Submitted to ACM Symp. Comp. Geom '95

# A Complete and Practical Algorithm for Geometric Theorem Proving

(Extended Abstract)

Ashutosh Rege *

Computer Science Division
University of California
Berkeley, CA 94720

**Abstract**

This paper describes a complete and practical algorithm for the problem of geometric theorem proving. The algorithm works over algebraically closed fields as well as over the reals and takes care of degenerate cases. Our work is motivated by several recent improvements in algorithms for sign determination and symbolic-numeric computation. Based on these, we provide an algorithm for solving triangular systems efficiently using straight-line program arithmetic. The report concludes with a description of an implementation and provides preliminary benchmarks from the same.

## 1 Introduction

In this paper, we describe a practical algorithm (and its implementation) for automatic geometric theorem proving and related problems. Earlier methods for geometric theorem proving were based on a synthetic geometric reasoning approach using natural deduction, forward and backward chaining and the like [5]. Following Wu's seminal work ([13], [14]), a significant amount of the recent work has focussed on an algebraic approach involving determining the feasibility of systems of polynomial equations [2], [8], [7]. Our work is based on this latter methodology and is motivated by recent improvements in algorithms for symbolic-numeric computation with polynomials [9].

The papers by Wu and others showed how a subclass of geometric theorems can be proved by expressing the hypotheses and conclusions as polynomial equations. A geometric theorem can then be proved by essentially determining the algebraic set corresponding to the hypotheses and checking whether the algebraic sets of the conclusions contain it. Different methods have been proposed for doing this - Wu's method involves reduction of the hypotheses system of polynomials to a triangular form (i.e. one in which each successive polynomial has at most one extra variable) and then using successive pseudo-division to determine the feasibility of the conclusion. Other work has focussed on using Gröbner bases to determine whether a system of polynomials (obtained from the original polynomials) has a common zero [8], [7].

Wu's original method is incomplete in the sense that the feasibility of the conclusions is determined over an algebraically closed field whereas geometric theorems are often assertions about real numbers. It is not possible to refute a conjecture over the reals using this approach and in general, confirmation of a theorem can not be obtained automatically from a proof over an algebraically closed field.

*E-mail: rege@cs.Berkeley.edu. Fax: +1-510-642-5775. Supported by a David and Lucile Packard Foundation Fellowship and by NSF P.Y.I. Grant FD93-20588

Another important consideration in geometric theorem proving is the notion of *genericity*. It is necessary to rule out *degenerate* cases such as non-distinct points or zero-radius circles. Wu's method and others generate a set of subsidiary polynomials representing such cases so that the conclusion polynomial vanishes at the common zeros of the hypotheses which are not zeros of the subsidiary polynomials.

From the above discussion, it seems important that a geometric theorem prover should satisfy the following criteria

1. It should work over the reals as well as the complex field.
2. It should handle degenerate cases.
3. It should be practicable.

In this paper we discuss a method for geometric theorem proving which satisfies the first two criteria and based on an initial implementation, seems to be rather efficient in practice.

The basic problem we consider is the following : *given a system of hypotheses polynomial equations in triangular form, determine whether the set of their common zeros over the reals is contained in the set of real zeros of the conclusion polynomial.* (One can use any standard algorithm for triangulating if the hypotheses are not triangular to begin with, see e.g. [3]). The observation here is that, in order to prove a theorem or refute it, one is only interested in the *sign* (+, - or 0) of the conclusion polynomial at the real zeros of the hypotheses polynomials. We provide an efficient method to encode the roots of a triangular system using the Sylvester resultant. To determine the signs, we make use of Sturm sequences which enable us to determine the number of real roots of the hypotheses where the conclusion is non-zero. Thus our algorithm can determine the truth or falsehood of an assertion over the reals.

The problem of genericity is handled by instantiating the independent variables (or parameters) of the given theorem with random values. Probabilistically, this takes care of degenerate cases without affecting the validity of the conclusion.

