Complex quantifier elimination in HOL

John Harrison

Intel Corporation, EY2-03
5200 NE Elam Young Parkway
Hillsboro, OR 97124, USA

Abstract. Building on a simple construction of the complex numbers and a proof of the Fundamental Theorem of Algebra, we implement, as a HOL derived inference rule, a decision method for the first order algebraic theory of $\mathbb{C}$ based on quantifier elimination. Although capable of solving some mildly interesting problems, we also implement a more efficient semidecision procedure for the universal fragment based on Gröbner bases. This is applied to examples including the automatic proof of some simple geometry theorems. The general and universal procedures present an interesting contrast in that the latter can exploit the finding-checking separation to achieve greater efficiency, though this feature is only partly exploited in the present implementation.

1 Introduction

The complex numbers have the interesting property of being algebraically closed, i.e. every non-constant polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ has a zero. This implies that a polynomial over $\mathbb{C}$ has a factorization into linear factors $(x - b_1) \cdots (x - b_n)$, which in turn gives rise to a relatively straightforward method of finding a quantifier-free equivalent for any first order algebraic formula involving polynomial equations and inequations, e.g.

$$\begin{aligned}&(\exists x\, y.\ ax^2 + bx + c = 0 \wedge ay^2 + by + c = 0 \wedge \neg(x = y))\\ &\equiv a = 0 \wedge b = 0 \wedge c = 0 \vee \neg(a = 0) \wedge \neg(b^2 = 4ac)\end{aligned}$$

In this paper, we describe the implementation of such a quantifier elimination procedure as a derived rule, i.e. one that proceeds at all stages constructing a formal proof. Although it is capable of tackling some moderately interesting examples (like the one above), it is not efficient enough for more difficult examples. We also implement a more limited semi-decision procedure to prove valid purely universal formulas, i.e. those that when put in prenex normal form have a quantifier prefix $\forall x_1, \ldots, x_n.\ \cdots$. This is based on the standard method of Gröbner bases, implemented so that it records the sequence of operations in a way that can be turned into a HOL proof. This procedure is applied to some examples derived by converting geometry theorems into coordinates.

2 Constructing the complex numbers

The complex numbers are isomorphic to $\mathbb{R} \times \mathbb{R}$, so the HOL definition of the new type is trivial. The mutually inverse type bijections are called `complex`: $\mathbb{R}^2 \to \mathbb{C}$ and `coords`$:\mathbb{C} \to \mathbb{R}^2$. We define convenient abbreviations for the real and complex parts of a complex number, the natural embedding `Cx`$:\mathbb{R} \to \mathbb{C}$ and the imaginary unit i (we use `ii` since `i` is a useful variable name).

```
|- Re(z) = FST(coords(z))

|- Im(z) = SND(coords(z))

|- ii = complex(&0,&1)
```

It's now straightforward to define the arithmetic operations on complex numbers in the usual way (note that we overload the usual symbols for $\mathbb{C}$ and $\mathbb{R}$), e.g.

```
|- w + z = complex(Re(w) + Re(z),Im(w) + Im(z))

|- w * z = complex(Re(w) * Re(z) - Im(w) * Im(z),
                   Re(w) * Im(z) + Im(w) * Re(z))
```

To help get started, we implement a trivial tool for very simple algebraic identities, which simply rewrites with definitions to decompose a complex equation into a pair of real equations for the real and complex parts, then applies the standard HOL Light semidecision procedure for $\mathbb{R}$:

```
let SIMPLE_COMPLEX_ARITH_TAC =
  REWRITE_TAC[COMPLEX_EQ; RE; IM; Cx;
              complex_add; complex_neg; complex_sub; complex_mul] THEN
  REAL_ARITH_TAC;;
```

This can prove most of the routine ring-type algebraic identities automatically, e.g:

```
let COMPLEX_RNEG_UNIQ = prove
 (`!x y. (x + y = Cx(&0)) = (y = --x)`,
  SIMPLE_COMPLEX_ARITH_TAC);;

let COMPLEX_DIFFSQ = prove
 (`!x y. (x + y) * (x - y) = x * x - y * y`,
  SIMPLE_COMPLEX_ARITH_TAC);;
```

Although not very efficient or wide-ranging, this saves a lot of tedious manual proofs during the "bootstrapping" phase. Later on, this simple tactic is subsumed by a more advanced semidecision procedure. In any case, when it comes to properties of the inverse, we need to perform a few manual proofs, though they are not at all difficult. We also need to define complex moduli; note that the type is `mod`$:\mathbb{C} \to \mathbb{R}$, since we often want to make inequality comparisons. We could, of course, work consistently in the real-valued subset of the complex numbers, but theorems would then become cluttered with conditions that are more conveniently encoded in the types.

```
|- mod(z) = sqrt(Re(z) pow 2 + Im(z) pow 2)
```

Many of the properties of this operation are also trivial to prove. Slightly harder is the triangle inequality, which requires a larger 13-step proof:

```
|- !w z. mod(w + z) <= mod(w) + mod(z)
```

We also need to define complex square roots; again the definition is easy enough in terms of the existing real function, though a bit messier than might be expected because of the special case of a real-valued complex number:

```
|- csqrt(z) = if Im(z) = &0 then
                 if &0 <= Re(z) then complex(sqrt(Re(z)),&0)
                 else complex(&0,sqrt(--Re(z)))
              else complex(sqrt((mod(z) + Re(z)) / &2),
                           (Im(z) / abs(Im(z))) *
                           sqrt((mod(z) - Re(z)) / &2))
```

though the proof of its basic property:

```
|- !z. csqrt(z) pow 2 = z
```

requires a fairly large manual proof.

