GEOTHER 1.1: Handling and Proving Geometric Theorems Automatically

Dongming Wang

Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie – CNRS
4 place Jussieu, F-75252 Paris Cedex 05, France

Abstract. GEOTHER provides an environment for handling and proving theorems in geometry automatically. In this environment, geometric theorems are represented by means of predicate specifications. Several functions are implemented that allow one to translate the specification of a geometric theorem into English and Chinese statements, into algebraic expressions, and into logic formulas automatically. Geometric diagrams can also be drawn automatically from the predicate specification, and the drawn diagrams may be modified and animated with mouse click and dragging. Five algebraic provers based on Wu's method of characteristic sets, the Gröbner basis method, and other triangularization techniques are available for proving such theorems in elementary (and differential) geometry. Geometric meanings of the produced algebraic nondegeneracy conditions can be interpreted automatically, in most cases. PostScript and HTML files can be generated, also automatically, to document the manipulation and machine proof of the theorem. This report presents these capabilities of GEOTHER.

1 Introduction

We refer to the proceedings of the first three International Workshops on Automated Deduction in Geometry (published as LNAI 1360, 1669, and 2061 by Springer-Verlag in 1997, 1999, and 2001 respectively) and the Bibliography on Geometric Reasoning (`http://calfor.lip6.fr/~wang/GRBib`) for the current state-of-the-art on automated theorem proving in geometry. The construction of theorem provers has been a common practice along with the development of effective algorithms on the subject. The GEOTHER environment described in this report is the outcome of the author's practice for more than a decade. An early version of it was ready for demonstration in 1991, and in 1996 was published a short description of the enhanced version GEOTHER 1.0 [8]. The current version GEOTHER 1.1 provides an environment for handling and proving theorems in geometry automatically. In this environment, geometric theorems are represented by means of predicate specifications. Several functions are implemented that allow one to translate the specification of a geometric theorem into English and Chinese statements, into algebraic expressions, and into logic formulas automatically. Geometric diagrams can also be drawn automatically from the predicate specification, and the drawn diagrams may be modified and animated with mouse click and dragging. Five algebraic provers based on Wu's method of characteristic sets, the Gröbner basis method, and other triangularization techniques are available for proving such theorems in elementary (and differential) geometry. Geometric meanings of the produced algebraic nondegeneracy conditions can be interpreted automatically, in most cases. PostScript and HTML files can be generated, also automatically, to document the manipulation and machine proof of the theorem.

The majority of GEOTHER code has been written as Maple programs, and one can use GEOTHER as a standard Maple package. Some of the GEOTHER functions need external programs written in Java (and previously in C) and interact with the operating system. This concerns in particular the functions for automatic generation of diagrams and documents and the graphic interface, which might not work properly under certain operating systems and Java installations. GEOTHER has been included as an application module in the author's Epsilon library [11] which will be made available publicly in later 2002. The reader will find more information about GEOTHER from `http://calfor.lip6.fr/~wang/GEOTHER`.

There are several similar geometric theorem provers implemented on the basis of algebraic methods. We refer to [1–3] for descriptions of such provers. As a distinct feature of it compared to other provers, GEOTHER is designed not only for proving geometric theorems but also for handling such theorems automatically. Our design and full implementation of new algorithms for (irreducible) triangular decomposition make GEOTHER's proof engine also more efficient and complete. We shall discuss some implementation strategies and report experimental data on the performance of GEOTHER's algebraic provers in an extended version of this report. The present version is restricted to describing the capabilities of GEOTHER.

2 Specification of Geometric Theorems

We recall the following specification of Simson's theorem:

```
Simson := Theorem(
          [arbitrary(A,B,C), oncircle(A,B,C,D), perpfoot(D,P,A,B,P),
           perpfoot(D,Q,A,C,Q), perpfoot(D,R,B,C,R)],
           collinear(P,Q,R), [x5, x6, x7, x8, x9] );
```