Computation in the prover is done over the field of reals (or rationals) extended by variables. The numbers in this extension field are then rational functions in these variables. It is, however, very expensive to compute with explicit rational functions. We circumvent this difficulty by using *straight-line programs* or SLP's to represent all intermediate calculations. SLP's are extremely useful for various types of computations with polynomials over field extensions. For example, the cost of multiplying two SLP's is a constant whereas that of explicitly multiplying the two rational functions they represent would be quadratic in the degree. In the algorithms we need for theorem-proving, one is primarily interested in the signs of the coefficients of polynomials over the extension field. These can be determined very easily and efficiently when the computation is done over the field of SLP's.

In order to make our prover more efficient, we use "mixed" arithmetic for algebraic computations : the idea is to represent a number as a structure with two values, the first is the integer value of the number modulo a prime $p$ and the second is a floating-point approximation to that number. We use the finite field part for equality tests and the floating point part for relative ordering of two numbers and sign-determination.

Geometric theorem proving falls in the more general class of problems defined by the first-order theory of the reals. The work described in the subsequent sections is part of ongoing research at Berkeley aimed at developing an efficient practicable toolkit for solving more general problems. We plan on applying the tools and techniques described in this paper to a broader scope of problems including solving systems of polynomial equations, point-location in varieties, existential theory of the reals etc. (See [9] for an overview.) Further, one can use these

techniques to extend our prover to work with polynomial inequalities and also to solve more general problems in constraint-based reasoning.

The paper is organized as follows : Section 2 gives the formulation of the problem of theorem proving we shall use. In section 3 we give an outline of the prover implemented by Chou [3], since we shall use that as a benchmark to compare our prover with. Section 4 gives an overview of our theorem prover. Section 5 sketches the algorithm for solving triangular systems. We conclude with Section 7 which describes the implementation and gives preliminary benchmarks on some problems.

## 2 Geometric Theorem Proving - Formulation of the Problem

This section provides the formulation we shall use of the questions of what constitutes a geometric theorem and what it means to prove it. Several formulations of these two questions have been proposed [5], [6],[8]. See the book by Chou [3] for a discussion of the various formulations and the limitations of each. We will use the formulation used by Chou [3] which seems to be the one that is the most satisfactory :

By a *geometric statement* (S) we shall mean a collection of polynomial equations

$$h_1(y_1, \ldots, y_m) = h_2(y_1, \ldots, y_m) = \ldots = h_n(y_1, \ldots, y_m) = 0$$

called the *hypotheses* and a *conclusion* $g(y_1, \ldots, y_m) = 0$ where the $h_i$ and $g$ are in $K[y_1, \ldots, y_m]$. Here $K$ is some field under consideration, usually $\mathbb{Q}$.

Let $V = V(h_1, \ldots, h_n)$ denote the set of common zeroes in $E^m$, where $E$ is an extension of $K$ Thus an instance of the theorem, i.e. a configuration which satisfies the given hypotheses, corresponds to an element of $V$ and vice versa. Note, however, that $V$ contains the so-called degenerate cases which happen to satifsy the hypotheses. In order to distinguish the degenerate cases from the nondegenerate ones, we use the variables $u_1, \ldots, u_d$, called the *parameters*, among the $y_i$ to denote the nonzero coordinates of those points which can be chosen arbitrarily and we use the variables $x_1, \ldots, x_n$ to denote the rest of the $y_i$. These latter are called ***dependent variables.***

If the extension field $E$ is ***algebraically closed*** a basic fact of algebraic geometry is that every algebraic set has a unique decomposition as a union of irreducible algebraic sets such that no one contains another. Thus the set $V$ can be expressed as a union,

$$V = V_1 \cup \ldots \cup V_s \cup V_1^* \cup \ldots \cup V_t^*$$

where the $V_i$ are all those (*nondegenerate*) components on which the $u_j$ are algebraically independent and the $V_k^*$ are those (*degenerate*) components on which they are dependent. We call the $V_i$ components which are *general for u.*

If $E$ is *not algebraically closed*, the decomposition is, in general, not an irreducible one. However, the $u_j$ are algebraically dependent on the algebraic sets $V_k^*$ and the latter still correspond to the degenerate cases. We will still call the $V_i$ components *general for u.* (Note, however, that in this case any $V_i$ can be reducible or empty.)