3 The fundamental theorem of algebra

To state the fundamental theorem of algebra we need to define polynomial functions. We could simply represent them as explicit finite summations, but to make the generic statement of later theorems simpler, we define a representation of univariate polynomial functions as lists of coefficients (without gaps, starting at the constant term). In fact, we simply re-used the existing HOL Light theory of real polynomial functions [11]. Essentially all the proofs were simply taken over with trivial changes, except for some with no complex analog (e.g. those concerned with ordering or, like Sturm's theorem, specific to $\mathbb{R}$), or where relevant concepts like differentiability had not yet been transferred over to $\mathbb{C}$. The core definition is simply:

```
|- (poly [] x = Cx(&0)) /\
   (poly (CONS h t) x = h + x * poly t x)
```

The cleanest statement of the Fundamental Theorem of Algebra uses the auxiliary notion of a constant function:

```
|- constant f = !w z. f(w) = f(z)
```

namely:

```
|- !p. ~constant(poly p) ==> ?z. poly p z = Cx(&0)
```

However, we sometimes want a more syntactic criterion for a polynomial to be nonconstant. The following states that a complex polynomial has a root unless the list of coefficients starts with a nonzero constant and has all other coefficients zero (note that our representation of polynomial functions does not impose canonicality).

```
|- !p. ~(?a l. ~(a = Cx(&0)) /\ ALL (\b. b = Cx(&0)) l /\ (p = CONS a l))
       ==> ?z. poly p z = Cx(&0)
```

We formalize a proof taken from [8], an inductive refinement [19, 9] of the classic 'minimum modulus' proof à la Argand. The crucial analytical component is the assertion that a continuous complex function attains its minimum modulus in a closed disc in the complex plane. This is essentially an assertion that a closed disc in $\mathbb{C}$ is topologically compact. As is well-known, there are numerous equivalent characterizations of compactness. We chose the Bolzano-Weierstrass type formulation 'every sequence in the set has a convergent subsequence', since this seemed the simplest to derive from the analogous result from $\mathbb{R}$ (already proved in HOL) by handling the real and complex coordinates successively and using the simple lemma:

```
|- !x y. mod(x - y) <= abs(Re(x) - Re(y)) + abs(Im(x) - Im(y))
```

The compactness theorem proved is as follows, where `subseq` f means that $f : \mathbb{N} \to \mathbb{N}$ is a strictly increasing function, and hence parametrizes a subsequence.

```
|- !s r. (!n. mod(s n) <= r)
         ==> ?f z. subseq f /\
                   !e. &0 < e ==> ?N. !n. n >= N ==> mod(s(f n) - z) < e
```

This is now easily used to show that a polynomial attains its minimum modulus on a finite disc. We also prove that a nonconstant polynomial goes to infinity as the magnitude of its argument does. Thus we can find a closed disc outside which the polynomial has larger magnitude than it does at (say) zero. It follows that the attained minimum modulus inside such a disc is in fact the *overall* minimal modulus:

```
|- !p. ?z. !w. mod(poly p z) <= mod(poly p w)
```

All that remains is to prove that for a nonconstant polynomial this modulus is zero, by showing that if not, we could actually find another point where the modulus is smaller, contradicting minimality. We start by showing that we can always reduce the magnitude of a unimodular complex number by moving in one of the four compass directions in the complex plane:

```
|- !z. (mod(z) = &1)
       ==> mod(z + Cx(&1)) < &1 \/
           mod(z - Cx(&1)) < &1 \/
           mod(z + ii) < &1 \/
           mod(z - ii) < &1
```

Now, in a sufficiently small neigbourhood of a point, a polynomial is dominated by the term of lowest degree. So we essentially just need to be able to find a ‘direction’ d so that $|1+bz^n q(z)| < 1$ for all $z = td$ with $t \in \mathbb{R}$ sufficiently small. First we prove:

```
|- !b n. ~(b = Cx(&0)) /\ ~(n = 0)
        ==> ?z. mod(Cx(&1) + b * z pow n) < &1
```

by wellfounded induction on n. If n is even we can just apply the inductive hypothesis to $n/2$ and take complex square roots. Otherwise if n is odd, note that the ‘compass point’ values $\{1,-1,i,-i\}$ are merely permuted by $x \mapsto x^n$ so we can reduce the problem to the special case $n = 1$, when it's simply a matter of performing division.

