An Application of CLP: Checking the Correctness of Theorems in Geometry

Denis Bouhineau, Laurent Trilling, Jacques Cohen

, , 1–24 ()

An Application of CLP: Checking the Correctness of Theorems in Geometry

DENIS BOUHINEAU denis.bouhineau@irin.univ-nantes.fr
IRIN, University of Nantes, France

LAURENT TRILLING laurent.trilling@imag.fr
LSR-IMAG University of Grenoble, France

AND JACQUES COHEN jc@cs.brandeis.edu
Brandeis University, Waltham, Massachusetts, USA

Keywords: constraints, planar Euclidean geometry, theorem verification, symbolic representation of radicals using rationals, unification.

Abstract. Constraint Logic Programming can be advantageously used to deal with quadratic constraints stemming from the verification of planar geometry theorems. A hybrid symbolic–numeric representation involving radicals and multiple precision rationals is used to denote the results of quadratic equations. A unification–like algorithm tests for the equality of two expressions using that representation. The proposed approach also utilizes geometric transformations to reduce the number of quadratic equations defining geometric constructions involving circles and straight lines. A large number (512) of geometry theorems has been verified using the proposed approach. Those theorems had been proven correct using a significant more complex (exponential) approach in a treatise authored by Chou in 1988. Even though the proposed approach is based on verification -rather than strict correctness utilized by Chou- the efficiency attained is polynomial thus making the approach useful in classroom situations where a construction attempted by student has to be quickly validated or refuted.

1. Introduction

This paper describes a novel application of Constraint Logic Programming (CLP) languages: verifying the correctness of theorems in two dimensional geometry involving straight lines and circles. The well known Constraint Logic Programming Languages (e.g., Prolog III and IV, CLP(R), Chip) can handle the test of satifiability of systems of linear equations describing straight lines. However, the case of circles can only be handled in particular cases where linearization of quadratic equations becomes feasible by resorting to lazy evaluation techniques (*freeze*.)

Furthermore, in the case of theorem verification, it is essential to utilize multiple precision to avoid floating point representations for which equality cannot be resolved without specifying approximations. From the above mentioned CLP languages, Prolog III and IV feature multiple precision rational solutions of systems of linear equations and are therefore appropriate for checking equality of linear terms.

However, in handling quadratic equations generated by the specification of circles, general solutions can only be expressed in terms of square roots, and that mode of expression is unavailable in Prolog III or IV. In Prolog IV one may resort to the use of numeric intervals but again the problem of equality of expressions cannot, in general, be resolved using intervals without approximations.

From a CLP point of view, one of the contributions of this paper is to extend the capabilities of a constraint language using multiple precision and linear equations to the case of the numeric determination of equality between expressions containing radicals (square roots.) Note that the radicals may themselves contain embedded radicals.

What is needed is a "unification-like" algorithm capable of solving quadratic equations and determining if two expressions containing radicals are exactly equal. In this work this is accomplished using a hybrid numerical-symbolic form by expressing radicals in terms of square roots of certain integers. A unification-like recursive algorithm is then used to solve quadratic equations and check precisely the equality of expressions.

Another contribution of this paper is the reformulation of the checking of satisfiability of mixed systems of linear and quadratic equations. By reformulation it is meant the careful generation of quadratic equations and linear equations representing a given theorem. It is shown that – in the case of geometry theorem verification – the checking of correctness using the proposed approach is applicable to all the 512 theorems considered in a classic treatise authored by Shang-Ching Chou [Chou-88] and the vast majority can be handled using only linear equation solving.

Those theorems had been proved "valid" using strictly symbolic manipulation, which is much costlier time-wise than the approach described in this paper. It should be remarked that the purely-symbolic approach used by Chou may yield results that are only valid in a complex domain; on the other hand the approach in this paper only checks the correctness of theorems that are specified using arbitrary numerical values representing the positions of lines and circles.

Presently, there are three approaches for solving problems involving quadratic constraints. These approaches are considered below in decreasing order of complexity and generality.

a. The Gröbner bases method [Kutzler-88]. This approach is the most general, and the most costly computationally. It can handle any polynomial constraints. Its inconvenient is that it may respond affirmatively to the validity of a theorem whose geometrical construction is only well-founded in the imaginary domain.

b. G. Pesant's method [Pesant-95]. This approach is the most general for processing quadratic constraints. It classifies a system of quadratic constraints into several classes including those for which there exist only complex solutions. Nevertheless, there are cases in which one has to resort to the use of approximations and specify a small value establishing the allowable difference between two real numbers that should be equal. In those cases, the equality of two expressions involving radicals can only be done approximately.

c. The proposed method. It can deal with a fairly large class of problems involving a mixture of quadratic and linear constraints. It succeeds in practically all cases in which the lazy evaluation method (used in CLP languages) leads to a failure due to its inability to express radicals. The method's only inconvenient is that, in certain (rare) cases, it may result in an exponential explosion of the number of nested radicals needed to represent a real number.

It is an often the case that one should not use a powerful but costly general algorithm to solve instances of a particular problem for which simpler algorithms exist. That premise is satisfied by adopting approaches b. and c. However, Pesant's method cannot avoid handling approximations. Furthermore, the existence of CLP languages, and the ease of implementation to verify theorems in geometry, amply justifies adopting the proposed approach.

Also notice that, in proving theorems in geometry, it is often assumed that some of the objects can be placed at fixed positions without loss of generality. (Say, one of the circles has its center at coordinates (0,0)) Such procedure is valid even when using the Gröbner basis approach.

The present authors also want to make it very clear that, the verification of the validity of a theorem – bypassing an actual formal proof – is an important topic that is often informally used in a classroom setting. Assume that a student proposes a new construction aiming to prove a given theorem. A quick counter-example usually suffices to redirect the student towards trying another construction. It is in that context that the proposed approach is of greatest value. Also, in that context, it is appropriate to verify a geometric property on a figure before proceeding to a formal proof of that property.

2. Comparison with Existing Approaches

In [Chou-88], a treatise on the automatic proofs of planar geometry theorems, Shang-Ching Chou used exponential algorithmic methods to assert the validity of 512 theorems in that area of geometry.