We say that a geometric statement (S) is *generally true* if $g$ vanishes on all nondegenerate components of $V$. Else it is *not generally true.* A geometric statement is *generally false* if $g$ vanishes on none of the nondegenerate components of V.

# 3 The Ritt-Wu-Chou Prover

In this section, we provide a brief description of the prover implemented by Chou [3] based on the work of Ritt and Wu. Among all the approaches that have been used and implemented, Chou's prover seems to be most complete and efficient and we will use this prover as the basis of comparison for our prover. This section will therefore try to give a basic knowledge of the algorithms and techniques used by Chou's prover so we can better understand its strengths and limitations.

In 1978, Wu introduced an elegant method for automated theorem proving for algebraic geometry [13]. The basis of his work was a triangulation procedure which was implicit in the work of Ritt [11], [12] and which was given in detail by Wu. Wu's approach was implemented with various modifications and improvements by Chou in his prover [2], [3].

The basic algorithm for the prover implemented by Chou is the following (assuming the notation introduced in the previous section) :

- **Triangulation** The hypotheses are converted into a triangular form

$$f_1(u_1, \ldots, u_d, x_1),\ f_2(u_1, \ldots, u_d, x_1, x_2),\ \cdots,\ f_n(u_1, \ldots, u_d, x_1, \ldots, x_n)$$

- **Successive pseudo-division** Letting $g = R_n$, the remainder $R_0$ is computed by taking successive pseudo-remainders : at the $i^{th}$ step $(i = n \ldots 0)$ the pseudo-remainder computed is $R_i = prem(R_{i+1}, f_{i+1}, x_{i+1})$. Depending on whether $R_0 = 0$ or not, and depending on other conditions (discussed below) the prover can conclude whether the thereom is generally true or not.

We will now discuss the various issues the come up in the process of triangulation and successive pseudo-division and how Chou's prover deals with them.

## 3.1 Triangulation

A triangular system can be obtained rather easily from the original system of hypotheses polynomials by using pseudo-remainders to eliminate variables. A simple algorithm for triangulation is the following (see e.g. [3], [4]) :

Assume we are given a system of hypotheses polynomials as before,

$$h_1(u_1, \ldots, u_d, x_1, \ldots, x_n) = h_2(u_1, \ldots, u_d, x_1, \ldots, x_n) = \cdots = h_n(u_1, \ldots, u_d, x_1, \ldots, x_n) = 0$$

In order to transform it into a triangular form, we first eliminate $x_n$ from $n-1$ polynomials to obtain

$$p_1(u_1, \ldots, u_d, x_1, \ldots, x_{n-1}) = \cdots = p_{n-1}(u_1, \ldots, u_d, x_1, \ldots, x_{n-1}) = p_n(u_1, \ldots, u_d, x_1, \ldots, x_n) = 0$$

We now apply the same procedure to eliminate $x_{n-1}$ from $n-2$ polynomials and so on. The algorithm eliminates $x_n$ by taking pseudo-remainders as required (see [4]).

At the end of this procedure we obtain a system of the form

$$f_1(u_1, \ldots, u_d, x_1) = f_2(u_1, \ldots, u_d, x_1, x_2) = \cdots = f_n(u_1, \ldots, u_d, x_1, \ldots, x_n) = 0$$

## 3.2 Successive pseudo-divsion

Once a triangular form $f_1 \ldots f_n$ is obtained, the next step is to do successive pseudo-division as explained earlier. This results in a final pseudo-remainder $R_0$ which we denote by $prem(g, f_1, \ldots, f_n)$. It is easy to show the following reaminder formula :

$$I_1^{s_1} \cdots I_n^{s_n} g = Q_1 f_1 + \cdots + Q_n f_n + R_0$$

where the $I_j$ are the leading coefficients or *initials* of the $f_j$r; $s_1, \ldots, s_n$ are non-negative integers and $Q_1, \ldots, Q_n$ are polynomials.