This is a typical example of predicate specification, which will be used throughout the report for illustration. In general, a geometric theorem specified in GEOTHER has the following form

```
T := Theorem(H,C,X)
```

where `Theorem` is a predicate specially reserved, `T` is the name, `H` the hypothesis and `C` the conclusion of the theorem, and the optional `X` is a list of *dependent* variables (see below). The hypothesis `H` is a list or set of geometric predicates or polynomials, while the conclusion `C` may be a single geometric predicate or polynomial, or a list or set of predicates or polynomials. Geometric predicates are usually of the form

```
Relation(A,B,C,...)
```

which declares a geometric relation among the objects `A`, `B`, `C`, ... For example, `oncircle(A,B,C,D)` declares that "`the point D is on the circumcircle of the` `triangle ABC` and `collinear(P,Q,R)` declares that "`the three points P, Q and R are collinear.`"

Most of the arguments to the geometric predicates in GEOTHER are points in the plane. In order to perform translation, drawing and proving, coordinates have to be assigned to these points, so that geometric problems may be solved by using algebraic techniques. The assignment of coordinates can be done either manually by the function `Let` or automatically by the function `Coordinate`.

2.1 `Let` and `Coordinate`

- Inputted a sequence `T` of equations $P_1 = [x_1, y_1], \ldots, P_n = [x_n, y_n]$, `Let(T)` assigns the coordinates (x_i, y_i) to each point P_i for $1 \leq i \leq n$.

For instance, the coordinates of the points in Simson's theorem may be assigned by

```
Let(A = [-x1,0], B = [x1,0], C = [x2,x3], D = [x4,x5], P = [x4,0],
    Q = [x6,x7], R = [x8,x9]):
```

A point may be *free* if both of its coordinates are free parameters, or *semi-free* if one coordinate is free and the other is a dependent variable (constrained by the geometric condition), or *dependent* if both coordinates are dependent variables. All the dependent coordinates listed according to their order of introduction are supplied as the third argument `X` to `Theorem`.

- Inputted the predicate specification `T` of a geometric theorem, `Coordinate(T)` reassigns the coordinates of the points appearing in `T` and returns a (new) specification of the same theorem, in which only the third argument (i.e., the list of variables) is modified.

For example, `Coordinate(Simson)` reassigns the coordinates

```
[u1,0], [u2,0], [0,v1], [x1,y1], [x2,y2], [x3,y3], [x4,y4]
```

respectively to the points `A`, `B`, `C`, `D`, `P`, `Q`, `R` in Simson's theorem. The output specification is the same as `Simson` but the third argument is replaced by `[x1, y1, x2, y2, x3, y3, x4, y4]`.

2.2 Algebraic Specification

There are a (limited) number of geometric predicates available in GEOTHER, which are implemented mainly for specifying theorems of equality type in plane Euclidean geometry. The user may add new predicates to the environment if he or she looks into the GEOTHER code and figures out how such predicates may be implemented (which in fact is rather easy). If the user wishes to specify a geometric theorem but cannot find applicable predicates to express the involved geometric relations, he or she may transform the geometric relations (for the hypothesis and conclusion) into algebraic equations manually and provide the set of hypothesis-polynomials to `H` and the conclusion-polynomial or the set of conclusion-polynomials to `C`. For example, the following is an algebraic specification of a theorem in solid geometry:

```
Solid := Theorem( {2*u1*x7-u1*(u2+u4),
          2*(u2-u1)*x7+2*u3*x8-u4*(u2-u1)-u3*u5,
          2*(u4-u2)*x7+2*(u5-u3)*x8+2*u6*x9-u1*(u4-u2),
          (2*u2-u1)*x11-2*u3*x10+u1*u3,
          (u2-2*u1)*x11-u3*x10+u1*u3,
          4*x12-3*x10-u4, 4*x13-3*x11-u5, 4*x14-u6,
          2*x15-u1, 2*u2*x15+2*u3*x16-u2^2-u3^2,
          2*u4*x15+2*u5*x16+2*u6*x17-u4^2-u5^2-u6^2},
         {x13*x15+x7*x16+x8*x12-x7*x13-x8*x15-x12*x16,
          x9*x12+x14*x15+x7*x17-x7*x14-x9*x15-x12*x17},
         [x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17] );
```

Geometric theorems with algebraic specifications may be proved by any of the algebraic provers as well. However, English and Chinese translation, automated diagram generation, and automated interpretation of nondegeneracy conditions will not work for algebraic specifications.