Chou's approach is based on Wu's methods [Wu-94] that are applicable to combinations of quadratic and linear equations with symbolic coefficients. Chou's method insures that the geometric constructions expressed by those equations result in an equation expressing the property that one wishes to prove. In other words, the equation stating the main property desired in the proof is redundant vis-à-vis the

equations specifying the constructions needed to state the theorem. Basically, the proof corresponds to determining the equality of two formulas containing the variables of the problem. Equivalently, one can replace that problem by one in which the difference of two formulas (a polynomial) is shown to be always equal to zero.

The algorithms used by Chou are purely symbolic (i.e., based on Wu's algorithm) and very likely have worst-case exponential complexity. This is not unusual in algebraic theorem proving, where algorithms may have doubly exponential complexity [Dubé, Yap-94].

The constraints expressing circles and lines are based on those objects being placed in arbitrary positions. Therefore, even straight lines are expressed by quadratic constraints since the variables a and x in the equation $y = ax+b$ are unknown. These equations are referred to as pseudo-linear. Note however that quadratic constraints representing circles do not result in cubic equations because the coefficients of squares are equal to one.

A first step in reducing the complexity of the proof is to assume that circles and straight lines which are used in the construction are placed in positions defined by numeric coordinates. We call this approach geometric theorem checking. It implies that, in some cases, the arbitrary choice of numeric coefficients might result in proving special instead of general cases of a theorem. In other cases the verification of a theorem using a carefully selected numeric example can yield a general proof [Hong-86], and [Deng, Zhang, Yang-90].

Note also that even in the case of using arbitrary numeric coefficients, a resulting failure in proving a theorem corresponds to determining a contradiction which is always useful in detecting the falsity of a conjectured theorem. Such approach is particularly advantageous when using CLP (Intervals) [Benhamou-94].

In the teaching of geometry one can also conveniently use numeric –instead of symbolic– values for the positions of objects in a proof [Allen, Idt, Trilling-93]. In that context both student and teacher are entitled to use numeric coefficients in outlining the constructions pertaining to a proof. The teacher has to insure the correctness of the constructions utilized by students who may use different numerical values. This implies solving numerous sets of possibly redundant quadratic and linear equations with some (but not all) numeric coefficients.

The goal of this paper is to show that, by using carefully designed hybrid numeric-symbolic algorithms, one can escape the curse of exponentiality in checking quadratic constraints of most geometric problems. Of the 512 problems suggested by Chou, the vast majority (487 problems) can be handled using strictly linear equation solving in the realm of rationals. That processing by Gauss-like methods has polynomial complexity. The remaining 25 problems can be solved using the representation detailed in the next section.

The following sections describe: (1) the number representation proposed in this work, (2) a classification of the geometric problems being considered, (3) the pro-

posed algorithm which was implemented using Prolog III as the CLP language of choice, and (4) the strategies for generating the constraints. Examples are interspersed among the various sections. The final sections present the results and include the final remarks.

The reader is referred to the Appendices that contain two illustrative examples. Appendix A illustrates the inadequacy of using a symbolic package, like Maple, in testing for the equality of two expressions containing radicals. Appendix B illustrates the approach utilized by Chou in proving a theorem that is verified using the proposed approach (Section 6.4)

3. Number Representation

3.1. An example

The following example illustrates the problems of using floating point operations to compute values of variables. Consider the expression, [Dubé, Yap-94] :

$$f = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + a/2b$$

where $a = 77617$ and $b = 33096$

The value of f is computed to be:

- 1.172603 in the case of an IBM 370,
- -1.18059e+21 using IEEE double precision,
- -.99999...999998827e+17 using Maple with 20 significant digits,
- -0.83 using Mathematica with a 2 digits accuracy, while operations are performed with 40 digits

The true value of f using the first 20 significant digits is:

$$-0.82739605994682136814116509547981629199 9$$

Obviously, there are great discrepancies among the above representations. Therefore, the problem of checking the equality of two arithmetic terms cannot be done accurately without using some approximation that may well distort the meaning of equality.

The above examples show the inadequacy of floating point representations in accurately testing the equality of two expressions. This problem is particularly accute when attempting to refute a proof of a theorem in geometry. If the equality of two expressions can only be made with a certain degree of precision, then a construction cannot be proved false, especially in the case where substantial rounding errors creep into the computations.

In this work we are interested in representations for arithmetic expressions containing radicals so that the equality between expressions can be determined exactly with a boolean answer "yes", or "no".

3.2. Number representation

The above problem of determining the equality and disequality of square roots of rational numbers is by no means trivial [Landau-92]. It amounts to checking, using a computer, if two numbers are identical. Equality on rationals can be checked if the numbers are expressed with multiple precision. However, when square roots are performed, the resulting floating point representations prevent checking for exact equality. Therefore, an important problem to be dealt with is finding representations of real numbers involving square roots that insure a correct testing for equality or disequality.

Real numbers belonging to the algebraic extension of the rationals Q using a single square root can be expressed by:

$$p + q\sqrt{r}$$

where p and q are rationals and r is not a square. The equality of the real numbers whose representations have the same square roots can therefore be expressed by:

$$p + q\sqrt{r} = p' + q'\sqrt{r}$$

implying that $p = p'$, $q = q'$.

Unfortunately, one cannot fully escape exponentiality using this notation. For example, the real number with double square roots will be expressed by:

$$(p + q\sqrt{r}) + (p' + q'\sqrt{r})\sqrt{r'}$$

containing the four numeric rational coefficients p, q, p', q'. If nested multiple square roots are required, the complexity for storing and processing equality or disequality becomes exponential with the number of square root operations [Bouhineau-97].

So, we have to compute in the quadratic extension $K_0 = Q$, $K_1 = Q[\sqrt{a_0}]$, ..., $K_n = K_{n-1}[\sqrt{a_{n-1}}]$ where K_n is an algebraic extension over K_{n-1} and a_{n-1} is a positive element of K_{n-1} and has no square root in K_{n-1}. Let us denote by SR-var a variable of K_n, i.e., of the form: $p + q\sqrt{r}$ where p, q, and r are either rationals or SR-var's. The acronym SR stands for Square roots involving Rationals.

Note that SR-variables are bound to real constant numbers expressed in a hybrid form of sums and products of rationals which can be "symbolically" multiplied by the square root of an SR number. Therefore, SR-variables are bound to constants, i.e., hybrid representations of rational numbers defined by the following syntax:

<*SR-number*> ::= <*SR-number*> <*op*> <*SR-number*> $\sqrt{<\textit{SR-number}>}$
<*SR-number*> ::= *rational number*
<*op*> ::= + | −

This representation parallels that of complex numbers.