If the final remainder $R_0 = 0$ then we can say that $g = 0$ if $I_1 \neq 0, \ldots, I_n \neq= 0$. Thus we get *subsidiary conditions*, namely $I_1 \neq 0, \ldots, I_n \neq= 0$ which are included in the hypotheses of the original geometric statement (S). These subsidiary conditions are usually connected with nondegeneracy.

## 3.3 Problems with the above algorithms

The main problem with the simple triangulation procedure given above is that the variety of the triangular system obtained at the conclusion of the procedure could be strictly larger than the one we started out with due to the process of taking pseudo-remainders. If it turns out that the conclusion polynomial $g$ vanishes on this new larger variety, then the validity of the theorem holds on the smaller variety *a fortiori*. It could easily be the case, however, that $g$ does not vanish on the larger variety. This problem is resolved in the Ritt-Wu-Chou algorithm by testing if the triangular system obtained above is *irreducible* that is, whether each $f_i$ is irreducible in the polynomial ring $K(u_1, \ldots, u_n)[x_1, \ldots, x_i]/(f_1, \ldots, f_{i-1})$. If the system is irreducible and $prem(g, f_1, \ldots, f_n) = 0$, the following can be shown [3]

1. The geometric statement (S) is generally true in all fields
2. For all fields, $(h_1 = 0, \ldots, h_n = 0, I_1 \neq 0, \ldots, I_n \neq 0) \Rightarrow g = 0$ where the $I_k$ are the initials of the $f_k$.

In other words, given an irreducible system, $prem(g, f_1, \ldots, f_n) = 0$ gives us a sufficiency condition for the validity of the theorem in question. For the converse, we first define

$$P = \{\ g \mid g \in K[u, x] \text{ and } prem(g, f_1, \ldots, f_n) = 0\}$$

Given an irreducible system, $prem(g, f_1, \ldots, f_n) = 0$ is a necessary condition for $(S)$ to be generally true if one of the following is satisfied

1. $V(P)$ is of degree $d$.
2. $E$ is an algebraically closed field.
3. The system $f_1, \ldots, f_n$ has a generic point in $E$.

The first question that arises is testing for irreducibility. If it turns out that $deg(f_i, x_i) = 1$ for all $i$, then the triangular system is trivially irreducible and real generic points trivially exist. In this case, $prem(g, f_1, \ldots, f_n) = 0$ gives a necessary and sufficient condition to determine whether the theorem is generally true or not. Several of the simpler theorems in the collection solved by Chou's prover fall in this category ([3], p. 56). If the degree of a polynomial is not 1 in its primary variable, Chou's prover uses factoring over extension fields to check for irreducibility. This approach, however, has been implemented only for *quadratic* extension fields. If the degree of some $f_i$ in $x_i$ is greater than 2, Chou's prover can not handle this case (unless $i = 1$ in which case only factoring over $\mathbb{Q}$ is required). Further, even in the case of quadratic polynomials, the prover works in general only over *algebraically closed fields*. For the prover to work over the *reals*, the existence of a real generic point needs to be shown as required by the sufficiency condition given above. As far as we know, a mechanical approach to this has not been implemented in Chou's prover ([3], p. 56).

Next, we come to the question of what happens when the triangular system is reducible. If this happens, in general Chou's prover uses Ritt's decomposition algorithm to obtain all irreducible components of the original variety. This also involves factoring over field extensions and is computationally expensive for high degrees. Again, this has been implemented only for quadratic cases.

Finally, we address the question of successive pseudo-division. As shown in the earlier section, we have the following remainder formula

$$I_1^{s_1} \cdots I_n^{s_n} g = Q_1 f_1 + \cdots + Q_n f_n + R_0$$

If the remainder $R_0 \neq 0$ then we can conclude that the theorem is not generally true since the right side would be non-zero at any instance of the theorem implying $g \neq 0$ at that instance. If, however, the remainder is zero, $g$ could still be non-zero unless $I_j \neq 0$ for all $j$. These then become the additional subsidiary hypotheses under which the theorem is true. It could be true however that $prem(I_j, f_1, \ldots, f_{j-1}) = 0$ in which the hypotheses would become inconsistent. To avoid this situation, in general, one has to use the complete method in Chou's prover which involves triangulation using the Ritt decomposition algorithm.