3 Basic Translations

The predicate specification (together with the optional assignment of coordinates by `Let`) is all what is needed for handling and proving the theorem.

3.1 `English` and `Chinese`

- Inputted the predicate specification `T` of a geometric theorem or a geometric predicate `T` with arguments, `English(T)` translates `T` into an English statement.

  For example, `English(Simson)` yields

```
Theorem: If the points A, B, and C are arbitrary, the point D is on
the circumcircle of the triangle ABC, P is the perpendicular foot of
the line DP to the line AB, Q is the perpendicular foot of the line
DQ to the line AC, and R is the perpendicular foot of the line DR to
the line BC, then the three points P, Q and R are collinear.
```

and `English(perpfoot(D, P, A, B, P))` yields:

```
P is the perpendicular foot of the line DP to the line AB
```

- Inputted the predicate specification `T` of a geometric theorem or a geometric predicate `T` with arguments, `Chinese(T)` translates `T` into a Chinese statement.

  For example, `Chinese(Simson)` yields

  定理：设A、B、C为任意点，D点位于三角形ABC的外接圆上，P点为直线DP对于直线AB的垂足，Q点为直线DQ对于直线AC的垂足，且R点为直线DR对于直线BC的垂足，则P、Q、R三点共线.

  and `Chinese(oncircle(A, B, C, D))` yields:

  D点位于三角形ABC的外接圆上

3.2 `Algebraic` and `Logic`

- Inputted the predicate specification `T` of a geometric theorem, or a geometric predicate `T` with arguments, or a sequence `T` of points occurring in the current theorem loaded to the GEOTHER session, `Algebraic(T)` translates `T` into an algebraic specification of the theorem or into one or several algebraic expressions, or prints out the coordinates of the points in `T`.

The output algebraic specification has the same form as the predicate specification `T`, but the geometric predicates are replaced by their corresponding polynomials (or polynomial inequations). Usually, a polynomial means a polynomial equation (i.e., "= `0`" is omitted), and a polynomial inequation is represented by means of "`<> 0`." For example, `Algebraic(Simson)` yields

```
Theorem([2 x1 (x1  x3 - x1  x5 - x5  x3 + x5 x2  - x4  x3 + x5 x3 ),
    x6 x2 + x6 x1 - x4 x2 - x4 x1 + x3 x7 - x3 x5,
    -x3 x6 - x3 x1 + x2 x7 + x1 x7,
    x8 x2 - x8 x1 - x4 x2 + x4 x1 + x3 x9 - x3 x5,
    -x3 x8 + x3 x1 + x2 x9 - x1 x9], -x7 x8 + x7 x4 + x6 x9 - x9 x4,
    [x5, x6, x7, x8, x9])
```

`Algebraic(A,B,C,D,P,Q,R)` prints

```
[-x1, 0], [x1, 0], [x2, x3], [x4, x5], [x4, 0], [x6, x7], [x8, x9]
```

and `Algebraic(not collinear(P,Q,R))` yields:

```
-x7 x8 + x7 x4 + x6 x9 - x9 x4 <> 0
```

- Inputted the specification `T` of a geometric theorem, `Logic(T)` translates `T` into a logic formula.

For instance, the output of `Logic(Simson)` is:

```
(Any D A R B C P Q) [ oncircle(A,B,C,D) /\ perpfoot(D,P,A,B,P) /\
perpfoot(D,Q,A,C,Q) /\ perpfoot(D,R,B,C,R) ==> collinear(P,Q,R) ]
```

The translations explained in this section are simple and straightforward. A more complicated translation discussed in Sect. 6 is to interpret algebraic nondegeneracy conditions geometrically.

4 Proving Geometric Theorems Automatically

The five provers presented in this section are the algebraic proof engine of GEOTHER. They are implemented on the basis of some sophisticated elimination algorithms using characteristic sets, triangular zero decompositions, and Gröbner bases. In fact, the main subroutines of these functions are from the Epsilon modules `CharSets` and `TriSys` and the Maple built-in `Groebner` package. The reader is pointed to the references given below in which the proving methods underlying our implementation are described.