3.3. Arithmetic Operations Involving SR-variables

Operations on K_n are defined recursively from operations in K_{n-1}(proofs appear in [Bouhineau-97].)

3.3.1. Addition

$$(p + q\sqrt{r}) + (p' + q'\sqrt{r}) = (p + p') + (q + q')\sqrt{r}$$

3.3.2. Multiplication

$$(p + q\sqrt{r})(p' + q'\sqrt{r}) = (pp' + qq'r) + (pq' + p'q)\sqrt{r}$$

3.3.3. Negation

$$-(p + q\sqrt{r}) = (-p) + (-q)\sqrt{r}$$

3.3.4. Inverse

$$\frac{1}{p + q\sqrt{r}} = \frac{p}{p^2 - rq^2} - \frac{q}{p^2 - rq^2}\sqrt{r}$$

3.3.5. Sign

$$Sign(p + q\sqrt{r}) = Sign(p) \text{ when } (pq > 0)$$

or else

$$= Sign(p(p^2 - rq^2)) \text{ when } (pq < 0)$$

3.3.6. Square Root

$$\sqrt{p + q\sqrt{r}} = x + y\sqrt{r}$$

when $\Delta = p^2 - rq^2$ has a square root δ in K_{n-1} and $Y = (p + \delta)/2r$ has a square root y in K_{n-1} and then $x = q/2y$,
or else

$$\sqrt{p + q\sqrt{r}} = 0 + 1\sqrt{r'}$$

with $r' = p + q\sqrt{r}$ in $K_{n+1} = K_n[\sqrt{r'}]$

3.3.7. Examples with $a = 2 + 3\sqrt{5}$ and $b = 6 + 2\sqrt{5}$ in $Q[\sqrt{5}]$

- $a + b = 8 + 5\sqrt{5}$
- $ab = 42 + 22\sqrt{5}$
- $-a = -2 - 3\sqrt{5}$
- $1/a = -2/41 + 3/41\sqrt{5}$
- Sign(a) = positive
- Sign(b) = positive
- $\sqrt{a} = \sqrt{2 + 3\sqrt{5}}$ in $Q[\sqrt{5}, \sqrt{2 + 3\sqrt{5}}]$ because a is not a square in $Q[\sqrt{5}]$
- $\sqrt{b} = 1 + \sqrt{5}$

Note that when multiplying two SR-vars one has to perform 5 multiplications. That introduces a 5^m component where m is the number of nested square roots; m should be small, otherwise the computation is becomes too costly. This implies that it is possible to construct unusual examples that require exponential complexity. It will be seen in Section 7 and 8 that these cases do not occur in any of the 512 examples considered by Chou, when processed according to the strategies described in this work.

4. Nature of Geometric Problems

The geometric problems considered in this paper can be classified according to the nature of the objects involved in geometric constructions. Among the two dozen constructions utilized by Chou, there are five that yield quadratic constraints whose number can be reduced by transformations. These five cases are subdivided into two classes.

4.1. Intersection of objects

a. A point belongs to the intersection of a line S and a circle C.

b. A point belongs to the intersection of two circles C_1 and C_2.

Notice that case (4.1.b) of the intersection of two circles can be reduced to the case of the intersection of a line and a circle (4.1.a) by considering the line passing through the intersection of the two circles.

4.2. Other constructions

a. Consider an arbitrary point belonging to a circle C.

b. Construct the bisectrix of an angle specified by the intersection of two lines S_1 and S_2.

c. Given a point P and a circle C, construct the so called inverse point P' (the notion of inverse will be detailed in Section 6).

Many of Chou's problems can be specified using the above classes and result in algebraic representations having a significantly reduced number of quadratic constraints. The above cases could be scrutinized by a transformation algorithm whose goal is to reduce the number of quadratic constraints by replacing, as much as possible, the quadratic constraints by linear ones.

In view of the above, one can describe a geometric problem in terms of a triplet $< Q, L, V >$ where Q is a set of quadratic constraints on many variables including those that represent arbitrary positions of objects (i.e., points, lines and circles); L is a set of pseudo-linear equations in the sense that some of the coefficients of a linear equation defining a straight line may still be unknown; finally, V is a set of numeric values chosen by the user to specify actual values for the variables defining the position of arbitrary values for the objects considered in the problem.

In the next sections, we use the following abbreviations in dealing with the above subcases:

Subcase 1a. Intersection line-circle.

Subcase 1b. Intersection circle-circle.

Subcase 2a. Arbitrary point on a circle.

Subcase 2b. Bisectrix.

Subcase 2c. Inverse point.

5. The Algorithm for Constraint Solving

Note that it is important to use a strategy that postpones (freezes) as much as possible the processing of the quadratic constraints with the hope that their number is reduced by assignments of variables to numeric values using the numeric data supplied by the user and the linear constraints.

This strategy is similar to that used in languages with linear constraints solver such as Prolog III and CLP(R) [Colmerauer-93], [Jaffar, Michaylov, Stuckey, and Yap-92]. However, there is an important difference: the processors for those languages will not be able to handle strictly quadratic constraints that cannot be linearized, whereas those constraints can be handled by the present approach.

The algorithm proper consists of two juxtaposed **while** statements embedded within an external **while** statement that stops the iteration if a final solution has been obtained. The first embedded **while** deals with pseudo-linear equations, the second with quadratic constraints. The algorithm can be described by:

1. **while** a solution is not found **do**
2. **beginloop0**
3. replace the values of V in Q and L
4. (this step may modify Q and L dynamically)
5. **while** there are elements in L **do**
6. **beginloop1**
7. check if an element l of L is of the form of a linear constraint
8. $\sum_{i=1}^{n} m_i x_i + p = 0$ where p and the m_i's are SR-vars
9. **if** that is the case **then**
10. replace x_1 by $(-p - \sum_{i=2}^{n} m_i x_i)/m_1$ in Q and L
11. (Gaussian elimination)
12. update L by removing l, and
13. postpone adding x_1 to V
14. until all x_i's $(2 \leq i \leq n)$ are in V;
15. **exit** by going back to **loop0**
16. **endif**
17. **endloop1;**
18. **while** there are elements in Q **do**
19. **beginloop2**
20. check if an element q of Q is of the form $n_1 X^2 + n_2 X + n_3 = 0$
21. where the n_i's are SR-vars
22. **if** that is the case **then**
23. **begin**
24. solve for X computing $X = p_x + q_x \sqrt{r}$;
25. (X now becomes an SR-var)
26. update Q by removing q, and adding X to V;
27. **exit** by going back to **loop0**
28. **endif**
29. **endloop2**
30. **endloop0**

Notice that in the case of checking redundant constraints, termination takes place with the last constraint in Q or L being of the type $U = U$ in which U is an SR-var. The proofs of correctness for the verification of the equality appear in [Bouhineau-97]. The proof of completeness of this algorithm is insured when V is properly

initialized, because at least one element of Q or L is found within the execution of **loop0**. An additional test can be incorporated to stop the computation if an SR-var has a large specified number of square-root embeddings.