To summarize, Chou's prover, though efficient and applicable in the case of a large class of geometric theorems, has the following limitations :

- Applicable in general to systems of degree quadratic or less.
- Does not work in all cases over $\mathbb{Q}$.
- Uses significant amount of computational resources for some problems of degree 2 ([3], p. 69).

# 4 Overview of the Theorem Prover

We will give a schematic overview of our theorem prover in this section. The notation used in this and subsequent sections is the same as introduced earlier. The prover takes as input a set of hypotheses polynomials $h_1, \ldots, h_n$ and a conjecture or conclusion polynomial $g$ in $\mathbb{Q}[u_1, \ldots, u_m, x_1, \ldots, x_n]$. Here, as before, the $x_i$'s represent the dependent variables and the $u_j$'s, the independent variables. In what follows, we will abuse notation and write the hypotheses as polynomials over the $x_i$'s only. It returns "true" or "false" according to whether the theorem is generally true or not over the reals. The prover can also be used to work over $\mathbb{C}$. The modules of the prover can be summarized as follows :

- **Random instantiation of parameters**: The parameters $u_i$ are set to random values to get the hypotheses as polynomials in $x_1, \ldots, x_n$ only.
- **Triangulation :** The hypotheses are first triangulated using the triangulation algorithm described earlier to obtain the polynomials

  $$f_1(x_1), f_2(x_1, x_2), \ldots, f_n(x_1, \ldots, x_n)$$

- **Root representation for triangular systems :** This module takes as input the triangular system

  $$f_1(x_1) = f_2(x_1, x_2) = \ldots = f_n(x_1, \ldots, x_n) = 0$$

  and returns a *symbolic* representation of the roots of the system : the output is a univariate polynomial $p(s)$ and rational functions $r_1(s), \ldots, r_n(s)$. If the roots of $p(s) = 0$ are $\alpha^{(i)} \in \mathbb{C}$, for $i = 1, \ldots, N$, then

  $$\xi^{(i)} = r(\alpha^{(i)})$$

One can think of $r(s)$ as a parametric curve in $\mathbb{C}^n$ which passes through all the roots $\xi^{(i)}$, $i = 1, \ldots, n$. The values of $s$ at which it passes through a solution of $f_1 = f2 = \ldots = f_n$ are precisely the roots of $p(s) = 0$. The fact that we have a symbolic representation of the roots allows us to determine correctly the feasibility of the conjecture polynomial given the hypotheses polynomials.

- **Substitution :** We put $x_i = r_i(s)$, $1 \leq i \leq n$, in the conjecture polynomial $g$. This reduces $g$ to a univariate polynomial $g(s)$.
- **Sign Determination :** This module determines the sign, i.e. +, - or 0, of the instantiated conjecture $g(s)$ at the real roots of $p(s) = 0$ using Sturm sequences.
- **Theorem proving or refutation :** The conjecture should vanish at all of the common real roots of the hypotheses polynomials. Thus the sign determination module should return 0 for the sign of $g(s)$ at every root. If not, there exists a root where the conjecture does not hold and is therefore false OR the triangulation process introduced extra roots. To verify that the latter is not the case, we instantiate the original hypotheses polynomials $h_j$ with the rational functions $r_i(s)$ as we did for $g$. We can now do sign-determination for the entire system $h_1(s), \ldots, h_n(s), g(s)$ at the roots of $p(s) = 0$. This procedure yields for each root of $p(s)$ an ordered sign sequence in $\{+, -, 0\}^{n+1}$ where the $i^{th}$ element in the sign sequence corresponds to the sign of the $i^{th}$ polynomial in $h_1(s), \ldots, h_n(s), g(s)$ at that root of $p(s)$. It is then trivial to check which of the roots of $p(s)$ actually correspond to roots of the original system of hypotheses and based on the signs of $g$ at these correct roots, the prover returns true or false.

# 5 An Algorithm for Solving Triangular Systems

This section sketches the basic algorithm for solving triangular systems used in our geometric theorem prover. Proof of correctness, complexity analysis etc. can be found in the complete paper.