4.1 `Wprover`

– Inputted the specification `T` of a geometric theorem, `Wprover(T)` proves or disproves `T`, with subsidiary conditions provided.

`Wprover` is essentially an implementation of Wu's method as described in [12]. The following is part of a machine proof produced by this prover for the butterfly theorem.

```
GEOTHER> Wprover(Butterfly);

Theorem: If the points A, B, and C are arbitrary, the point O is the
circumcenter of the triangle ABC, the point D is on the circumcircle of
the triangle ABC, the two lines AB and CD intersect at H, the point E
is on the line AC, the line OH is perpendicular to the line EH, and the
two lines EH and BD intersect at F, then H is the midpoint of E and F.

Proof:

char set:   [2 x4 1]   [6 x5 1]   [6 x7 2]   [4 x8 1]   [5 x9 1]
            [2 x10 1]   [8 x11 1]   [4 x12 1]
pseudo-remainder:  [3 x11 1] = -x11 + ... (2 terms)
pseudo-remainder:  [9 x10 1] = x1*x8*x10 + ... (8 terms)
pseudo-remainder:  [9 x9 2] = -x2*x7*x9^2 + ... (8 terms)
pseudo-remainder:  [10 x8 3] = x2^2*x8^3*x7*x5*x3 + ... (9 terms)
pseudo-remainder:  [36 x7 4] = -x7^4*x2^5*x5*x3 + ... (35 terms)
pseudo-remainder:  [22 x5 2] = -2*x3^2*x1*x2^7*x5^2 + ... (21 terms)
pseudo-remainder:  [37 x4 2] = -4*x4^2*x2^9*x1 + ... (36 terms)
pseudo-remainder:  [37 x5 2] = -2*x2^9*x3*x5^2 + ... (36 terms)
pseudo-remainder:  [61 x4 2] = -4*x4^2*x2^11 + ... (60 terms)
pseudo-remainder:  [22 x5 2] = 4*x2^6*x3^3*x5^2 + ... (21 terms)
pseudo-remainder:  [37 x4 2] = 12*x4^2*x2^8*x3 + ... (36 terms)
pseudo-remainder:  [30 x5 2] = 2*x3^2*x2^7*x5^2 + ... (29 terms)
pseudo-remainder:  [62 x4 2] = 4*x4^2*x2^9 + ... (61 terms)
pseudo-remainder:  [19 x5 2] = 4*x3^3*x2^5*x5^2 + ... (18 terms)
pseudo-remainder:  [44 x4 2] = 8*x4^2*x2^7*x3 + ... (43 terms)
pseudo-remainder:  [8 x5 2] = -4*x3^5*x2^2*x5^2 + ... (7 terms)
pseudo-remainder:  [24 x4 2] = -12*x4^2*x2^4*x3^3 + ... (23 terms)
pseudo-remainder:  [16 x5 2] = -4*x3^4*x2^4*x5^2 + ... (15 terms)
pseudo-remainder:  [40 x4 2] = -12*x4^2*x2^6*x3^2 + ... (39 terms)
pseudo-remainder:  [10 x5 2] = -2*x3^4*x2^3*x5^2 + ... (9 terms)
pseudo-remainder:  [27 x4 2] = -4*x4^2*x2^5*x3^2 + ... (26 terms)
pseudo-remainder:  [2 x12 1] = -x12 + ... (1 terms)
pseudo-remainder:  [4 x11 1] = -x7*x11 + ... (3 terms)
pseudo-remainder:  [9 x10 2] = x1^2*x10^2 + ... (8 terms)
pseudo-remainder:  [9 x9 2] = x3*x9^2*x2*x7*x1 + ... (8 terms)
pseudo-remainder:  [29 x8 4] = -x2^3*x8^4*x3*x7*x1 + ... (28 terms)
pseudo-remainder:  [146 x7 5] = -x7^5*x2^7*x3*x1 + ... (145 terms)
pseudo-remainder:  [54 x5 1] = 2*x3^2*x1^2*x2^11*x5 + ... (53 terms)
pseudo-remainder:  [38 x4 2] = 2*x3*x1^2*x2^11*x4^2 + ... (37 terms)
pseudo-remainder:  [96 x5 1] = -2*x2^13*x3*x5 + ... (95 terms)
```

```
    . . . . . .

pseudo-remainder:  [82 x5 1] = 2*x3^2*x2^11*x5 + ... (81 terms)
pseudo-remainder:  [62 x4 2] = 2*x3*x2^11*x4^2 + ... (61 terms)

  The theorem is true under the following subsidiary conditions:

.   the three points A, B and C are not collinear
.   the line AB is not parallel to the line CD
.   the line EH is not parallel to the line BD
.   the line AC is not perpendicular to the line AB
.   the line DB is not perpendicular to the line AB
.   the line OH is not perpendicular to the line CA

QED.
```