Now we prove by interlocking induction on degrees that the fundamental theorem holds together with a stronger form of the above lemma. Using this technique, we avoid separately having to justify the existence of complex n^{th} roots for $n > 2$. More precisely, let $P(n)$ be the statement that the fundamental theorem holds for polynomials of degree n and $Q(n)$ be the statement that for any polynomial $h(z)$ of degree n of the form:

$$h(z) = 1 + z^k(b + zg(z))$$

there is a u with $|h(u)| < 1$. What we prove is

- $(\forall k.\ k < n \Longrightarrow P(k)) \Longrightarrow Q(n)$
- $Q(n) \Longrightarrow P(n)$

The second part is the standard minimum modulus argument sketched above. For the first part, note that if we simply have $h(z) = 1 + bz^k$, i.e. $g(z) = 0$, then we already have the result as proved above. Otherwise we have $k < n$ so by the $P(k)$ hypothesis we can find a ‘direction’ $d \in \mathbb{C}$ such that $1 + bd^k = 0$. If $t \in \mathbb{R}$, we have

$$\begin{aligned} h(td) &= 1 + (td)^k(b + (td)g(td)) \\ &= 1 - \frac{1}{b}t^k(b + (td)g(td) \\ &= 1 - t^k(1 - \frac{d}{b}tg(td)) \end{aligned}$$

and since g is continuous, we can make t sufficiently small that $|\frac{d}{b}tg(td)| < 1$ and hence $|h(td)| < 1$.

4 Full quantifier elimination procedure

It's a simple consequence of the FTA that every complex polynomial splits into linear factors. For in general (not just over the complexes) if $p(a) = 0$ then $x - a$ divides $p(x)$, and so we can proceed by induction on the degree. This has the interesting consequence that the formula:

$$\forall x.\ p(x) = 0 \Longrightarrow q(x) = 0$$

is equivalent to $p \mid q^{\partial(p)}$ where $\partial(p)$ denotes the degree of p and '$|$' the divisibility relation on polynomials. For if we imagine the two polynomials split into linear factors, the assertion becomes:

$$\forall x.\ (x-a_1)\cdots(x-a_n)=0 \Longrightarrow (x-b_1)\cdots(x-b_m)=0$$

That is, each a_i must also be among the b_j and hence each $x-a_i$ must occur among the $x-b_j$. This doesn't imply that q is divisible by p, since there may be repetitions of a_i in p. However there may only be as many as n repetitions where $n=\partial(p)$, so we *do* have $p \mid q^n$. In HOL:

```
|- !p q. (!x. (poly p x = Cx(&0)) ==> (poly q x = Cx(&0))) =
         p divides (q exp (degree p)) \/
         ((poly p = poly []) /\ (poly q = poly []))
```

Note that, as so often when proving theorems formally, we have to note a degenerate case that it's easy to overlook informally, namely that the formula is also true when p and q are identically zero; in this case $\partial(p)=0$ so $q^{\partial(p)}=1$. In keeping with the rest of the development in terms of polynomial functions, we define divisibility in these terms, with `**` denoting the syntactic multiplication operation on coefficient lists.

```
|- p1 divides p2 = ?q. poly p2 = poly (p1 ** q)
```

The equivalence of the quantified implication and divisibility forms the core of the quantifier elimination algorithm. It is a standard result that for full quantifier elimination it suffices to be able to eliminate a single existential quantifier whose scope is a conjunction of equations and negated equations:

$$\exists x.\ p_1(x)=0 \wedge \cdots \wedge p_m(x)=0 \wedge q_1(x)\neq 0 \wedge \cdots \wedge q_p(x)\neq 0$$

for then one can eliminate any quantifier whose scope is quantifier-free by first transforming it to an existential one if necessary by $(\forall x.\ p)\equiv\neg(\exists x.\ \neg p)$, putting the body into disjunctive normal form and then distributing the existential quantifier over the disjuncts. This can then be done repeatedly starting with the innermost quantifier. So we now concentrate on that special case; we apply the indicated transformations in an outer wrapper around the core procedure for a single existential quantifier. Actually we optimize the DNF transformation slightly for the special case $\neg(p\wedge q\vee\neg p\wedge r)=(p\wedge\neg q\vee\neg p\wedge\neg r)$, since as will be seen shortly these "conditionals" often appear in the formulas resulting from quantifier elimination because of case splits.

Note that if there is just one equation and one inequation in the conjunction, then this simply reduces to the consideration of divisibility, because

$$(\exists x.\ p(x)=0 \wedge q(x)\neq 0) \equiv \neg(\forall x.\ p(x)=0 \Longrightarrow q(x)=0)$$

We can also easily deal with two degenerate cases: if there is just one equation, $\exists x.\ p(x)=0$ is, by the fundamental theorem, equivalent to a simple case analysis on the coefficients (either the polynomial is constant zero or nonconstant), and if there is just one inequation, $\exists x.\ q(x)\neq 0$ is equivalent to one of the coefficients being nonzero.