Also notice that a crucial "filtering algorithm", detailed in Section 6, is needed to obtain the triplet $< Q, L, V >$ using the two cases (and subcases) described in the previous section. Recall that the screening is needed to reduce as much as possible the number of elements in Q.

A rough estimate of the complexity disregarding the multiple precision component is as follows. Let m be the number of elements in Q , p the number of elements in L. So the complexity would be roughly of the order of (m^2p^3) assuming that a single SR-var is determined in each execution of **loop0**. The factor m^2 corresponds to the elimination of quadratic constraints in **loop2**. This term corresponds to the worst-case scenario in which a single quadratic equation is the last one to be eliminated each time the **loop2** is executed. In that case the complexity consists of executing the loop: $m + (m-1) + ...1 = O(m^2)$ times. The factor p^3 corresponds to Gaussian elimination.

In what follows we provide the practical details of using a CLP language (Prolog III) to verify a given theorem. The initial triplet $< Q, L, V >$ is input in the form of a list containing sublists that specify the symbolic equations for the quadratic, linear and bound variables pertaining to a given theorem. The above described algorithm is then executed by "asserting" the values of L and V so that the built-in Gaussian elimination algorithm of Prolog III can compute the updated values for V (lines 7 to 14 in **loop1**.)

The **loop2** is executed by inspecting the contents of the sublist Q and adding whenever feasible the new semi-symbolic values for the X's (lines 24 to 26.)

In the remainder of this section we present an example of the input and output of the algorithm in processing a simple theorem. That example is followed by a short subsection providing the arguments for a proof of correctness of the algorithm.

5.1. An Example

Consider the following theorem: Let M and M' be points on a circle C of center I and radius R. Show that D, the perpendicular bisector of $[M, M']$ passes through I.

5.1.1. Direct translation A direct translation of the theorem statement yields the following equations:

$C : I = (0, 0), R = 1$

$M = (X_m, 1/3)$ where (1) $X_m^2 + (1/3)^2 = 1$

$M' = (X_{m'}, 3/4)$ where (2) $X_{m'}^2 + (3/4)^2 = 1$

$D : Y = M_d X + P_d$ where (3) $M_d = (X_m - X_{m'})/(3/4 - 1/3)$

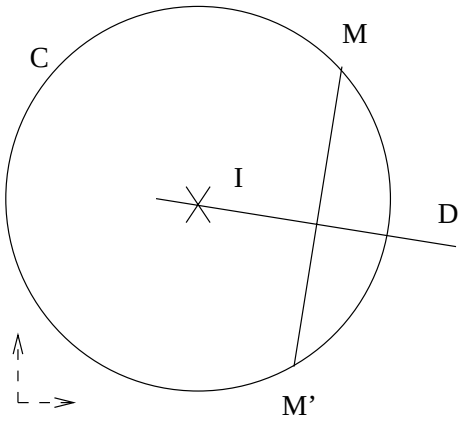

Figure 1 : Perpendicular bisector

and (4) $P_d = 1/2(1/3 + 3/4 - M_d(X_m + X_{m'}))$

One wishes to insure that: (5) $P_d = 0$

The triplet $< Q, L, V >$ becomes: $< \{(1), (2)\}, \{(3), (4)\}, \{(5)\} >$

Note that the coefficients $0, 1, 1/3, 3/4$ .. were arbitrarily chosen to facilitate the reader's understanding of the formulas.

5.1.2. Reformulation The proposed reformulation (see 6.1) of the above theorem results in the triplet:
$C : I = (1, 0), R = 1$
$M = (X_m, Y_m)$ where (1) $Y_m = 4/5X_m$
and (2) $25/4 = Y_m + 5/4X_m$
$M' = (X_{m'}, Y_{m'})$ where (3) $Y_{m'} = 3/2X_{m'}$
and (4) $2 \times 2/3 = Y_{m'} + 2/3X_{m'}$
$D : Y = M_d X + P_d$ where (5) $M_d = (X_m - X_{m'})/(Y_{m'} - Y_m)$
and (6) $P_d = 1/2(Y_m + Y_{m'} - M_d(X_m + X_{m'}))$

One wishes to insure that: (7) $M_d = -P_d$

$< Q, L, V > : < \{\}, \{(1), (2), (3), (4), (5), (6), (7)\}, \{\} >$

The successive values for the triplets according to the proposed algorithm are:
$< Q, L, V > \rightarrow < \{\}, \{(2), (3), (4), (5), (6), (7)\}, \{\} >$
$< Q, L, V > \rightarrow < \{\}, \{(3), (4), (5), (6), (7)\}, \{X_m = \frac{50}{41}, Y_m = \frac{40}{41}\} >$
$< Q, L, V > \rightarrow < \{\}, \{(4), (5), (6), (7)\}, \{X_m = \frac{50}{41}, Y_m = \frac{40}{41}\} >$
$< Q, L, V > \rightarrow < \{\}, \{(5), (6), (7)\}, \{X_m = \frac{50}{41}, Y_m = \frac{40}{41}, X_{m'} = \frac{8}{13}, Y_{m'} = \frac{12}{13}\} >$

$< Q, L, V > \rightarrow < \{\}, \{(6), (7)\}, \{X_m = \frac{50}{41}, Y_m = \frac{40}{41}, X_{m'} = \frac{8}{13}, Y_{m'} = \frac{12}{13}, M_d = -\frac{23}{2}\} >$
$< Q, L, V > \rightarrow < \{\}, \{(7)\}, \{X_m = \frac{50}{41}, Y_m = \frac{40}{41}, X_{m'} = \frac{8}{13}, Y_{m'} = \frac{12}{13}, M_d = -\frac{23}{2}, P_d = \frac{23}{2}\} >$

and (7) becomes the identity $23/2 = 23/2$ thus checking that no inconsistencies are found.