The geometric interpretation of algebraic nondegeneracy conditions is done by using the function `Generic` (see Sect. 6).

4.2 `Gprover`

– Inputted the specification `T` of a geometric theorem, `Gprover(T)` or `Gprover(T, Kapur)` proves `T` or reports that it cannot confirm the theorem.

`Gprover` is an implementation of Kutzler–Stifter's method as described in [5] and Kapur's method as described in [4], both based on Gröbner bases. The former is used if the optional second argument is omitted or given as `KS`, and the latter is used if the second argument is given as `Kapur`.

4.3 `CGprover`

– Inputted the specification `T` of a geometric theorem, `CGprover(T)` proves or disproves `T`, with subsidiary conditions provided.

`CGprover` is an implementation of the method proposed in [9]. It first uses the method of Gröbner bases according to Kutzler and Stifter's approach [5] and, if fails, then takes a quasi-basic set of the computed Gröbner basis and perform pseudo-reduction (instead of normal form computation).

4.4 `Tprover`

– Inputted the specification `T` of a geometric theorem, `Tprover(T)` proves or disproves `T`, with subsidiary conditions provided.

`Tprover` implements the method using zero decomposition proposed by the author in [6]. The system of hypothesis-polynomials and inequation-polynomials are fully decomposed into (irreducible) triangular systems. The theorem is proved to be true or false for all the components including degenerate ones.

4.5 `Dprover`

– Inputted the algebraic specification `T` of a (differential geometry) theorem in the local theory of curves, `Dprover(T)` proves or disproves `T`, with algebraic subsidiary conditions provided.

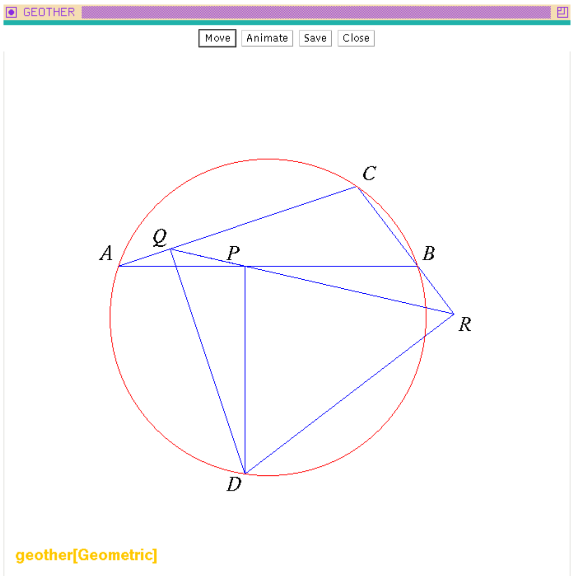
`Dprover` implements the method using ordinary differential zero decomposition proposed in [7]. The system of hypothesis-differential-polynomials (equations and in-equations) are fully decomposed into (irreducible) differential triangular systems. The theorem is proved to be true or false for all the components including degenerate ones.

5 Automated Generation of Diagrams and Documents

5.1 `Geometric`

– Inputted the predicate specification `T` of a geometric theorem, `Geometric(T)` generates one or several diagrams for `T`.