Thus, we just need to reduce the conjunction to the special case of at most one equation and at most one inequation. To reduce the number of equations, we pick the equation of lowest apparent degree $p(x) = ax^n + q(x)$, and perform a case split over whether its leading coefficient a is zero. (If it is simply a constant or can be resolved simply from context, we avoid generating a split, but in general it can be an arbitrary polynomial in other variables.) If we keep getting zero, then eventually we get rid of all the equations. Otherwise, we eventually find an equation with nonzero leading coefficient, and this can be used to perform elimination with the other equations and inequations, reducing their degree. The basic theorem justifying such simplification is trivial:

```
|- (p = Cx(&0)) /\ ~(a = Cx(&0))
   ==> !q b. (q = Cx(&0)) = (a * q - b * p = Cx(&0))
```

where a is the leading coefficient of p and b the leading coefficient of q multiplied by the appropriate power of x to ensure that the subtraction cancels the leading term of q. For example, if $p(x) = yx^2 + 3zx$ and $q(x) = zx^3 + y^2x + 7$ we have $a = y$ and choose $b = zx$, so:

$$\begin{aligned} aq(x) - bp(x) &= y(zx^3 + y^2x + 7) - zx(yx^2 + 3zx) \\ &= (y^3 - 3z^2)x + 7y \end{aligned}$$

Continuing in this way, using the lowest-degree equation each time, we eventually reach a state in which we have either no equations at all or just one. Note that in the latter case the final polynomial equation is actually the GCD of the original set of equations, given the assumptions arising from case splits. Whether or not we have an equation left, we can combine all the inequations into one just by repeatedly using:

$$q_i(x) \neq 0 \land q_{i+1}(x) \neq 0 \equiv q_i(x)q_{i+1}(x) \neq 0$$

Thus we have reached the stage of simplicity where (at most) we just need to compute the divisibility relation of two polynomials in terms of their coefficients. This is quite straightforward, since we can perform a similar kind of elimination based on the following theorem:

```
|- ~(a = Cx(&0))
   ==> p divides p'
       ==> (!x. a * poly q x - poly p' x = poly r x)
           ==> (p divides q = p divides r)
```

and then, when we've finally performed elimination as much as possible, can settle the matter with:

```
|- !p q. p divides q ==> degree(p) <= degree(q) \/ (poly q = poly [])
```

In all cases, the quantified variable is successfully eliminated. Of course, this generally only comes at the cost of increasing the complexity of the polynomials in other variables. However, compared with the more complicated and difficult quantifier elimination procedure for $\mathbb{R}$ described in [12], the efficiency of the present procedure is not calamitously bad. This is to be expected, as the decidability of $\mathbb{R}$ appears to be an inherently more difficult problem [7]. To start with a trivial example[1], which is solved in 4.6 seconds.[2]

```
|- !x y.
       (x pow 2 = Cx (&2)) /\ (y pow 2 = Cx (&3))
       ==> ((x * y) pow 2 = Cx (&6))
```

The following takes 10.5 seconds. It is a genuine 'practical' example in that, when verifying in HOL the polylogarithmic series for π presented in [1] we wanted to prove that $x^2+\sqrt{2}x+1$ has no real roots. The following theorem presents a simple route to that result since as it is universal its restriction to $\mathbb{R}$ also holds, and clearly x^4+1 has no real roots.

```
|- !x a.
       (a pow 2 = Cx (&2)) /\ (x pow 2 + a * x + Cx (&1) = Cx (&0))
       ==> (x pow 4 + Cx (&1) = Cx (&0))
```

Generally, the runtimes increase dramatically with the number of quantifiers. For example, the following, giving a criterion for a general quadratic to have two distinct (complex) roots, takes hours:

```
|- !a b c.
     (?x y. (a * x pow 2 + b * x + c = Cx(&0)) /\
            (a * y pow 2 + b * y + c = Cx(&0)) /\
            ~(x = y)) =
     (a = Cx(&0)) /\ (b = Cx(&0)) /\ (c = Cx(&0)) \/
     ~(a = Cx(&0)) /\ ~(b pow 2 = Cx(&4) * a * c)
```

However, note that although we have hitherto just presented examples of deciding closed formulas, one can simply use it to eliminate some quantifiers, leaving other variables as parameters. Doing so on the LHS of the above example works relatively quickly (25.8 seconds), though the result arising from elimination is not immediately recognizable as equivalent to the simple condition on the above RHS without a bit more manual work, mainly case-splitting over whether variables are zero.

5 Gröbner bases

Although the above procedure is amusing, and satisfactory for some simple problems, it would be nice to have a more efficient procedure. The main reason for the inefficiency

[1] Dedekind once observed when presenting a formal foundation of the real numbers that this "trivial" fact had not hitherto been proved rigorously. Note that our procedure does not make any special optimization but twice applies the standard algorithm presented above.

[2] All timings are on my 366MHz Pentium II laptop, running CAML Light 0.74 under Red Hat Linux 6.0. Note that CAML Light is an interpreted language.

of the general procedure is that when eliminating one quantifier it causes a blowup in the degree of other variables. It would be nicer if we could eliminate a series of like quantifiers $\exists x_1, \ldots, x_n$ *en bloc*. If we can do that, an additional simplification is that we don't need to deal with negated equations, since they can be eliminated at the cost of introducing new quantified variables via:

$$(x \neq y) \equiv \exists z.\ (x - y)z + 1 = 0$$

This transformation is known as the *Rabinowitsch trick*, and is widely used as a theoretical device when proving the full Hilbert Nullstellensatz from a weak special case (see below). Seidenberg [25] also used it when presenting his quantifier elimination procedure for $\mathbb{R}$ and Kapur [15] appears to have been the first to use it in a practical computer implementation.

