2J008 - Bachelorproject
Automatic Geometric Theorem Proving

Dan Roozemond - 0492344
Eindhoven University of Technology
Industrial and Applied Mathematics

9th July 2003

Contents

Chapter 1

Introduction

This Bachelor's project was performed in the scope of *Automatic Geometric Theorem Proving.* It is part of a project that in the end should make it possible to put forward a geometric theorem by clicking and pointing. This theorem should then be tested automatically by means of, among others, Gröbner Bases. The software to be used can be divided in three parts:

Cinderella "Software for doing geometry on the computer, and it is designed to be both mathematically robust and easy to use" [1].

Singular "A Computer Algebra System for Polynomial Computations" [5].

OpenMath "A new, extensible standard for representing the semantics of mathematical objects" [4] - The communication between Cinderella and Singular will be implemented with OpenMath.

A schematic overview of the process can be found in Figure 1.1. This Bachelor's project focuses on Steps 3 and 4.

In this report first a summary of parts of the course titled 'Algebra 3' [7] is given, in order to refresh the knowledge on rings, ideals, and Gröbner Bases. Via some practical examples in Chapters 4 and 5 we develop an algorithm for proving such theorems. This algorithm is intended for an implementation in Singular, see Chapter 6 and Appendix B.

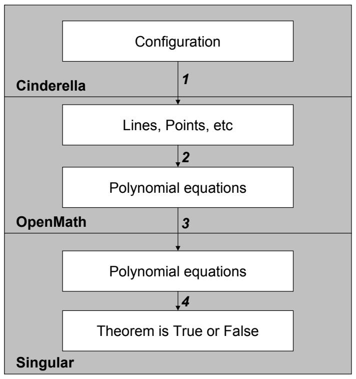

Figure 1.1: The process of Automatic Geometric Theorem Proving

Furthermore, the work done on the connection between Singular and the OpenMath standard can be found in Chapter 7. Finally, the conclusion can be found in Chapter 8.

Chapter 2

Polynomials, Gröbner bases and Buchberger's algorithm

Below a summary of parts of the course titled Algebra 3 [2] is given, in order to refresh the knowledge on rings, ideals and, mainly, Gröbner Bases.

2.1 Polynomial rings and systems of polynomial equations

Let $k[X_1, \dots, X_n]$ be a polynomial ring in n indeterminates over the field k.

Definition 2.1.1 *A ring R is* Noetherian *if every ideal in R is finitely generated.*

Lemma 2.1.2 *A ring R is Noetherian if and only if every ascending chain of ideals $I_1 \subset I_2 \subset I_3 \subset \dots$ in R stabilizes.*

Theorem 2.1.3 [Hilbert]*: If R is Noetherian, then so is $R[X]$.*

Corollary 2.1.4 *Every polynomial ring over a field is Noetherian. If I is an ideal in such a ring then there exist elements $f_1, \dots, f_s \in I$ such that $I = (f_1, \dots, f_s)$.*

With this corollary the problem of finding zeros of a system of polynomial equations is equivalent to the problem of finding the common zeros of an ideal.

2.2 Monomial Orderings

The *Lexicographic Order*, *Graded Lex Order* and *Graded Reverse Lex Order* are introduced, and so are the definitions of multideg(f), lt(f), lm(f) and lc(f).

2.3 A division algorithm

Below an algorithm to divide p by $f_1, \dots, f_s$ is presented.

0. $r = 0, q_i = 0$ $(i = 1, \dots, s)$

1. Look for the **first** polynomial among the f_i with $\mathrm{lt}(f_i)|\mathrm{lt}(p)$.
   If such an f_i exists, set:
$$p := p - \frac{\mathrm{lt}(p)}{\mathrm{lt}(f_i)} f_i, \qquad \text{and} \qquad q_i := q_i + \frac{\mathrm{lt}(p)}{\mathrm{lt}(f_i)} \tag{2.1}$$
   If such an f_i does not exist, set:
$$p := p - \mathrm{lt}(p), \qquad \text{and} \qquad r := r + \mathrm{lt}(p) \tag{2.2}$$
2. If $p = 0$ stop, if not go back to 1.

Termination is guaranteed, since multideg$(\mathrm{lt}(p))$ decreases in each step. Upon termination we have $f = q_1 f_1 + \ldots + q_s f_s + r$. Note that in general the result depends on the order of the f_i.

2.4 Monomial ideals and Gröbner Bases

Definition 2.4.1 *A monomial ideal in $k[X_1, \ldots, X_n]$ is an ideal generated by monomials.*

Lemma 2.4.2 *Let $A \subseteq \mathbb{N}^n$, and let I be a monomial ideal, generated by $\mathbf{X^a}$, $a \in A$.*

1. *$f \in I$ if and only if every term of f is in I;*
2. *$X^b \in I$ if and only if $X^a | X^b$ for some $a \in A$;*
3. *$I = (X^{a(1)}, \ldots, X^{a(m)})$ for some m.*

Definition 2.4.3 *The leading term ideal is the ideal $(\mathrm{lt}(I))$ generated by $\mathrm{lt}(I)$, where $\mathrm{lt}(I) := \{\mathrm{lt}(f) \mid f \in I, f \neq 0\}$.*

Lemma 2.4.4 *If $I \subset k[X_1, \ldots, X_n]$ is an ideal, $I \neq (0)$, then $(\mathrm{lt}(I))$ is a monomial ideal and there exist $f_1, \ldots, f_s \in I$ such that $\mathrm{lt}(f_1), \ldots, \mathrm{lt}(f_s)$ generate this ideal.*

Definition 2.4.5 *A subset $\{g_1, \ldots, g_s\}$ of I is a* Gröbner Bases *of I if*
$$(\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_s)) = (\mathrm{lt}(I)).$$

Theorem 2.4.6 *Let $I \neq (0)$ be an ideal in the polynomial ring $k[X_1, \ldots, X_n]$. Then:*

1. *I has a Gröbner basis.*
2. *If $\{g_1, \ldots, g_s\}$ is a Gröbner basis of I then $(g_1, \ldots, g_s) = I$.*
3. *If $\{g_1, \ldots, g_s\}$ is a Gröbner basis of I then division by $g_1, \ldots, g_s$ leaves a **unique** remainder independent of the order of the q_i. The remainder is the unique polynomial r such that:*
   (a) *Either $r = 0$ or no term of r is divisible by any $\mathrm{lt}(g_i)$;*
   (b) *$f - r \in I$.*

2.5 Buchberger's algorithm

Proposition 2.5.1 *Let G be a Gröbner basis for the ideal $I \subset k[X_1, \ldots, X_n]$ and let $f \in k[X_1, \ldots, X_n]$. Then $f \in I$ if and only if the remainder on division of f by G is zero.*

Definition 2.5.2 *A minimal Gröbner basis for an ideal I is a Gröbner basis G satisfying for all $f \in G$:*

1. $\mathrm{lc}(f) = 1$
2. $\mathrm{lt}(f) \notin (\mathrm{lt}(G - \{f\}))$

A reduced Gröbner basis for an ideal I is a Gröbner basis G satisfying for all $f \in G$:

1. $\mathrm{lc}(f) = 1$
2. *No (nonzero) term of f is in* $(\mathrm{lt}(G - \{f\}))$

Theorem 2.5.3 *Every nonzero ideal has a unique reduced Gröbner basis (for a given monomial ordering).*

Corollary 2.5.4 *Two ideals are equal if and only if they have the same reduced Gröbner basis .*

Definition 2.5.5 *Let $f, g \in k[X_1, \ldots, X_n]$, with multidegrees a and b, respectively.*

$$S(f,g) = \frac{\mathbf{X^c}}{\mathrm{lt}(f)} f - \frac{\mathbf{X^c}}{\mathrm{lt}(g)} g$$

where $\mathbf{c} = (\max(a_1, b_1), \ldots, \max(a_n, b_n))$*, so* $\mathbf{X^c}$ *is the least common multiple of* $\mathrm{lt}(f)$ *and* $\mathrm{lt}(g)$.

Theorem 2.5.6 *A basis $G = \{g_1, \ldots, g_s\}$ for the nonzero ideal I is a Gröbner basis for I if and only if the division of $S(g_i, g_j)$ by G is zero for all $i \neq j$.*

Buchberger's algorithm

Input: $F = (f_1, \ldots, f_t)$, a basis of the nonzero ideal I.
Output: A Gröbner basis $\{g_1, \ldots, g_s\}$.

```
$G = F$
repeat
        $G' = G$
        For each pair $p, q$ in $G'$ do
            Compute $S(p, q)$ and its remainder $r(p, q)$ on division by $G'$.
            If $r(p, q) \neq 0$ then
                $G := G \bigcup \{r(p, q)\}$
until $G = G'$
```

Chapter 3

The application in Geometric Theorem Proving

With the theory presented in the previous chapter it appears to be possible to prove Geometric Theorems. This concerns theorems about geometry, i.e. points, lines, circles, angles, perpendicularity, etc. We suppose there are two sets of polynomials, one describing the configuration, the other one describing the thesis. This can be done as follows.

We suppose that the *configuration* of the various points, circles, lines, etc., is given by polynomial equations in $X_1, \ldots, X_l$. For example, two points (say (X_1, X_2) and (X_3, X_4)) on the same line (say $y = Ax + B$) are given by the following two equations:

- $X_2 - AX_1 - B = 0$,
- $X_4 - AX_3 - B = 0$.

From now on, we will say the configuration is given by $c_1 = X_2 - AX_1 - B$ and $c_2 = X_4 - AX_3 - B$. In general, the configuration is given by polynomial equations $c_1(\underline{X}) = \ldots = c_n(\underline{X}) = 0$.

In the same manner we can describe one or more *theses* by polynomial equations in $X_1, \ldots, X_l$. For example, we might want to test if a certain point is on the line above, say $(\frac{X_1+X_3}{2}, \frac{X_2+X_4}{2})$. The incidence is given by the following equation:

- $\frac{(X_2+X_4)}{2} - A\frac{(X_1+X_3)}{2} - B = 0$, or $(X_2 + X_4) - A(X_1 + X_3) - 2B = 0$.

We will say the thesis is given by $t = (X_2 + X_4) - A(X_1 + X_3) - 2B$. In general, the thesis holds when $t_1(\underline{X}) = \ldots = t_k(\underline{X}) = 0$ for all $\underline{X}$ that characterize the configuration, so:

$$\text{Thesis holds } \Leftrightarrow \forall(\underline{X} : c_1(\underline{X}) = \ldots = c_n(\underline{X}) = 0 : t_1(\underline{X}) = \ldots = t_k(\underline{X}) = 0) \tag{3.1}$$

Now we use the algebra from Chapter 2. We will work in the ring $K[X_1, \ldots, X_l]$. The 'configuration ideal' $I \subseteq K$ is defined as follows:

$$I = (c_1, \ldots, c_n). \tag{3.2}$$

Suppose t_i is in I, then

$$t_i = f_1 c_1 + \ldots + f_n c_n \quad \text{(for some } f_1, \ldots, f_n). \tag{3.3}$$

So, if t_i is in I and $c_1(\underline{X}) = \ldots = c_n(\underline{X}) = 0$, then $t_i(\underline{X}) = 0$. In practice, this means that the thesis holds if the configuration is such as described by the c_j. Now the only thing we need is a (preferably efficient) way to determine if $t_i \in I$:

Calculate a Gröbner basis G of I. For each t_i determine the remainder on division of t_i by G. If this remainder is zero for every $i = 1, \ldots, k$, then every t_i is in I (by Proposition 2.5.1).

This gives us an algorithmic way of proving the theorem. In the example above the theorem obviously holds, but we can prove this with any mathematical algebra package, for instance Singular. However, the choice of the package certainly is a concern - some packages are faster for this task than others.

```
> /*The ring with 0 as characteristic of the coefficient ring,
>                 variables X(1),...,X(4),A and B,
>                 and degree reverse lexicographical ordering. */
> ring r=0,(X(1..4),A,B),dp;
> poly c1=X(2)-A*X(1)+B;
> poly c2=X(4)-A*X(3)+B;
> ideal I=(c1,c2);
> groebner(I);
_[1]=X(3)*A-X(4)-B
_[2]=X(1)*A-X(2)-B
_[3]=X(2)*X(3)-X(1)*X(4)-X(1)*B+X(3)*B
> poly t=(X(2)+X(4)) - A*(X(1)+X(3)) + 2*B;
> /*Calculate the remainder on division of t
>    by the Groebner Basis of I.*/
> reduce(t,groebner(I));
0
```

Later on we might want to test if t is in the radical $\sqrt{I}$ of I. After all, if t_i is in $\sqrt{I}$ then

$$t_i^k = f_1 c_1 + \ldots + f_n c_n \quad \text{(for some } f_1, \ldots, f_n), \tag{3.4}$$

and if $c_1(\underline{X}) = \ldots = c_n(\underline{X}) = 0$, then $t_i^k(\underline{X}) = 0$, so $t_i(\underline{X}) = 0$. This can be one by testing if $1 \in (zt - 1, I) \subset k[a_1, a_2, b_1, b_2, s, z]$, as is shown in appendix A.

In the next chapter some practical examples of this process will be given.

Chapter 4

Some examples of Geometric Theorems

4.1 A simple problem

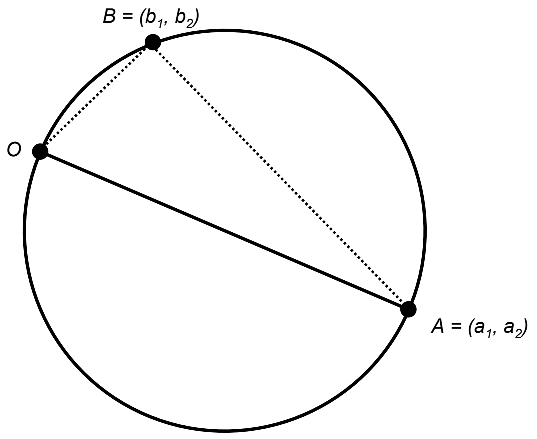

Figure 4.1: The configuration

A simple problem, as seen in Algebra 3, is presented in figure 4.1. The line OA is the diameter of a circle, and the point B is on that circle. We have to prove $OB \perp BA$.

The configuration can be described by the following two equations:

- OA is the diameter of the circle, so $c_1 : a_1^2 + a_2^2 - (2s)^2 = 0$,
- B is on circle, so $(b_1 - \frac{1}{2}a_1)^2 + (b_2 - \frac{1}{2}a_2)^2 - s^2 = 0$ or, equivalently, $c_2 : (2b_1 - a_1)^2 + (2b_2 - a_2)^2 - (2s)^2 = 0$.

The hypothesis $OB \perp BA$ is described by:

$$t : b_1(b_1 - a_1) + b_2(b_2 - a_2) = 0$$

The configuration is characterized by the ideal $I = (c_1, c_2) \subset \mathbb{Q}[a_1, a_2, b_1, b_2, s]$. For now, it suffices to test if $t \in I$. (Later on we might want to test $t \in \sqrt{I}$, see appendix A). Testing if $t \in I$ is done in Singular as follows:

```
>//The ring we use
> ring r=0,(a(1..2),b(1..2),s),lp;
>
>//The configuration
> poly c1=a(1)^2+a(2)^2-(2*s)^2;
> poly c2=(2*b(1)-a(1))^2 + (2*b(2)-a(2))^2 - (2*s)^2;
>
>//The thesis
> poly t=b(1)*(b(1)-a(1)) + b(2)*(b(2)-a(2));
>
>//The verification of the thesis
> ideal i=c1,c2;
> reduce(t,groebner(i));
0
```

So $t \in I$, so by Proposition 2.5.1 our thesis holds!

4.2 The circle theorem of Apollonius

We try the same approach for another exercise from Algebra 3: "In a right triangle OAB with right angle at O, the midpoints of the three sides and the foot of the altitude from O to AB lie on one circle."

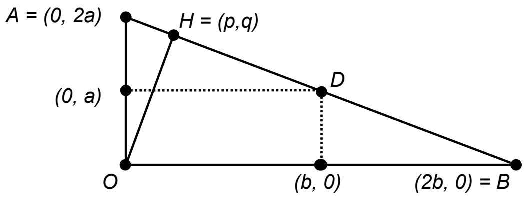

Figure 4.2: The configuration

The configuration is described by the following two equations:

- H on AB: The line AB is described by $y = 2a - x\frac{a}{b}$, since (p, q) is on the line c_1 : $pa + bq - 2ab = 0$;
- $OH \perp AB$: $((2b, 0) - (0, 2a), (p, q)) = 0 \Leftrightarrow c_2 : bp - aq = 0$.

The thesis:

- Obviously, the (unique) circle through $(0, a), (b, 0)$ and D has center $(\frac{a}{2}, \frac{b}{2})$ and squared radius equal to $(\frac{a}{2})^2 + (\frac{b}{2})^2$. The point H must be on this circle, so: $(p - \frac{b}{2})^2 + (q - \frac{a}{2})^2 - (\frac{a}{2})^2 - (\frac{b}{2})^2 = 0$, or $t : (2p - b)^2 + (2q - a)^2 - a^2 - b^2 = 0$.

The Singular session is as follows:

```
>//The ring we use
>ring r=0,(a,b,p,q,y),lp;
>
```

```
>//The configuration
>poly c1=p*a+b*q-2*a*b;
>poly c2=b*p-a*q;
>
>//The thesis
>poly t=(2*p-b)^2 + (2*q-a)^2 - a^2 - b^2;
>
>//The verification of the thesis
>ideal i=c1,c2;
>reduce(t,groebner(i));
-8bp+4p2+4q2
```

That is not what we want. Maybe there are some problems with so-called degeneration.We should, for example, make sure that a and b are not equal to zero. Such things can, up to a certain level, be deduced from the remainder on division of t by $GB(I)$, more information on this subject can be found in [2].

The fact that a and b are not equal to zero can be guaranteed by adding the configuration polynomial $aby - 1$:

```
>//The ring we use
>ring r=0,(a,b,p,q,y),lp;
>
>//The configuration
>poly c1=p*a+b*q-2*a*b;
>poly c2=b*p-a*q;
>poly c3=a*b*y-1;
>
>//The thesis
>poly t=(2*p-b)^2 + (2*q-a)^2 - a^2 - b^2;
>
>//The verification of the thesis
>ideal i=c1,c2,c3;
>reduce(t,groebner(i));
0
```

This shows that $t \in (c_1, c_2, c_3)$, so the theorem described by t holds in the configuration described by c_1, c_2 and c_3.

4.3 Thales' Theorem

Thales' Theorem states (Figure 4.3):

> Given two secant lines r and r', the triangles obtained by intersecting any two parallel lines m and m' with the two secants are similar.

This theorem can be described by the following equations [2, p. 281]:

- $c_1 : qu - pv$;

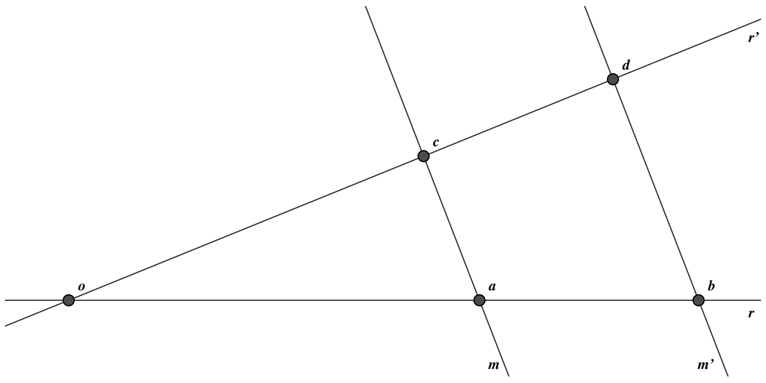

Figure 4.3: Thales' Theorem

- $c_2 : q(u-s) - v(p-l)$;
- $t_1 : (u^2+v^2)l^2 - s^2(p^2+q^2)$;
- $t_2 : ((s-u)^2+v^2)l^2 - s^2((p-l)^2+q^2)$.

In Singular:

```
> ring r=0,(z,u,v,p,q,s,l,a),dp;
> poly c1=q*u-p*v;
> poly c2=q*(u-s)-v*(p-l);
> poly c3=a*q-1; //anti-degeneration;
> poly t1=(u^2+v^2)*l^2 - s^2*(p^2+q^2);
> poly t2=((s-u)^2 + v^2)*l^2 - s^2*((p-l)^2 + q^2);
> ideal i=(c1,c2,c3);
> reduce(t1,groebner(i)); reduce(t2,groebner(i));
0
0
```

A result that shows us that the theorem is true. Note that c_3 has to be included, because $t_1 \notin (c_1, c_2)$.

Chapter 5

Towards the Human Readable proof

It would be nice to be able to produce a human readable proof that $t \in I$, i.e., find q_i such that $t = q_1c_1 + \ldots + q_nc_n$. When this can be obtained, our system is a so called 'oracle' that produces some kind of certificate that the thesis holds. It is not needed to have any knowledge on the process that produces the proof, to be able to verify it.

A method by which this can be done will be explained in the following example.

5.1 Example

Consider $I = (X + Y, X^2 + Y) \subset \mathbb{Q}[X, Y]$. Then $\{Y^2 + Y, X + Y\}$ is a Groebner basis for I:

```
> ring r=0,(x,y),lp;
> poly c1=x+y; poly c2=x^2+y; ideal i=c1,c2;
> groebner(i);
_[1]=y2+y
_[2]=x+y
```

We now test if $t : X^2 + X + 2Y \in I$:

```
> poly t = x^2+x+2y; reduce(t,groebner(i));
0
```

So $t \in I$. Now we use a 'module' instead of an ideal. The module M is generated by the vectors $m_1 = (c_1, 1, 0)^T$ and $m_2 = (c_2, 0, 1)^T$.

$$M = \begin{pmatrix} c_1 & c_2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5.1}$$

Adding the identity matrix gives us the possibility to trace what happens to the c_i when the Groebner basis is generated, because we have

$$\begin{pmatrix} c_1 \\ 1 \\ 0 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \mod M, \text{ so } \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \equiv -c_1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \mod M. \tag{5.2}$$

or, in general:

$$e_i \equiv -c_{i-1}e_1 \ (i \geq 2, \quad \mod M) \tag{5.3}$$

In Singular, e_i is denoted by `gen(i)`.

```
> module m=[c1,1,0],[c2,0,1];
> reduce([t,0,0],std(m));
-gen(3)-gen(2)
> t==-(-c2)-(-c1);
1
```

Having foud the relation $t = q_1c_1 + q_2c_2$, we obtained a straightforward proof of the implication:

$$c_1 = c_2 = 0 \Rightarrow t = 0. \tag{5.4}$$

Example continued

Another method is altering the configuration equations sligthly, in order to be able to trace the generation of the Groebner basis. This is done as follows:

```
> ring r=0,(x,y,s(1..2)),lp;
> poly c1=x+y;
> poly c2=x^2+y;
> poly d1=c1 -s(1); poly d2=c2 -s(2);
> ideal i=d1,d2;
> poly t = x^2+x+2y;
> reduce(t,groebner(i));
s(1)+s(2)
> t==c1+c2;
1
```

(Of course it is not necessary to introduce both the c_i and the d_i). Although this seems like a good way to obtain a proof, for some reason it appears that it is not as robust as the algorithm using the modules.

5.2 A simple problem

In this example the verification is as follows:

```
> ring r=0,(a(1..2),b(1..2),s),lp;
> poly c1=a(1)^2+a(2)^2-(2*s)^2;
> poly c2=(2*b(1)-a(1))^2 + (2*b(2)-a(2))^2 - (2*s)^2;
> poly t=b(1)*(b(1)-a(1)) + b(2)*(b(2)-a(2));
> module m=[c1,1,0],[c2,0,1];
> reduce([t,0,0],std(m));
-1/4*gen(3)+1/4*gen(2)
> t==-1/4*(-c2) + 1/4*(-c1);
1
```

Thanks to the equation $t = \frac{1}{4}(-c_1 + c_2)$ we actually have a proof for the theroem: if $c_1 = c_2 = 0$ then $t = 0$.

In the alternative method:

```
> ring r=0,(a(1..2),b(1..2),s, z(1..2)),lp;
> poly c1=a(1)^2+a(2)^2-(2*s)^2;
> poly c2=(2*b(1)-a(1))^2 + (2*b(2)-a(2))^2 - (2*s)^2;
> poly t=b(1)*(b(1)-a(1)) + b(2)*(b(2)-a(2));
> poly d1=c1 -z(1); poly d2=c2-z(2);
> ideal i=(d1,d2);
> reduce(t,groebner(i));
-1/4*z(1)+1/4*z(2)
> t==-1/4*c1+1/4*c2;
1
```

5.3 The circle theorem of Appolonius

In the second example things should work the same.

```
> ring r=0,(a,b,p,q,y),lp;
> poly c1=p*a+b*q-2*a*b;
> poly c2=b*p-a*q;
> poly c3=a*b*y-1;
> poly t=(2*p-b)^2 + (2*q-a)^2 - a^2 - b^2;
> module n=[c1],[c2],[c3]; reduce([t,0,0,0],std(n));
0
> module m=[c1,1,0,0],[c2,0,1,0],[c3,0,0,1]; reduce([t,0,0,0],std(m));
-8bp*gen(1)+4p2*gen(1)+4q2*gen(1)-4*gen(3)
```

From this we can only conclude (`gen(3)` is replaced by `-c2`):

```
> -8bp*gen(1)+4p2*gen(1)+4q2*gen(1)-4*(-c2)==t*gen(1);
1
```

However, this does not give us what we want, namely $t = g_1c_1 + g_2c_2 + g_3c_3$ for certain $g_{1,2,3}$. Some research shows that a simple change in the ordering does the trick:

```
> ring r=0,(a,b,p,q,y),(c,lp);
> poly c1=p*a+b*q-2*a*b;
> poly c2=b*p-a*q;
> poly c3=a*b*y-1;
> poly t=(2*p-b)^2 + (2*q-a)^2 - a^2 - b^2;
> module m=[c1,1,0,0],[c2,0,1,0],[c3,0,0,1]; reduce([t,0,0,0],std(m));
[0,-4bpy,4bqy-4,-8bp+4p2+4q2]
> -4bpy*(-c1)+(4bqy-4)*(-c2)+(-8bp+4p2+4q2)*(-c3)==t;
1
```

The `(c,lp)` denotes that **first** the ordering should be on the generators ("A small c sorts in descending order, i.e., gen(1) > gen(2) > ...." [6, Section 3.3.3]), and next a regular lexicographical ordering is used. The problems we observed were caused by the default value, which is `(lp,C)` ("a capital C sorts generators in ascending order, i.e., gen(1) < gen(2) < ....").

Appolonius continued

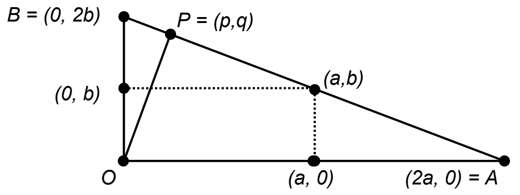

Figure 5.1: The alternative configuration

A more straightforward way of translating this theorem into equations is as follows. Unfortunately, some coordinates changed on the way, now $A = (2a, 0)$, the point $B = (0, 2b)$, still P is on the line AB and the center of the circle is now M. We don't know where M is yet. (See Figure 5.1.)

```
ring r=0,(a,b,m(1..2),p(1..2),s,y),(c,dp);

poly c1=(m(1)-a)^2+m(2)^2-s^2; //(a,0) is on the circle
poly c2=(m(1))^2+(m(2)-b)^2-s^2; //(0,b) is on the circle
poly c3=(m(1)-a)^2+(m(2)-b)^2-s^2; //(a,b) is on the circle
poly c4=-2*a*p(1)+2*b*p(2); //OP perpendicular AB
poly c5=-2*a*p(2)-2*b*p(1)+2*a*2*b; //P on AB
poly c6=a*b*y-1; //a,b not equal to zero
poly t=(m(1)-p(1))^2+(m(2)-p(2))^2-s^2; //P is on the circle
```

The result (using the library defined in Chapter 6) is:

```
Thesis holds:
 t1==
  c1 * (-2*m(1)*p(2)*y+1) +
  c2 * (2*m(2)*p(1)*y+2*a*p(2)*y-3) +
  c3 * (-2*m(2)*p(1)*y-2*a*p(2)*y+2*m(1)*p(2)*y+3) +
  c4 * (-b*m(1)*y-a*m(2)*y+2*m(1)*m(2)*y+1/2*a*p(2)*y) +
  c5 * (-a^2*y+2*a*m(1)*y-1/2*a*p(1)*y) +
  c6 * (4*a^2-8*a*m(1)+2*m(1)*p(1)-p(1)^2+2*m(2)*p(2)-p(2)^2) .
Final tests: OK
```

However, one should be careful when generating the human readable proof of this theorem: If instead of 'degree reverse lexicographical ordering' (denoted by the `dp` in the ring-declaration) for instance the 'lexicographical ordering' is used, it appears to be a very difficult problem. A Pentium 2GHz with 256 MB RAM (running Windows XP) didn't even finish the job (after 30 minutes) because of memory problems.

5.4 Thales' theorem

```
> ring r=0,(z,u,v,p,q,s,l,a),(c,lp);
> poly c1=q*u-p*v;
> poly c2=q*(u-s)-v*(p-l);
> poly c3=a*q-1; //anti-degeneration;
> poly t1=(u^2+v^2)*l^2 - s^2*(p^2+q^2);
> poly t2=((s-u)^2 + v^2)*l^2 - s^2*((p-l)^2 + q^2);
> module m=[c1,1,0,0],[c2,0,1,0],[c3,0,0,1];
> vector v1=reduce([t1,0,0,0],std(m));
> vector v2=reduce([t2,0,0,0],std(m));
> v1;v2;
[0,vp2la2-2vpl2a2+vl3a2+vl+p2sa+qs-sl2a,-ul2a-vp2la2+vpl2a2-vl-p2sa-qs,\\
    u2l2+uvpl2a-uvl3a-vp2sla+vpsl2a-p2s2]
[0,vp2la2-2vpl2a2+vl3a2+vl+p2sa-2psla+qs+sl2a,-ul2a-vp2la2+vpl2a2-vl-p2sa+2psla-qs, \\
    u2l2+uvpl2a-uvl3a-2usl2-vp2sla+vpsl2a-p2s2+2ps2l]
> (transpose(v1)*[0,-c1,-c2,-c3])[1,1]==t1;
1
> (transpose(v2)*[0,-c1,-c2,-c3])[1,1]==t2;
1
```

If we don't use the ordering directive `(c,lp)` the proof does not work, but now it does. Note that, instead of replacing `gen(i)` by $-c_{i-1}$, we use the inner product, where obviously the inner product $(\underline{a}, \underline{b})$ is equal to the only value in the $1{\times}1$ matrix $\underline{a}^T\underline{b}$.

Chapter 6

Implementation

The examples above give rise to the following algorithms.

6.1 True or False?

If the only thing needed is a decision if the theorem given by configuration equations $c_1, \ldots, c_n$ and thesis equations $t_1, \ldots, t_k$ holds, the algorithm is relatively easy:

- Define the ring $K = [X_1, \ldots, X_l]$,
- Define the ideal $I = (c_1, \ldots, c_n)$,
- Calculate Gröbner basis G of I,
- For each t_i, $i = 1, \ldots, k$, calculate the remainder r_i on division of t_i by G,
- The theorem is true if and only if $r_1 = \ldots = t_k = 0$.

6.2 The Human Readable Proof

Suppose the theorem is true, and we want to have a certificate for the proof, i.e. find f_i such that $t = g_1 c_1 + g_n c_n$. The following algorithm, derived from the examples above, will be used in this case:

- Define the ring $K = [X_1, \dots, X_l]$,
- Define the ideal M:

$$M = \begin{pmatrix} c_1 & c_2 & \dots & c_n \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & -1 \end{pmatrix} \tag{6.1}$$

- Calculate S, a basis of M,
- For $i = 1, \dots, k$:
  - Calculate:

$$\text{the remainder} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n+1} \end{pmatrix} \text{ on division of } t_i \cdot \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ by } S,$$

  - Check if $f_1 = 0$. If not, this algorithm fails to generate the human readable proof.
  - Return:

$$g_i = \begin{pmatrix} f_2 \\ \vdots \\ f_{n+1} \end{pmatrix}$$

As was shown in Section 5.1 this algorithm returns g_i such that:

$$t_i = g_{i,1} c_1 + \dots + g_{i,n}. \tag{6.2}$$

A Singular library implementing these algorithms is given in appendix B. Please note that a few tricks are used to minimize the amount of code needed, it should however be relatively easy to see through this.

Chapter 7

OpenMath and Singular

7.1 The OpenMath standard

The OpenMath standard is intended for representing mathematics in such a way that mathematical objects can easily be exchanged between computer programs. From their website [4, