degree 19 factor, and the degree 2 factors.

Try $r = 2$ We see if each pair of factors: 27+2 (three options), 2+2 (three options), 10+2 (three options), 27+19, 19+2 (three times), 19+10 is a factor, *except* the degree 27+10, which is ruled out by Table 1 [degree 37, so co-factor would have degree 25, which is ruled out. 5 primes would also have ruled out 27+2 (three options), 27+19 and 19+10.

Try $r = 3$ We make one unsuccessful attempt with a degree 31 combination (27+2+2), then we find a successful one.

4.1 Digression about how many primes

We have seen that using 5 primes rather than Flint's choice of 3 would mean that several more recombinations could be ruled out. But in fact, if we used 5 primes, we would also decide that $p = 17$, with five factors, was a better choice than $p = 11$, with six factors. As we can see in Table 2,we would look at five combinations in all, versus ten with $p = 11$ and five primes or nineteen with $p = 11$ and three primes.

Table 2. Factorisation shortcuts

p raw	using 3 primes	using 5 primes
11 6/15	5/14	1/9
17 5/10	—	2/3

Moving to seven primes doesn't help, as can be seen in Table 1. We do get some progress at $p = 41$ (11 primes),when the factorisation has degrees 2,2,4,23,31, ruling out possible degrees 12 and 21, leaving 4,23,27,31... as the options. $p = 43$ then has factorisation degrees 1,3,3,27,28, ruling out 23. The next progress is at $p = 59$, with factorisation shapes 1,5,8,17,31, ruling out both 4 and 27, thus forcing a 31/31 split.

5 So what is the certificate?

As far as we can tell, these are the components needed.

1. The 5(?) Musser primes (or the useful subset)
2. The factorisations modulo these
* Need to verify that these are irreducible.
3. The chosen p and n
4. The set S of factors modulo p^n
* Need to check they match the mod p ones, and multiply. We have already proved irreducibility of the mod p versions
5. The partition $S = \bigcup S_i$ that corresponds to the true factorisation.
\+ Note that we need merely check that subsets of each S_i do not give rise to factors, rather than subsets of the whole of S.

Thanks to Tobias Nipkow for asking this explicitly.

6 Towards a Formal Proof

We will use Lean as our theorem prover, though the choice is probably not fundamental.

We need various results which we suggest should be proved as "baseline" results: we call these "theorems".

Theorem 1 (Landau–Mignotte Inequality [Lan05,Mig74,Mig82]). *Let $Q = \sum_{i=0}^{q} b_i x^i$ be a divisor of the polynomial $P = \sum_{i=0}^{p} a_i x^i$ (where a_i and b_i are integers). Then*

$$eq : LM1 \max_{i=0}^{q} |b_i| \leq \sum_{i=0}^{q} |b_i| \leq 2^q \left| \frac{b_q}{a_p} \right| \sqrt{\sum_{i=0}^{p} a_i^2}. \tag{2}$$

If we regard P as known and Q as unknown, this formulation does not quite tell us about the unknowns in terms of the knowns, since there is some dependence on Q on the right, but we can use a weaker form:

$$\sum_{i=0}^{q} |b_i| \leq 2^p \sqrt{\sum_{i=0}^{p} a_i^2}. \tag{3}$$

In practice, we use [Abb13] as our source for the bounds on factors. Note, however, that these theorems rely on real and complex analysis, even though the statements are about $\mathbf{Z}[x]$. The precise formulation in Flint is slightly different (taken from [Coh93, Theorem 3.5.1]):

$$|b_j| \leq \binom{n-1}{j} \sqrt{\sum_{i=0}^{p} a_i^2} + \binom{n-1}{j-1} |a_m| \tag{4}$$

There are various algorithms for factoring polynomials modulo p, but all we need to do is verify that the factors given the verifier by the algebra system multiply correctly and are irreducible. The following theorem is probably the easiest way of doing that.

Theorem 2 (After [CZ81]). *A square-free polynomial f of degree d modulo p is irreducible if* $\gcd(f, x^{p^i} - x) \equiv 1 \pmod{p}$ *for* $1 \leq i \leq d/2$.

Acknowledgements

The fourth author is supported by EPSRC grant EP/T015713.

References

Abb13. J.A. Abbott. Bounds on Factors in $\mathbf{Z}[x]$. *J. Symbolic Comp.*, 50:532–563, 2013.

ASZ00. J.A. Abbott, V. Shoup, and P. Zimmermann. Factorization in $\mathbf{Z}[x]$: The Searching Phase. In C. Traverso, editor, *Proceedings ISSAC 2000*, pages 1–7, 2000.

Ber67. E.R. Berlekamp. Factoring Polynomials over Finite Fields. *Bell System Tech. J.*, 46:1853–1859, 1967.

Ber70. E.R. Berlekamp. Factoring Polynomials over Large Finite Fields. *Math. Comp.*, 24:713–735, 1970.

Coh93. H. Cohen. A Course in Computational Algebraic Number Theory. *Graduate Texts in Mathematic 138*, 1993.

Col79. G.E. Collins. Factoring univariate integral polynomials in polynomial average time. In *Proceedings EUROSAM 79*, pages 317–329, 1979.

CZ81. D.G. Cantor and H. Zassenhaus. A New Algorithm for Factoring Polynomials over Finite Fields. *Math. Comp.*, 36:587–592, 1981.

Dav23. James Harold Davenport. Proving an Execution of an Algorithm Correct? In Catherine Dubois and Manfred Kerber, editors, *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, pages 255–269, 2023.

Kon23. O. Konovalov. RE: Degree 62 polynomial whose Galois Group I would like to know. *E-mail to JHD 09:31 5 October*, 2023.

Lan05. E. Landau. Sur Quelques Théorèmes de M. Petrovic Relatif aux Zéros des Fonctions Analytiques. *Bull. Soc. Math. France*, 33:251–261, 1905.

LLL82. A.K. Lenstra, H.W. Lenstra Jun., and L. Lovász. Factoring Polynomials with Rational Coefficients. *Math. Ann.*, 261:515–534, 1982.

LP97. T. Łuczak and L. Pyber. On random generation of the symmetric group. In *Proceedings Combinatorics geometry and probability*, pages 463–470, 1997.

Meh11. K. Mehlhorn. Certifying Algorithms. `https://people.mpi-inf.mpg.de/~mehlhorn/ftp/CertifyingAlgs.pdf`, 2011.

Mig74. M. Mignotte. An Inequality about Factors of Polynomials. *Math. Comp.*, 28:1153–1157, 1974.

Mig82. M. Mignotte. Some Useful Bounds. *Symbolic and Algebraic Computation (Computing Supplementum 4) Springer-Verlag*, pages 259–263, 1982.

MMNS11. R.M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5:119–161, 2011.

MN81. P.M.A. Moore and A.C. Norman. Implementing a Polynomial Factorization and GCD Package. In *Proceedings SYMSAC 81*, pages 109–116, 1981.

Mus75. D.R. Musser. Multivariate Polynomial Factorization. *J. ACM*, 22:291–308, 1975.

SD69. H.P.F. Swinnerton-Dyer. Letter to E.R. Berlekamp. *Mentioned in [Ber70]*, 1969.

tea23. The FLINT team. *FLINT: Fast Library for Number Theory*, 2023. Version 2.9.0, `https://flintlib.org`.

Zas69. H. Zassenhaus. On Hensel Factorization I. *J. Number Theory*, 1:291–311, 1969.