Although there are general methods that can eliminate a block of existential quantifiers even when the polynomials include other variables, these are complicated. We elected just to implement a special case: proving purely universal formulas by negating them and refuting the (purely existential) negation. As usual, after a DNF transformation this comes down to deciding whether a conjunction of polynomial equations:

$$p_1(x_1, \ldots, x_n) = 0 \wedge \cdots \wedge p_k(x_1, \ldots, x_n) = 0$$

has a solution. Since we are only aiming at a semi-decision procedure, we just need to be able to prove *negative* results of the form 'there is no common solution to this conjunction of equations'. It turns out that the standard Gröbner basis algorithm [27] invented by Buchberger [3] allows us to do this easily.

The Gröbner basis algorithm has a strong resemblance to Knuth-Bendix completion [17], which it predated by several years. The basic idea is to order the monomials $ax_1^{k_1} \cdots x_n^{k_n}$ in some wellfounded fashion and treat a polynomial (i.e. sum of monomials) $m_1 + \cdots + m_p = 0$ as a rewrite rule $m_1 \longrightarrow -(m_2 + \cdots + m_p)$ to be applied to (some monomial of) the other polynomials. Given some basic properties of the monomial ordering such as monotonicity under monomial multiplication, this rewriting relation '$\longrightarrow$' is easily seen to be wellfounded by the multiset ordering, as it replaces a monomial with a finite number of monomials of smaller degree. However it is not in general confluent, and the same polynomial may be reduced to different irreducible forms. Just as with completion, the algorithm proceeds by considering how a 'most general' monomial m can be rewritten in two different ways by other polynomial rewrites, say $m \longrightarrow m_1$ and $m \longrightarrow m_2$, analogous to critical pairs in Knuth-Bendix completion. If these are not joinable by more rewrites, then $m_1 - m_2 = 0$ is itself converted into a new rewrite rule, usually itself generating additional critical pairs, and the process continues. Unlike in the case of completion, it is not hard to show that Buchberger's algorithm always terminates in success, giving a confluent rewrite set known as a *Gröbner basis*. Buchberger's algorithm can in principle be catastrophically inefficient, but in practice it often works very well.

Note that if the initial polynomials are all zero, then so are all the polynomials derived, since they are all of the form

$$q_1(x_1, \ldots, x_n) \cdot p_1(x_1, \ldots, x_n) + \cdots + q_n(x_1, \ldots, x_n) \cdot p_n(x_1, \ldots, x_n)$$

If we end up deriving a nonzero constant polynomial, therefore, it contradicts the fact that the input polynomials have a common zero. One can in fact show that the converse is true: if the input polynomials have no common zero, then a nonzero constant polynomial will be derived. To prove this, however, requires a slightly more delicate analysis of properties of Gröbner bases and a proof of the Hilbert Nullstellensatz for multivariate polynomials. If we wanted to be able to *falsify* universal formulas, we would have to formalize this in HOL; though not difficult in principle it's relatively hard work.

However since we aim only to verify universal formulas, and fail otherwise, all we need is the mathematically trivial observation that a linear combination of polynomials of value zero cannot be equal to a nonzero constant. Thus, we just implement Buchberger's algorithm in a way that records how the new polynomials can be expressed as "linear" combinations (with polynomial coefficients) of the input polynomials. Then at the end we get a sequence of polynomials:

$$q_1(x_1,\ldots,x_n)\cdot p_1(x_1,\ldots,x_n)+\cdots+q_n(x_1,\ldots,x_n)\cdot p_n(x_1,\ldots,x_n)=1$$

and by verifying this in HOL, we trivially derive

$$(\exists x_1,\ldots,x_n.\ p_1(x_1,\ldots,x_n)=0\wedge\cdots\wedge p_k(x_1,\ldots,x_n)=0)=\perp$$

A nice feature of separating the reconstruction of the proof from the details of the algorithm is that, apart from some additional recording, we can implement the standard algorithm with arbitrary optimization techniques. At present, our implementation is fairly naive, but at least implements some of the simpler criteria for avoiding redundant critical pairs. If we really intended to tackle challenging examples, we could tune it in many ways without affecting the later proof reconstruction, since logging is already built into the basic polynomial operations.

The 'coefficients' $q_i(x_1,\ldots,x_n)$ can be quite large, and we should simplify them by further reductions with the initial polynomials. At present we do not do so, but intend to in the future. In this way, it should always be possible to obtain rather compact certificates, suggesting that the Buchberger algorithm can, contrary to the negative remarks made in [12], exploit the finding-checking separation for efficiency. In fact, it's quite striking that the first reasonably efficient fully-expansive decision procedure for linear arithmetic over $\mathbb{N}$, implemented by Boulton [2] used exactly the same kind of certificate to separate search from inference-based proof reconstruction, the only differences being that the p_i were linear polynomials and the q_i constants. As is pointed out in [22], similar certificates can be produced by algorithms that are more efficient on large problems, such as the standard simplex algorithm for linear programming [6]. We consider the ability to separate proof and checking in this way to be fundamentally important in implementing certain (semi)decision procedures both soundly and reasonably efficiently in a fully-expansive context, often much more practical and flexible than the use of so-called 'reflection principles' [10].