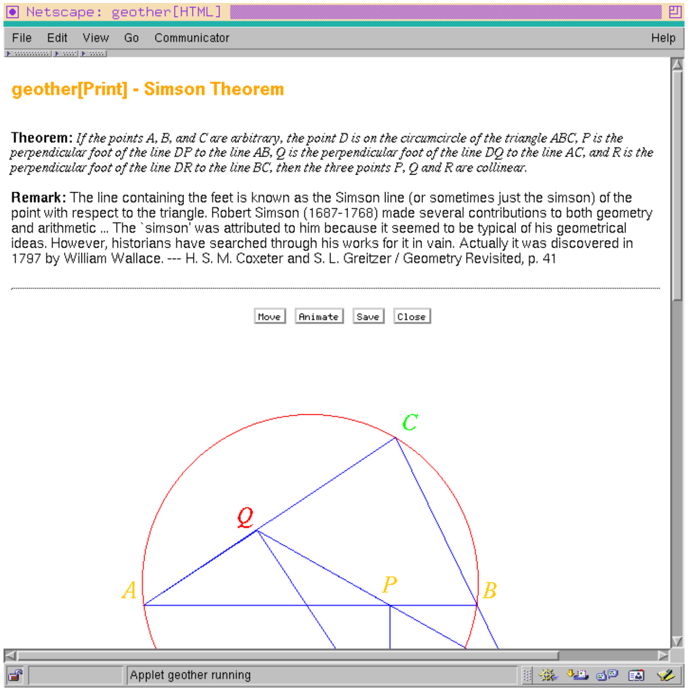
Each generated diagram is displayed in a new window and can be modified and animated by mouse click and dragging. The Maple function `system` is used for the interaction with the operating system. Some implementation details about this drawing function is described in [10]. The following window is an output of `Geometric(Simson)`.

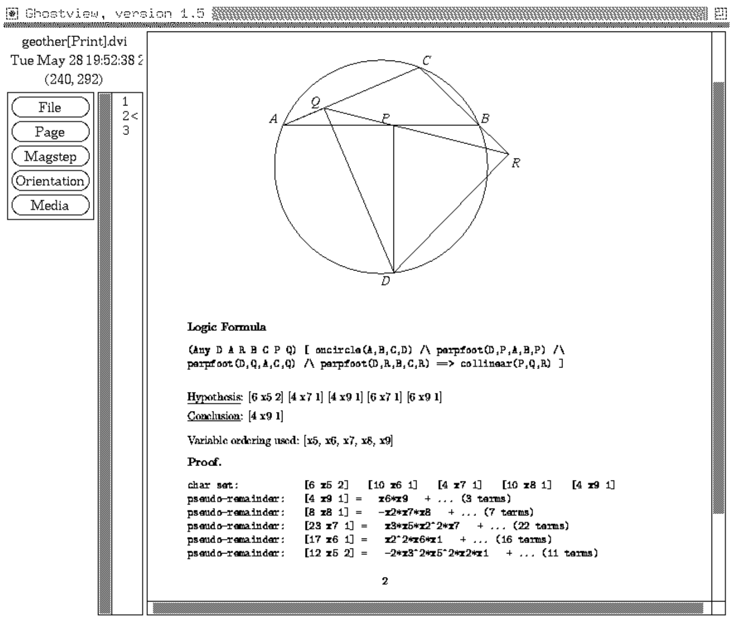


5.2 `Print`

– Inputted optionally the name `N` of the current geometric theorem `T` in the session, `Print(T, N)` generates an HTML file (with Java applet) documenting the last manipulations and proof of `T` and invokes Netscape to view the file. If the optional third argument is given as `LaTeX`, then `Print(T, N, LaTeX)` generates a PostScript file documenting the theorem and invokes Ghostview to view the file.

The following window dumps show parts of the generated documents (with `Print(Simson, 'Simson')` and `Print(Simson, 'Simson', LaTeX))` viewed by Netscape and Ghostview for Simson's theorem.





6 Interpretation of Nondegeneracy Conditions

– Inputted the predicate specification `T` of a geometric theorem and a (simple) polynomial `p` (or a set `p` of polynomials) in the coordinates of the points occurring in `T`, `Generic(T, p)` translates the algebraic condition `p <> 0` with respect to `T` into predicate and geometric form.