5.2. An Informal Proof of Correctness

The following arguments summarize the proof provided in [Bouhineau-97]:

1. The number of constraints processed within the main loop always decreases.
2. The meaning of the transformed equations is always preserved.
3. The two basic geometrical constructions (intersection line-line, line-circle) are correctly solved.

As a consequence, if there exists a geometrical construction applicable to a given theorem statement that can be directly established using the two basic constructions from that statement, the proposed theorem checker is guaranteed to find it.

6. Strategies for Generating the Constraints

As mentioned above, the strategy is to obtain the smallest number of constraints in Q. As seen in Section 4, the statement of a geometric problem may contain the constructions 1a, 1b, 2a, 2b, 2c that introduce quadratic constraints. This section presents geometric figure transformations allowing to statically reduce the number of quadratic constraints. Let $|S|$ indicate the number of elements in a set S. The transformations allow $|Q|$ to be further reduced dynamically by the algorithm in Section 5. It will be seen in Section 7 that the number of quadratic constraints introducing square roots is significantly reduced in the case of Chou's 512 problems.

The strategies corresponding to the cases and subcases of Section 4 can be described by showing the transformation of an original triplet $< Q, L, V >$ into $< Q', L', V' >$ the latter being the one used by the algorithm of Section 5. Therefore, $|Q'| \leq |Q|$ and there are no constraints on the sizes of the other elements of the triplets. In general though, it is likely that $|L'| \geq |L|$, and $|V'| \geq |V|$.

Four transformations are presented in the sequel. The first two are called local and the last two global. Local transformations are easily automated whereas global are not. Local transformations are those which follow the original order of the statement of a problem for generating the elements in Q, L and V. In contrast, global transformations are those which have to alter the order of generation to fullfill the goal of reducing |Q|. The global transformations are harder to automatize and may

require a shuffling of the order in which statements in Chou's text are considered. They will be described by specific examples.

A word about the potential for automating the global transformations is in order. In that context, one should recall Daniel Bobrow's seminal thesis [Bobrow-68] in which a program reads the statements of simple algebra problems – using a natural language pre-processor – and then translates them into a system of equations. The transformations proposed herein could, in principle, be detected by a pre-processor of natural language which would then generate the corresponding versions of the equations minimizing the number of quadratic components. This, however, is in itself a sizable project beyond the objectives of the present paper.

There is of course a relationship between a given geometric figure F generating the triplet $< Q, L, V >$ and its counterpart F' generating $< Q', L', V' >$. In this presentation we have chosen to depict the figures and the triplets corresponding to the case 2a of Section 4 : arbitrary point on a circle. For the subcase 2c : inverse point, only F and F' are presented; for the remaining cases only F is presented. The reader should have no difficulty in reconstructing the corresponding triplets.

6.1. Arbitrary Point on a Circle

The figures representing F and F' for the subcase 2a namely, –"Consider an arbitrary point M on a circle C"– are presented below with their corresponding triplets. In the figure depicting F, a point having an arbitrary abcissa x is considered and the corresponding value of y is expressed in terms of square roots of x. In contrast, the construction in the figure representing F' considers an arbitrary line D passing through the point R. R' is the symmetric of R with respect to the center of the circle. The projection of R' on D defines the point M on the circle that can be represented by rational numbers.

6.1.1. Figure F for the Subcase 2a : Arbitrary point on a Circle

Coordinates of the objects :
$C : (X_C, Y_C)$,
$R : (X_R, Y_R)$,
$M : (X, Y)$

Triplet $< Q, L, V >$:
$Q : (X_C - X)^2 + (Y_C - Y)^2 = (X_C - X_R)^2 + (Y_C - Y_R)^2$
L : empty
$V : X_C = ..., Y_C = ..., X_R = ..., Y_R = ..., X = ...$
where the ellipses denote a rational number, or in general, an SR-number

6.1.2. Figure F' Arbitrary point on a Circle

Coordinates of the objects :
$C : (X_C, Y_C)$,

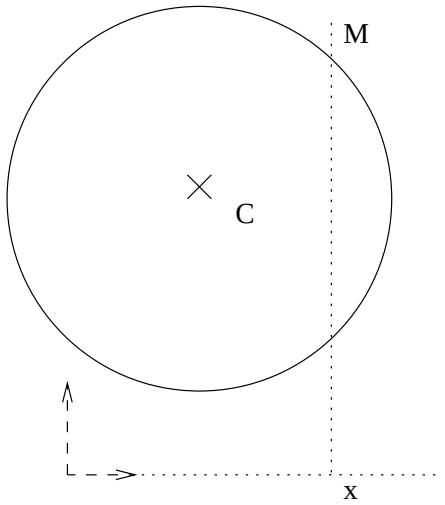

Figure 2 : Arbitrary point on a circle

$R : (X_R, Y_R)$,
$R' : (X_{R'}, Y_{R'})$,
$M : (X, Y)$,
$D : (M_D, P_D)$,
$D' : (M_{D'}, P_{D'})$

Triplet $< Q', L', V' >$:
Q' : empty
$L' : X_R + X_{R'} = 2X_C$
$Y_R + Y_{R'} = 2Y_C$
$Y_R = X_R M_D + P_D$
$M_D M_{D'} = -1$ (perpendicular)
$Y_{R'} = X_{R'} M_{D'} + P_{D'}$
$Y = X M_D + P_D$
$Y = X M_{D'} + P_{D'}$
$V' : X_C = ..., Y_C = ..., X_R = ..., Y_R = ..., P_D = ...$ where the ellipses are SR-numbers.

In this case the previous quadratic constraint is simply eliminated.

Remark that in two cases (2b and 1a) the constructions contain the same objects but those in F' are considered in an ordering that is guaranteed to decrease $|Q'|$ when the algorithm of Section 5 is executed.

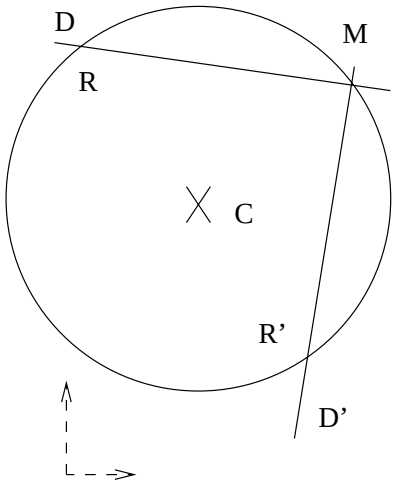

Figure 3 : Arbitrary point on a circle using rational numbers

6.2. Inverse points

This case involves the construction of the inverse of a point N with respect to a circle C. That point is denoted by M and is obtained as follows (Figure 4.) If N is inside the circle C, construct the line D passing through N and the center of the circle. The perpendicular to D from N intersects the circle at the point P. The tangent T to the circle from P intersects D at the desired inverse point M. It can be shown that this construction involves processing square roots.

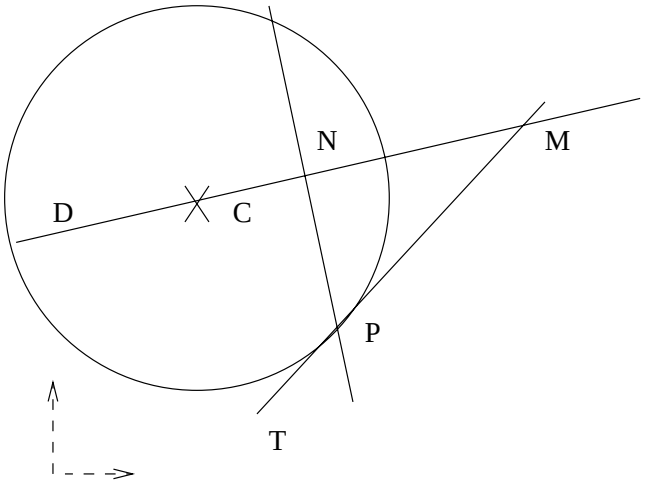

Figure 4 : Construction of the inverse of a point with respect to a circle

The alternate construction representing F' is as follows (Figure 5): consider a horizontal axis H passing through the center of the circle. The inverse point M is obtained by an algebraic computation defined by:

$$X_M = X_C + (X_N - X_C)\frac{R^2}{CN^2} \qquad (1)$$

where R is the radius of the circle and CN is the length of the segment CN.

This computation involves only rationals. Y_M is determined as the point on the line CN having X_M as abcissa.

Proof:

Let us introduce a horizontal line H passing through the center of the circle and consider the angle $\alpha = (NCH)$, $cos(\alpha) = \frac{X_N - X_C}{CN}$, $CN = \frac{X_N - X_C}{cos(\alpha)}$ and $CM = \frac{X_M - X_C}{cos(\alpha)}$ (see Figure 5.)

N and M are inverse with respect to C when :

$$CN \times CM = R^2$$

that is

$$\frac{X_N - X_C}{cos(\alpha)} \times \frac{X_M - X_C}{cos(\alpha)} = R^2$$

since $cos(\alpha) = \frac{X_N - X_C}{CN}$,

$$(X_N - X_C)(X_M - X_C)(\frac{CN}{X_N - X_C})^2 = R^2$$

then N and M are inverse with respect to C when

$$X_M - X_C = (X_N - X_C)\frac{R^2}{CN^2}$$

which is equivalent to (1)

■

6.3. Bisectrices

This subcase appears, for example, in the following theorem [Chou-88] :

Theorem 111: Let D be the intersection of the bisectrix of the angle A in the triangle ABC with the side BC; let E be the intersection of the circle passing through A,B, and C and the line passing through AD. Show that $AB.AC = AD.AE$ (i.e., the ratio of the lenghts AB and AD equals the ratio of the lengths AE and AC)

In the theorem, subcase 2b corresponds to the construction of the bisectrix of the angle A in the triangle ABC. The construction proposed by Chou is quadratic and

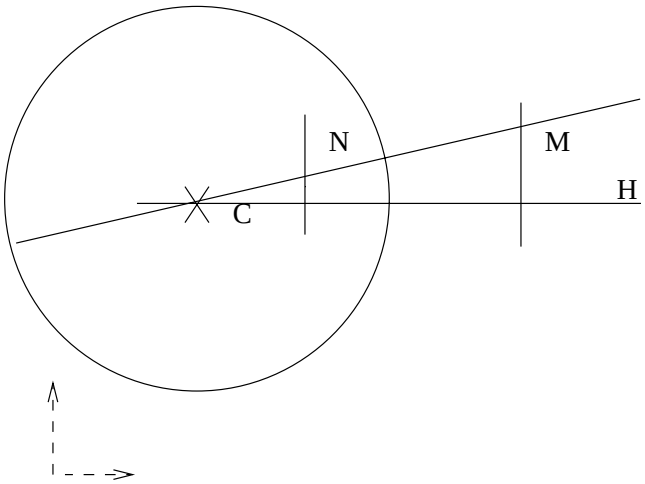

Figure 5 : Construction of the inverse of a point with respect to a circle using rational numbers

follows the constructions enunciated in the theorem. A linear rational version of the theorem is as follows:

Let A, B, and E be three arbitrary points. Construct the line AC symmetrical to AB with respect to AE. One should point out that C is at the intersection of the circle passing through the points A, B, and E, and the line symmetrical to AB with respect to AE. D is the intersection of the lines passing through BC and AE. All these constructions involve linear equations. Remark that the construction assigns, as desired, rational coordinates to C provided that those of the initially given points A, B and E are rational.

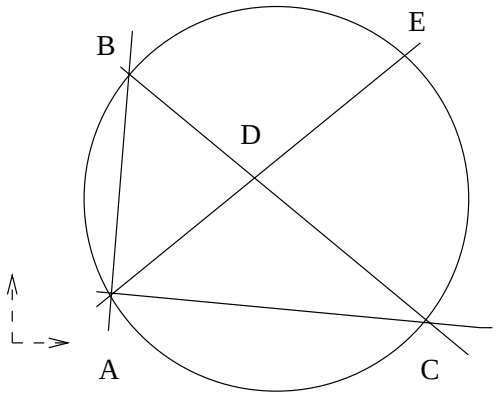

Figure 6 : Construction corresponding to Theorem 111.

6.4. Intersection line-circle

This subcase appears, for example, in the following theorem (see Figure 7):
Theorem 108: Let C be the middle of the arc AB of a circle with center O. D is a point on the circle. (AB) meets (CD) at E. Show that $CA^2 = CE \ . \ CD$.

Subcase 1a corresponds to the construction of C which is the intersection of a circle with center O and the perpendicular bisector of AB. The quadratic solution proposed by Chou consists of considering two arbitrary points A and B, and determining the midpoint M of AB. The perpendicular to AB through M contains a selected point O which is the center of the circle. The point C is the intersection of the circle with the perpendicular.
The linear solution is:

Let A and B be arbitrary points and determine as before the middle point M of AB. Select a point C in the perpendicular to AB passing through M. The point O can then be determined by the intersection of the perpendicular bisectors of the segments AC and CB.

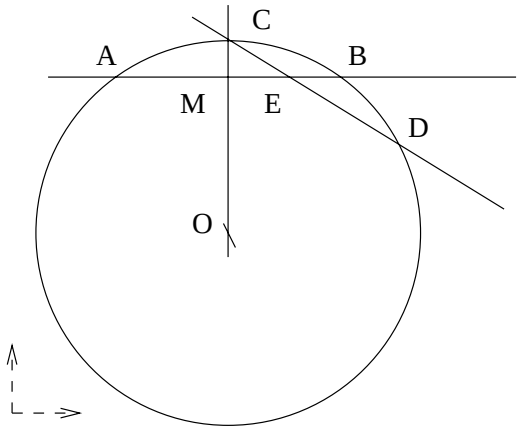

Figure 7: Constructing the middle of an arc (Theorem 108)

7. Results

The results of applying the preprocessing –based on the classification given in Section 4– and the described algorithm are shown in Table A. The contents of the table indicate the nature and the number of problems considered by Chou. The various columns (a total of 12) in that table correspond to the actual number of embeddings in the multiple square root representation of rational numbers. The maximum value 12 is difficult to process in reasonable time due to the space and time complexity involving up to $2^{12} = 4,096$ rational numbers for a single SR-variable.

n	0	1	2	3	4	5	6	7	8	9	10	11	12
$Q(n)$	202	28	80	80	55	35	11	2	8	5	5	0	1
$Max(n)$	354	96	42	15	2	2	1						
$Loc(n)$	441	57	11	3									
$Glob + Loc(n)$	487	25											

Table A

Notation:

$Q(n)$ the number of cases (among the 512 problems) that require the solution of n quadratic constraints without using the transformations in Section 6.

$Max(n)$ the number of cases that require the introduction of n nested square roots without using the transformations in Section 6.

$Loc(n)$ the number of cases that require the introduction of n nested square roots using local transformation for cases 2a: arbitrary point on a circle and 2c: inverse point described in Section 6.

$Glob(n)$ the number of cases that require the introduction of n nested square roots and that benefit from global transformations described in Section 6.

The reduction in complexity due to the preprocessing using the subsets of the cases in Section 4 are shown in the rows of table A. The final row shows that the combined usage of all cases in Section 6 results in reducing the vast majority of the problems proposed by Chou to solving linear equations where the complexity is cubic. The remaining problems are handled using the un-nested square root representation which also results in a polynomial complexity. In 10 of the above 25 problems, $\sqrt{r}$ turns out to be $\sqrt{3}$.

8. Final Remarks

One should first notice that the proposed method has two advantages over the method used by Chou. The first is efficiency, and the second is the capability of the present approach to detect complex solutions when square roots of negative rationals are encountered. Chou's solution checks for the existence of solutions disregarding the fact that some of them are only true in the domain of complex numbers. On the other hand it should be remarked that Chou solves a harder class of problems by considering that objects are placed in arbitrary positions.

Although one cannot in principle extrapolate the results obtained in this work to more complex cases, say of cubics, we feel that a case has been made for using

carefully applied classification rules, described in Section 4, that can be used to reduce the complexity of an exponential problem.

Finally, it should be mentioned that Chou's theorems were originally proposed by humans who can make use of clever constructions to "manually" prove difficult theorems. This may explain why non-exponential solutions are possible. This situation parallels the one that happens when using the simplex method. Even though the worst-case complexity of that method is exponential, most real practical problems are solved in almost linear time.

Appendices

Appendix A presents a short account of an interaction with the Maple symbolic package in trying to establish the validity of an equality involving expressions containing radicals. The results clearly indicate that Maple even provides incorrect results (false) when asked to check for the equality of two symbolic formulas. Nevertheless, when asked to evaluate the two formulas in the case where all the variables are bound to numbers, Maple indicates that the corresponding values are very close to each other. These tests clearly indicate the need for representations like the one proposed in this work.

Appendix B illustrates an example of Chou's approach for the theorem that has been verified using the proposed approach (Section 6.4.) We reiterate that these two approaches are essentially different in the sense that the former attempts to symbolically prove the validity of a theorem, whereas the latter verifies if a given representation of the theorem, using constraints, is satisfiable or not. The former is complex and time consuming, whereas the latter can quickly indicate, for a vast majority of cases, if the constraints are unsatisfiable, thus revealing a false hypothesis or construction by humans –typically students– in attempting to prove a theorem.

Appendix A

The following examples illustrate the inconsistencies of using Maple to check the equality of two symbolic formulas containing radicals (from [Zippel-85]), namely:

$\sqrt{22+2\sqrt{5}}+\sqrt{5}$ and $\sqrt{11+2\sqrt{29}}+\sqrt{16-2\sqrt{29}+2\sqrt{55-10\sqrt{29}}}$

The purpose of the built-in function *evalb* is to force evaluation of expressions involving relational operators, using a three-valued logic system. The returns are true, false, and FAIL. If evaluation is not possible, an unevaluated expression is returned.

The query:

```
evalb(0=sqrt(22+2*sqrt(5))+sqrt(5)
 -sqrt(11+2*sqrt(29))
 -sqrt(16-2*sqrt(29)+2*sqrt(55-10*sqrt(29))));
```

yields the erroneous value false. The correct result would be true or may be even FAIL, the latter admitting the incapacity of Maple to make a decision.

Nevertheless, an attempt to evaluate the difference between the two formulas utilizing purely numerical values (Maple's *evalf*) yields a very small number :

```
evalf(sqrt(22+2*sqrt(5))+sqrt(5)
 -sqrt(11+2*sqrt(29))
 -sqrt(16-2*sqrt(29)+2*sqrt(55-10*sqrt(29))),20);

                  -18
    Result  .1*10
```

Even if a higher accuracy is requested, Maple still finds that the two expressions differ by a minuscule quantity.

```
evalf(sqrt(22+2*sqrt(5))+sqrt(5)
 -sqrt(11+2*sqrt(29))
 -sqrt(16-2*sqrt(29)+2*sqrt(55-10*sqrt(29))),146);

                   -144
    Result  -.1*10
```

(It should be remarked that our version of Maple behaved erratically when asked to further increase the accuracy of the quantity representing the difference between the two given expressions, whose variables were bound to numeric values. We noticed that, in certain cases, an increased specified accuracy would sometimes -but not always- yield the value zero.)

Appendix B

One of the theorems considered by Chou is transcribed below with the generation of the polynomial equations needed to assert its validity.

Theorem 108: Let C be the midpoint of the arc AB of circle center O (see Figure 7.) D is a point on the circle. $E = AB$ inter CD. Show that $CA^2 = CE.CD$

Points A, B are arbitrarily chosen. Points O, M, C, D, E are constructed (in order) as follows: $OA = OB$; M is the midpoint of A and B; $CO = OA$; C is on line OM; $DO = OA$; E is on line AB; E is on line CD. The conclusion is $CA.CA = CE.CD$

Let $A = (0, 0)$, $B = (u_1, 0)$, $O = (x_1, u_2)$, $M = (x_2, 0)$, $C = (x_4, x_3)$, $D = (x_5, u_3)$, $E = (x_6, 0)$

The reader should notice that the coordinates of the circle were taken to be $(0, 0)$ without loss of generality. B is considered with a single arbitrary coordinate.

Therefore, Chou introduces the minimal number of symbolic parameters needed to describe a general initial configuration needed to generate the equations whose satisfiability constitutes the proof.

This is also the case of the proposed approach, except that those parameters are given numerical values. This is consistent with our objectives of quickly verifying the validity of constructions proposed by students attempting to prove theorems in geometry.

The circle has to satisfy :

$$x_1^2 + u_2^2 = (x_1 - u_1)^2 + u_2^2, \tag{1}$$

The point M has to satisfy :

$$x_2 = \frac{u_1}{2}, \tag{2}$$

The point C has to satisfy :

$$(x_1 - x_4)^2 + (x_3 - u_2)^2 = x_1^2 + u_2^2, \tag{3}$$

$$x_2 x_3 + x_4 u_2 = x_1 x_3 + x_2 u_2, \tag{4}$$

The point D has to satisfy :

$$(x_1 - x_5)^2 + (u_2 - u_3)^2 = x_1^2 + u_2^2, \tag{5}$$

The point E has to satisfy :

$$x_4 u_3 + x_4 x_6 = x_5 x_3 + x_6 u_3, \tag{6}$$

The theorem is given by the equation :

$$(x_3^2 + x_4^2)^2 = ((x_3 - u_3)^2 + (x_4 - x_5)^2)((x_4 - x_6)^2 + x_3^2), \tag{7}$$

And the proof corresponds to the elimination of variables $x_6, ..., x_1$ in equation (7) using the equations $(6), ..., (1)$. If the resulting expression of (7) is of the form $0 = 0$ then the theorem is valid.

References

[Allen, Idt, Trilling-93] Richard Allen, Jeanne Idt and Laurent Trilling, *Constrained based automatic construction and manipulation of geometric figures*, Proceedings of the 13th IJCAI Conference, Chambery, Morgan Kaufmann Publishers, Los Altos, August 1993.

[Benhamou-94] Frédéric Benhamou, *Interval Constraint Logic Programming*, Constraint Programming : Basics and Trends, pp 1-21, May , 1994.

[Bobrow-68] Daniel G. Bobrow *Natural Language Input for a Computer Problem Solving System*, Semantic Information Processing, Marvin Minsky, ed., MIT Press, Cambridge, MA, 1968.

[Bobrow-68] Daniel G. Bobrow *A Question-Answerer for Algebra Word Problems*, Semantic Information Processing, Marvin Minsky, ed., MIT Press, Cambridge, MA, 1968.

[Bouhineau-97] Denis Bouhineau,*Construction automatique de figures géométriques & Programmation Logique avec Contraintes*, Thèse de l'Université J. Fourier de Grenoble, France, Juin 1997.

[Chou-88] Schang-Ching Chou, *Large Mechanical Geometry Theorem Proving*, Reidel Publishing, Norwell, 1988.

[Colmerauer-93] Alain Colmerauer, *Naïve Solving of Non-linear Constraints*, Constraint Logic Programming : Selected Research, The MIT Press, Frédéric Benhamou et Alain Colmerauer editors, pp 89-112, 1993.

[Deng, Zhang, Yang-90] Mike Deng, Jingzhong Zhang and Lu Yang, *The parallel numerical method of mechanical theorem proving* Theoretical Computer Science No.74 pp. 253–271, 1990.

[Dube, Yap-94] Thomas Dubé and Chee Yap, *Computing in Euclidean Geometry, The exact computation paradigm*, World Scientific Press, editors D.Z. Du, F.K. Hwang, 7 [Jaffar, Michaylov, Stuckey, Yap-92] J. Jaffar, S. Michaylov, P-J. Stuckey, and R. Yap, *The CLP(R) Language and System*, ACM Trans. on Programming Languages and Systems, Vol. 14, No. 3, pp 339–395, 1992.

[Kutzler-88] Kutzler, Bernhard, *Algebraic approaches to automated geometry theorem proving*, Ph. D thesis, RISC-LINZ, Johannes Kepler Univ., Austria, 1988.

[Landau-92] Susan Landau, *Simplification of nested radicals*, SIAM Journal of Computing Vol 21,No1 pp85-110, Feb 1992.

[Hong-86] Jiawei Hong, *Proving by Example and Gap Theorems*, 107–116, 27th Annual Symposium on Foundations of Computer Science, Toronto, Ontario, Canada, IEEE, Oct. 1986.

[Pesant, Boyer-94] Gilles Pesant and Michel Boyer, *Quad- Clp(R) : Adding the Power of Quadratic Constraints*, Proceedings of the Second Workshop on Principles and Practice of Constraint Programming (PPCP'94), 1994.

[Pesant-95] Gilles Pesant, *Une approche géométrique aux contraintes arithmétiques quadratiques en programmation logique avec contraintes*, Thèse de l'Université de Montréal, 1995.

[Wu-94] Wen-Tsun Wu, *Mechanical Theorem Proving in Geometries*, Springer-Verlag Wien New-York, 1994.

[Zippel-85] Richard Zippel, *Simplification of Expressions Involving Radicals*, J. Symbolic Computation **1**, 1985.