In order to verify the final polynomial identity in HOL, we implemented simple routines for maintaining polynomials in canonical sum-of-monomials form ordered according to the monomial ordering used (total degree then inverse lexicographic). Though trivial in principle, this actually took more time to implement than (our naive form of) Buchberger's algorithm itself! We tested the polynomial normalization routines on some identities connected with Waring's problem, taken from [21]. In fact, we discovered that one of them (Theorem 3.3) is stated wrongly:[3]

$$\begin{aligned}(x_1^2+x_2^2+x_3^2+x_4^2)^4 = \; & \tfrac{1}{840}(x_1 \pm x_2 \pm x_3 \pm x_4)^8 + \\ & \tfrac{1}{5040}\Sigma_{1\leq i<j<k\leq 4}(2x_i \pm x_j \pm x_k)^8 + \\ & \tfrac{1}{84}\Sigma_{1\leq i<j\leq 4}(x_i \pm x_j)^8 + \\ & \tfrac{1}{840}\Sigma_{1\leq i\leq 4}(2x_i)^6\end{aligned}$$

The 6^{th} power in the last term on the right should be the 8^{th}, and in the second term, the multiple of 2 should be distributed (separately) over x_j and x_k symmetrically, leading to a 48-way sum. The former is expected from the context, while the latter should be obvious given that all other terms in the sum treat the variables symmetrically. We confess that we consulted the original source [14] before making these trivial observations.

The Gröbner-based method is usually far more efficient on purely universal formulas than the general procedure outlined above, and by more intelligent simplification of the coefficients, it should be possible to further reduce the time spent on the proof-checking phase. For example, the following takes 1.8 seconds, whereas the full quantifier elimination procedure takes 203.2 seconds.

```
|- !a b c x y.
        (a * x pow 2 + b * x + c = Cx(&0)) /\
        (a * y pow 2 + b * y + c = Cx(&0)) /\
        ~(x = y)
        ==> (a * (x + y) + b = Cx(&0))
```

On larger examples, the Gröbner basis procedure is often quite fast where the full quantifier elimination procedure cannot feasibly be applied because it uses too much time or memory.

6 Geometry examples

Note that using Gröbner bases to prove universal formulas doesn't, unlike the full quantifier elimination process, depend on special properties of the complex numbers. The basic procedure would work equally well for $\mathbb{R}$ or even for $\mathbb{Q}$. The only advantage of working over $\mathbb{C}$ is that we can draw negative conclusions from failure (and this only externally and informally without further HOL formalization). But then, if a universal formula is true over $\mathbb{C}$, it's also true over $\mathbb{R}$ and $\mathbb{Q}$, since these are subinterpretations. So given the complex procedure, we can easily derive a real procedure by restriction. The code simply rewrites a formula with homomorphism properties of `Cx` then calls the complex procedure:

[3] The $\pm$ sign implicitly means that all possible combinations of the signs are summed, e.g. $(x_1 \pm x_2 \pm x_3)^2$ means $(x_1+x_2+x_3)^2+(x_1+x_2-x_3)^2+(x_1-x_2+x_3)^2+(x_1-x_2-x_3)^2$.

```
let GROBNER_REAL_ARITH =
  let trans_conv = GEN_REWRITE_CONV TOP_SWEEP_CONV
     [GSYM CX_INJ; CX_POW; CX_MUL; CX_ADD; CX_NEG; CX_SUB] in
  fun tm -> let th = trans_conv tm in
            EQ_MP (SYM th) (COMPLEX_ARITH(rand(concl th)));;
```

Naively it might be considered fairly improbable that interesting universal formulas that are true over $\mathbb{R}$, except those that amount to pure algebraic simplification, would actually turn out to be true in the broader interpretation of $\mathbb{C}$. However, it was a remarkable observation by Wu [28] that a very broad class of geometry problems formulated in terms of coordinates also turn out to be true when the individual x and y coordinates themselves are generalized to $\mathbb{C}$. Wu actually developed his own special decision method related to the *characteristic set* method [23] in the field of *differential algebra* [24]. Although Wu's method is often very efficient and has some particular advantages, it is often possible to apply Gröbner bases with reasonable results. For an excellent comparative survey of the application to geometry theorem proving of the characteristic set method, Gröbner bases and yet another technique called *Dixon resultants*, see [16].

There seems no entirely satisfactory theoretical explanation for just why so many geometrical statements turn out to be true if the natural algebraic formulation is extended to the complex numbers, and any explanation must at present be essentially heuristic. The surprisingly rare cases where this property fails often involve the notion of distance. For example, consider the (true) statement in Euclidean geometry that if three distinct points (x_1,y_1), (x_2,y_2) and (x_3,y_3) all lie on a circle with centre (x_0,y_0), and also all lie on a circle with centre (x'_0,y'_0), then in fact $x'_0 = x_0$ and $y'_0 = y_0$. However, as a moderately lengthy run of the Gröbner basis method will confirm, this statement is not in fact valid over the complex numbers. One counterexample is that the three points are $(2,2i)$, $(3,3i)$ and $(4,4i)$ while the two centres are $(0,0)$ and $(1,i)$. However, we can see that if we explicitly rule out the possibility of the radial distance being zero, the statement becomes valid even over $\mathbb{C}$. Another way to describe this is to say that the line joining the centres to the points on the circumference are *non-isotropic*, i.e. not perpendicular to themselves.

We define certain geometric concepts in HOL in terms of coordinates in a fairly obvious way. A point is represented as a pair of real numbers and geometrical properties are expressed in terms of the x and y coordinates of points using first and second projections `FST` and `SND`. For example:

```
|- collinear a b c =
       ((FST a - FST b) * (SND b - SND c) =
        (SND a - SND b) * (FST b - FST c))

|- is_midpoint b (a,c) =
      (&2 * FST b = FST a + FST c) /\
      (&2 * SND b = SND a + SND c)

|- is_intersection p (a,b) (c,d) =
     collinear a p b /\ collinear c p d
```

Now we code up a trivial tactic GEOM_TAC that expands out definitions and converts points into pairs of coordinates. In conjunction with GROBNER_REAL_ARITH we can now prove some simple geometric theorems. For example, the Centroid theorem is solved in 13.71 seconds (mostly spent in HOL proof reconstruction)

```
|- is_midpoint d (b,c) /\
   is_midpoint e (a,c) /\
   is_midpoint f (a,b) /\
   is_intersection m (b,e) (a,d)
   ==> collinear c f m
```

and Gauss's theorem in 17.01 seconds

```
|- collinear x a0 a3 /\
   collinear x a1 a2 /\
   collinear y a2 a3 /\
   collinear y a1 a0 /\
   is_midpoint m1 (a1,a3) /\
   is_midpoint m2 (a0,a2) /\
   is_midpoint m3 (x,y)
   ==> collinear m1 m2 m3
```

However, the procedure can still take a long time on more difficult problems; theorems involving midpoints tend to be particularly easy since they give rise to linear equations. To make our system into a serious tool for geometry theorem proving would require extensive tuning and experimentation. Moreover, as Wu originally pointed out, many geometric theorems are only true in the presence of certain *nondegeneracy* conditions, e.g. that the three points of a "triangle" are not collinear. Whereas Wu's method can find such conditions automatically, we have not implemented any such automatic method. It often happens that if necessary conditions of this type are left out, not only goes the Gröbner basis method not succeed, but the runtimes are often dramatically worse before failure occurs, making the interactive tracking down of such conditions quite tedious.

7 Conclusions and related work

The complex numbers have been constructed in theorem provers several times. The first machine-checked versions of the Fundamental Theorem of Algebra seems to have been done in Mizar [20]. A completely constructive proof has also been formalized in the Coq prover by a team in Nijmegen.[4]

Several quantifier elimination methods have been coded in HOL as derived inference rules. The subject really starts with the pioneering work of Boulton [2], who presented an implementation of some classic semidecision procedures for universal and existential theories of linear arithmetic over $\mathbb{N}$. This not only showed that fully-expansive implementation of decision procedures could be practical and useful, but was also influential in sharpening the appreciation of how the finding-checking separation, already used e.g. in an early first order prover for HOL [18], can be exploited, a point we have further emphasized here. The first full quantifier elimination method to be implemented as a derived rule seems to have been real quantifier elimination, both linear and nonlinear, by the present author [12]. More recently Norrish, in as yet unpublished work,[5] has implemented a full quantifier elimination procedure for linear arithmetic on $\mathbb{Z}$ based on Cooper's algorithm.

Although this work seems to be the first implementation of Buchberger's algorithm with a fully-expansive proof reconstruction, there have actually been two formal verifications of a version of Buchberger's algorithm by Théry [26] and by Coquand and Persson [5]. Although both verifying algorithms and reconstructing proofs have their strengths and weaknesses, we think the latter approach is quite promising as an approach to actually using the method as a decision procedure in a theorem prover, because it lets us make essentially arbitrary algorithmic and implementation optimizations provided we retain logging. As we have noted, it seems likely that the proof reconstruction could be done much more efficiently if the resulting coefficient polynomials were reduced.

Wu's method has been impressively applied to a huge range of geometry problems by Chou [4], and we have taken some examples and coordinate translations from there. The use of Gröbner bases in geometry theorem proving has been investigated by several researchers, notably Kapur [15]. In comparison, our examples above are quite trivial, but the approach of performing fully-expansive proofs in HOL seems to have potential for two reasons. First, we have greater reliability of results. Second, by working in a general high-level framework such as HOL we are not obliged to be satisfied with treating the geometric concepts as unanalyzed definitions, but can attempt to relate them to more intuitive formalizations, e.g. parallelism in terms of non-intersection. Indeed, it would be interesting to try to relate the coordinate definitions to more traditional definitions in the style of Euclid's *Elements*, and to the modern axiomatic treatment of geometry given by Hilbert [13].

[4] See `http://www.cs.kun.nl/gi/projects/fta/xindex.html`.

[5] See `http://www.cl.cam.ac.uk/ mn200/research`.

Acknowledgements

Thanks to Freek Wiedijk for interesting me in the formalization of the complex numbers in HOL, and hence leading to all this work. I'm also grateful to Joe Hurd for comments and corrections on an early draft which have improved the paper significantly.

References

1. D. Bailey, P. Borwein, and S. Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66:903–913, 1997.
2. R. J. Boulton. Efficiency in a fully-expansive theorem prover. Technical Report 337, University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, UK, 1993. Author's PhD thesis.
3. B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Mathematisches Institut der Universität Innsbruck, 1965.
4. S.-C. Chou. *Mechanical Geometry Theorem Proving*. Reidel, 1988.
5. T. Coquand and H. Persson. Gröbner bases in type theory. In T. Alternkirch, W. Naraschewski, and B. Reus, editors, *Types for Proofs and Programs, International Workshop TYPES'98*, volume 1657 of *Lecture Notes in Computer Science*, pages 33–46. Springer-Verlag, 1998.
6. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
7. L. v. d. Dries. Alfred Tarski's elimination theory for real closed fields. *Journal of Symbolic Logic*, 53:7–19, 1988.
8. H.-D. Ebbinghaus et al. *Numbers*, volume 123 of *Graduate Texts in Mathematics*. Springer-Verlag, 1990. Translation of the 2nd edition of 'Zahlen', 1988.
9. T. Estermann. On the fundamental theorem of algebra. *Journal of the London Mathematical Society*, 31:238–240, 1956.
10. J. Harrison. Metatheory and reflection in theorem proving: A survey and critique. Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, UK, 1995. Available on the Web as `http://www.cl.cam.ac.uk/users/jrh/papers/reflect.dvi.gz`.
11. J. Harrison. Verifying the accuracy of polynomial approximations in HOL. In E. L. Gunter and A. Felty, editors, *Theorem Proving in Higher Order Logics: 10th International Conference, TPHOLs'97*, volume 1275 of *Lecture Notes in Computer Science*, pages 137–152, Murray Hill, NJ, 1997. Springer-Verlag.
12. J. Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998. Revised version of author's PhD thesis.
13. D. Hilbert. *Grundlagen der Geometrie*. Teubner, 1899. English translation 'Foundations of Geometry' published in 1902 by Open Court, Chicago.
14. A. Hurwitz. Über die Darstellung der ganzen Zahlen als Summen von n^{ten} Potenzen ganzer Zahlen. *Mathematische Annalen*, 65:424–427, 1908.
15. D. Kapur. A refutational approach to geometry theorem proving. *Artificial Intelligence*, 37:61–93, 1988.
16. D. Kapur. Automated geometric reasoning: Dixon resultants, Gröbner bases, and characteristic sets. In D. Wang, editor, *Automated Deduction in Geometry*, volume 1360 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
17. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*. Pergamon Press, 1970.

18. R. Kumar, T. Kropf, and K. Schneider. Integrating a first-order automatic prover in the HOL environment. In M. Archer, J. J. Joyce, K. N. Levitt, and P. J. Windley, editors, *Proceedings of the 1991 International Workshop on the HOL theorem proving system and its Applications*, pages 170–176, University of California at Davis, Davis CA, USA, 1991. IEEE Computer Society Press.
19. J. E. Littlewood. Mathematical notes (14): every polynomial has a root. *Journal of the London Mathematical Society*, 16:95–98, 1941.
20. R. Milewski. Fundamental theorem of algebra. *Journal of Formalized Mathematics*, 12, 2000. See `http://mizar.org/JFM/Vol12/polynom5.html`.
21. M. B. Nathanson. *Additive Number Theory: The Classical Bases*, volume 164 of *Graduate Texts in Mathematics*. Springer-Verlag, 1996.
22. G. C. Necula and P. Lee. Proof generation in the Touchstone theorem prover. In D. McAllester, editor, *Automated Deduction — CADE-17*, volume 1831 of *Lecture Notes in Computer Science*, pages 25–44, Pittsburgh, PA, USA, 2000. Springer-Verlag.
23. R. F. Ritt. *Differential Equations from Algebraic Standpoint*, volume 14 of *AMS Colloquium Publications*. American Mathematical Society, 1938.
24. R. F. Ritt. *Differential Algebra*. AMS Colloquium Publications. American Mathematical Society, 1950. Republished in 1966 by Dover.
25. A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.
26. L. Théry. Proving and computing: a certified version of the Buchberger's algorithm. Research report RR-3275, INRIA Sophia Antipolis, 1997.
27. V. Weispfenning and T. Becker. *Groebner bases: a computational approach to commutative algebra*. Graduate Texts in Mathematics. Springer-Verlag, 1993.
28. W. Wen-tsün. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.