This translation does not necessarily work for an arbitrary polynomial `p`. It works only for some simple polynomial whose nonvanishing has a clear geometric meaning with respect to the theorem. When the translation fails, the algebraic condition `p <> 0` is returned. For instance, `Generic(Simson, x1-x2)` results in

```
.   the line AB is not perpendicular to the line BC
                          not perpendicular(A, B, B, C)
```

and `Generic(Simson, x1-2*x2)` returns

```
.   x1-2*x2 <> 0
                                    not x1 - 2 x2
```

The function `Generic` is used to produce nondegeneracy conditions in geometric form stated in English or Chinese. It is implemented according to the following two principles. First, in the routine of each predicate an information entry is included that predetermines possible nondegeneracy conditions associated with this predicate. For example, in the predicate `intersection(A, B, C, D, P)` (meaning "`the two lines AB and CD intersect at P`") the nondegeneracy condition `not parallel (A, B, C, D)` is included. Then for each given predicate specification `T` of a geometric theorem, there is a finite collection *C* of such nondegeneracy conditions in predicate form. For any input `p`, `Generic(T, p)` works by translating all the conditions in *C* into polynomial inequations and then comparing if any of the inequations is equivalent to `p <> 0`. If so, then `p <> 0` is "translated" into the corresponding nondegeneracy condition of predicate form in *C*.

The second principle works by fixing a finite set of predicates corresponding to popular degeneracy conditions such as two points coincide, three points are collinear, and two lines are parallel. For a given specification `T`, `Generic` applies each of these predicates to the points occurring in `T` in a combinatorical way, translates the predicate applied to concrete points into polynomial equation, and verify if the obtained polynomial is equivalent to the input `p`. This is done one by one for different combinations of points and thus involves heuristic search. If an equivalence is discovered, then the translation is done for `p`. This approach is used when the approach according to the first principle fails. It requires extensive search, but the involved computation is not expensive, so it works rather well.

7 Miscellaneous Functions and Remarks

7.1 `Load` and `Search`

– Inputted the name `T` of a geometric theorem, `Load(T)` reads the specification of the theorem from the built-in library to the GEOTHER session.

GEOTHER has a small collection of well-known geometric theorems specified by the author. The user may load any of these theorems to the GEOTHER session and write his or her own specifications for interesting geometric theorems (in plane Euclidean geometry). To see which theorems are contained in the library, one may use the following function.

– Inputted a string `S`, `Search(S)` lists all the names of theorems containing `S` in the library.

If `S` is given as `all` or `All`, then all the theorems in the library are listed.

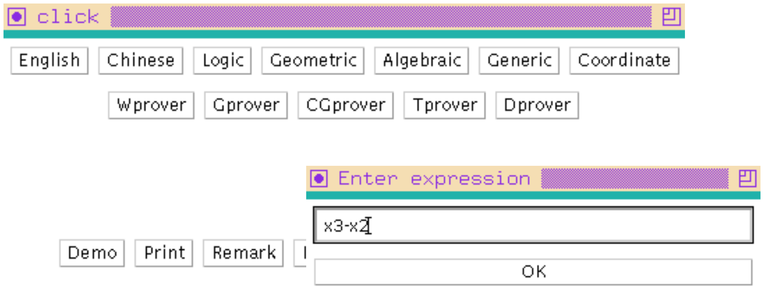
7.2 `Demo` and `Click`

– `Demo()` gives an automated demonstration of GEOTHER.

During the demonstration, GEOTHER commands appear automatically and the user only needs to enter semicolon `;` and then return.

– `Click()` starts a mouse-driving GEOTHER interface.

The following windows show the interface, with which the user does not need to type in commands except for the name of a theorem, a string, or a polynomial when calling `Load`, `Search`, or `Generic` respectively.



Click on a button causes an application of the indicated function to the current theorem (last loaded) in the session. This interface works only under Unix and Linux, of which the pipe facility is used for data communication.

7.3 Online Help

GEOTHER provides online help for all its functions with examples. This help facility is used in the usual way as for standard Maple packages. This report and part of a forthcoming book by the author will serve as an introduction to GEOTHER. Additional documents and future updates of GEOTHER will be made available on the author's web page.

7.4 Remarks

The implementation of most GEOTHER functions is easy and straightforward, but as usual it is a time-consuming task and requires special care to handle all the technical details at the coding level. We do not enter into these details. Since a predicate or algebraic specification of a geometric theorem may be a false proposition in the logical sense, proving the theorem is not simply a yes or no confirmation. Often we want and try to produce a proof of the theorem, even if its specification is not correct logically, by imposing certain subsidiary conditions. This makes the proving problem much harder, in particular, when a *bad* nonsensical specification is submitted to the prover. In this case, a nonsensical proof of the "theorem" may be produced. This is not the fault of our provers because determining whether a meaningful theorem can be derived from an arbitrary specification is much beyond the scope of the provers. Therefore, the user must make sure that his or her specification of the geometric theorem is correct, or at least *almost* correct. If a nonsensical proof is produced, we advise the user to check the specification carefully, or try to use a different specification.

Moreover, GEOTHER functions for manipulation and proof are made as automatic as possible. To achieve this, we have adopted a number of heuristics at the implementation level. These heuristics may fail in certain cases, and thus some functions may not work or work unpleasantly for some specifications of theorems. It is completely normal if the user gets an automatically drawn diagram that has a poor look or even looks bizarre. GEOTHER is not yet a perfect piece of software, but it shows how geometric theorems can be handled and proved automatically and nicely.

Acknowledgements. The author wishes to thank Roman Winkler and Xiaofan Jin who have contributed to the implementation of `Coordinate` and `Generic` respectively. Part of this work is supported by the SPACES project (`http://www.spaces-soft.org`) and the Chinese national 973 project NKBRSF G19980306.

References

1. Chou, S.-C., Gao, X.-S., Liu, Z., Wang, D.-K., Wang, D.: Geometric theorem provers and algebraic equation solvers. In: *Mathematics Mechanization and Applications* (X.-S. Gao and D. Wang, eds.), pp. 491–505. Academic Press, London (2000).
2. Gao, X.-S., Zhang, J.-Z., Chou, S.-C.: *Geometry Expert* (in Chinese). Nine Chapters Publ., Taiwan (1998).
3. Hong, H., Wang, D., Winkler, F.: Short description of existing provers. *Ann. Math. Artif. Intell.* **13**: 195–202 (1995).
4. Kapur, D.: Using Gröbner bases to reason about geometry problems. *J. Symb. Comput.* **2**: 399–408 (1986).
5. Kutzler, B., Stifter, S.: On the application of Buchberger's algorithm to automated geometry theorem proving. *J. Symb. Comput.* **2**: 389–397 (1986).
6. Wang, D.: Elimination procedures for mechanical theorem proving in geometry. *Ann. Math. Artif. Intell.* **13**: 1–24 (1995).
7. Wang, D.: A method for proving theorems in differential geometry and mechanics. *J. Univ. Comput. Sci.* **1**: 658–673 (1995).

8. Wang, D.: GEOTHER: A geometry theorem prover. In: *Automated Deduction* (M. A. McRobbie and J. K. Slaney, eds.), LNAI 1104, pp. 166–170. Springer-Verlag, Berlin Heidelberg (1996).
9. Wang, D.: Gröbner bases applied to geometric theorem proving and discovering. In: *Gröbner Bases and Applications* (B. Buchberger and F. Winkler, eds.), pp. 281–301. Cambridge University Press, Cambridge (1998).
10. Wang, D.: Automated generation of diagrams with Maple and Java. In: *Algebra, Geometry, and Software Systems* (M. Joswig and N. Takayama, eds.), to appear. Springer-Verlag, Berlin Heidelberg (2002).
11. Wang, D.: Epsilon: A library of software tools for polynomial elimination. In: *Proc. Int. Congr. Math. Softw.* (ICMS 2002) (Beijing, China, August 17–19, 2002), to appear. World Scientific, Singapore River Edge (2002).
12. Wu, W.-t.: *Mechanical Theorem Proving in Geometries: Basic Principles* (translated from the Chinese by X. Jin and D. Wang). Springer-Verlag, Wien New York (1994).