The input to the algorithm is a system of triangular polynomial equations in $\mathbb{Q}(x_1, \ldots, x_n)$,

$$f_1(x_1) = f_2(x_1, x_2) = \cdots = f_n(x_1, \ldots, x_n) = 0$$

Let $\xi^{(i)} \in \mathbb{C}^n$ denote the solutions to the given system.

As described earlier, the algorithm returns a *symbolic* representation of the roots of the system i.e. the output is a univariate polynomial $p(s)$ and rational functions $r_1(s), \ldots, r_n(s)$. If the roots of $p(s) = 0$ are $\alpha^{(i)} \in \mathbb{C}$, for $i = 1, \ldots, n$, then

$$\xi^{(i)} = r(\alpha^{(i)})$$

Our algorithm is based on an approach for computing $(p, r)$ for more general systems due to Renegar [10]. We use his basic method but take advantage of certain properties of triangular systems.

The algorithm proceeds iteratively; at the $i^{th}$ step it adds the polynomial $f_i$ and eliminates the extra variable $x_i$ introduced by the polynomial to get a rational univariate representation $r_1(s_i), \ldots, r_n(s_i)$ of the roots of the first $i$ polynomials. More precisely, at the start of the $i^{th}$ step, the algorithm has computed

$$r_1(s_{i-1}), \ldots, r_{i-1}(s_{i-1}) \text{ and } p_{i-1}(s_{i-1})$$

The algorithm now introduces $f_i(x_1, \ldots, x_i)$. It sets $x_1 = r_1, \ldots, x_{i-1} = r_{i-1}$ in $f_i$. For the new variable, $x_i$, the algorithm sets,

$$x_i = \frac{s_i - l(x)}{l_i} = \frac{s_i - l_0 - l_1 x_1 - \cdots - l_{i-1} x_{i-1}}{l_i}$$

where $x_1, \ldots, x_{i-1}$ are instantiated to $r_1, \ldots, r_{i-1}$ respectively. Thus we obtain from $f_i$ a rational function $g_i$ in $s_{i-1}$ and $s_i$. We denote by $g_{i_n}$ the numerator polynomial of $g_i$ and by $g_{i_d}$ the denominator.

The algorithm now computes the (univariate) Sylvester resultants $R(g_{i_n}, p_{i-1})$ and $R(g_{i_d}, p_{i-1})$ with respect to the variable $s_{i-1}$. We now have :

**Lemma 5.1** *$R(g_{i_d}, p_{i-1})$ divides $R(g_{i_n}, p_{i-1})$ exactly.*

**Proof** Omitted in extended abstract. □

Thus we can set

$$R = \frac{R(g_{i_n}, p_{i-1})}{R(g_{i_d}, p_{i-1})}$$

If the polynomial $l(x) = l_0 + l_1 x_1 + \cdots l_{i-1} x_{i-1}$ is specialized to a random linear polynomial $L$, then with probability one, the values $l(\xi^{(j)})$ will be distinct for distinct $\xi^{(j)}$. The roots of the resultant $R$ will also be distinct. The algorithm sets $p_i(s_i) = R$.

We obtain the functions $r_i$ by differentiating $R$. We define $r_i$ as

$$r_i(s_i) = \left. \frac{\left(\frac{dR}{dl_i}\right)}{\left(\frac{dR}{dl_0}\right)} \right|_{l=L}$$

It can be verified that the $r_i$'s have the correct values at $s_i = \alpha^{(j)}$.

At termination the algorithm eliminates all the variables $x_1, \ldots, x_n$ and returns rational functions $r_j(s)$ in the variable $s = s_n$ and the polynomial $p(s) = p_n(s_n)$.

All of the above computations at the $i^{th}$ step are done with *univariate* polynomials in $s_{i-1}$. The coefficients of the various polynomials are themselves rational functions in $s_i$. However, we represent these with *straight-line programs* as discussed in the introduction which means that the computation time required is the same as that for univariate polynomials. In order to recover the polynomials in $s_i$ required at the $(i+1)^{st}$ step, we interpolate the SLP representation into a polynomial representation. It can then be shown :

**Theorem 5.2** *The complexity of the above algorithm for determining the root representation of a triangular system is $\mathcal{O}(d_1 \ldots d_n)$ where $d_i$ is the total degree of $f_i$. i.e. $\mathcal{O}(d^{\mathcal{O}(n)})$ where $d$ is an upper bound on each $d_i$.*

In contrast the complexity of computing characteristic sets as required by the algorithm of Wu requires time $\mathcal{O}(d^{\mathcal{O}(n^3)})$. Thus, at even relatively small degrees, $d \geq 4$, the computation required by Wu's method can be rather expensive.

# 6 Sign Determination and Theorem Proving

Once we have a symbolic representation $(p, r)$ of the roots of the triangular system of hypotheses, it is relatively straightforward to determine whether the theorem is true or not. The first step is to reduce the conjecture polynomial $c$ to a univariate one by substituting $x_i = r_i(s)$, $1 \leq i \leq n$. This reduces $c$ to a univariate polynomial $g(s)$. We now need to compute the sign of $g(s)$ at the real roots of $p(s)$. We can use Sturm sequences to do that as follows :

If $f(s)$ and $g(s)$ are polynomials, let the Sturm sequence of $f$ and $g$ be denoted $r_0, r_1, \ldots, r_k$, where $r_0(s) = f(s)$, $r_1(s) = g(s)$, and the intermediate remainders are computed via

$$r_{i+1} = q_i r_i - r_{i-1} \tag{1}$$

where $q_i$ is the pseudo-quotient of the polynomial division of $r_{i-1}$ by $r_i$. For a real value $v$, let $\mathrm{SA}(f, g, v)$ denote the number of changes in sign in the sequence $r_0(v), r_1(v), \ldots, r_k(v)$ and $\mathrm{SC}(f, g)$ denote $\mathrm{SA}(f, g, +\infty) - \mathrm{SA}(f, g, -\infty)$. The classical Sturm theorem states that $\mathrm{SC}(f, f')$ equals the number of real roots of $f(s)$. The most general form of the theorem states that $\mathrm{SC}(f, f'g)$ is the number of real roots of $f(s)$ where $g(s) > 0$, minus the number where $g(s) < 0$, common roots making no difference to the count. For our purpose, we consider $\mathrm{SC}(f, f'g^2)$ which counts real roots of $f$ where $g \neq 0$. Thus we compute $\mathrm{SC}(p, p'g^2)$; if this number is 0 the theorem is true. If not, we could have the problem discussed earlier of having introduced extra roots in the triangulation procedure. To check for this eventuality, we instantiate the original hypotheses $h_i$ with the rational functions $r_j(s)$ as described in the overview of the theorem prover. We now have a collection of polynomials $h_1(s), \ldots, h_n(s), g(s)$ whose signs we wish to determine at the roots of $p(s) = 0$. This can be done using the sign-determination algorithm of [1]. We then check the signs of $g$ at the actual common roots of the $h_i$ as described in the overview.

# 7 Implementation

In this section, we briefly describe a first implementation of the above algorithms. Most of the modules are part of a larger project involving the development of a toolkit for solving problems in non-linear algebra. (This toolkit will soon be available in the public domain). The basic modules perform polynomial arithmetic over different fields such as finite fields, rationals, infinitesimal extensions, "mixed" arithmetic and most importantly straight-line programs or SLP's.

## 7.1 Straight-line Programs

All our computations are done over the field of arithmetic straight-line programs. An SLP can be represented by a directed acyclic graph, with each node representing an operation such as addition, and a value. *All the polynomials are thus represented as polynomials over the field of SLP's, i.e. the coefficients are straight-line programs and not numbers.* As mentioned in the introduction, this allows us to represent coefficients which are rational functions in the parameters $u_1 \ldots u_m$. In other words the extension field $\mathbb{Q}(u_1, \ldots, u_m)$ has a 1-1 correspondence with SLP's defined over $u_1, \ldots, u_m$.

Arithmetic with SLP's is easy - for example to add two SLP's one creates a new node with the operation "+" and two edges directed from the new node to the two operand nodes. The value of an SLP node (over the base field such as $\mathbb{Q}$), can be obtained by simply adding the values of its children nodes. Thus evaluation involves a depth-first search of the SLP graph and can be done in time linear in the size of the SLP.

Computing derivatives as required by the algorithm in Section 5 is particularly straightforward, as is determination of the signs of the coefficients of various polynomials (required by the Sturm sequence algorithm). To compute the derivative SLP of an SLP node one simply creates a new SLP using the usual rules of differentiation. This has roughly double the size of the original SLP. For sign determination, one uses successive differentiation and evaluation of SLP's. The details of how this works can be found in the complete version of the paper.

### 7.2 Some benchmarks

We have run our prover on some of the standard problems in geometric theorem proving. Unfortunately, at the time of writing we didn't have all of the geometric theorems in the format accepted by our code. We did run it on the following problems and obtained the times as shown in the following table. All the benchmarks were run on a Sun Sparc 10 machine. The code is written in C. Though very incomplete, the initial benchmarks seem to be rather promising and we hope to have a more complete set of benchmarks very soon. Further, the benchmarks were obtained from a first implementation and we expect to reduce the running time by using various optimizations. (The times given for Chou's prover are from [3] and were obtained on a Symbolics 3600. The geometric statements for these theorems are also from [3].)

| Geometric Theorem | Total degree | Number of variables | Chou's Prover | This Prover r |
|---|---|---|---|---|
| Pappus | 1 | 7 | 1.52s | 0.11 |
| Pappus Dual | 1 | 7 | 1.45s | 0.12 |
| AMS-1 | 1 | 13 | 4.05s | 0.33s |
| Pascal Conic-1 | 1 | 15 | 14.43s | 0.56s |
| Simson | 2 | 11 | - | 0.47s |
| Ptolemy | 8 | 6 | - | 2.9 |
| Steiner | 8 | 17 | - | 33.0 |

# 8 Bibliography


[1] J.F. Canny. An improved sign determination algorithm. In *AAECC-91*, 1991. New Orleans.

[2] S.-C. Chou. Proving elementary geometry theorems using Wu's algorithm. In W.W. Bledsoe and D.W. Loveland, editors, *Theorem Proving : After 25 Years.* American Mathematical Society, 1984.

[3] S.-C. Chou. *Mechanical geometry theorem proving.* Mathematics and its applications. D. Reidel, Holland, 1988.

[4] J. Little D. Cox and D. O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1992.

[5] H. Gelernter. Realization of a geometry theorem proving machine. In E.A. Feigenbaum and J.E. Feldman, editors, *Computers and Thought.* McGraw-Hill, New York, 1963.

[6] D. Kapur. Geometry theorem proving using Hilbert's Nullstellensatz. In *Symposium on Symbolic and Algebraic Computation*, pages 202–208, 1986. Waterloo, Ontario.

[7] D. Kapur. A refutational approach to geometry theorem proving. In D. Kapur and J.L. Mundy, editors, *Geometric Reasoning.* MIT Press, 1988.

[8] B. Kutzler and S. Stifter. On the application of Buchberger's algorithm to automated geometry theorem proving. *J. Symbolic Comput.*, 2:409–420, 1986.

[9] A. Rege and J. Canny. A toolkit for algebra and geometry. Submitted to the *International Symposium on Symbolic and Algebraic Computation*, Montreal, 1995.

[10] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, parts I, II and III. Technical Report 852,855,856, Cornell University, Operations Research Dept., 1989.

[11] J.F. Ritt. *Differential Equations from the Algebraic Standpoint.* Number 14 in AMS Colloquium publications. American Mathematical Society, New York, 1932.

[12] J.F. Ritt. *Differential Algebra.* Number 33 in AMS Colloquium publications. American Mathematical Society, New York, 1950.

[13] W. Wu. On the decision problem and mechanization of theorem proving in elementary geometry. *Sci. Sinica*, 21:150–172, 1978.

[14] W. Wu. Some recent advances in mechanical theorem proving of geometries. In W.W. Bledsoe and D.W. Loveland, editors, *Theorem Proving : After 25 Years.* American Mathematical Society, 1984.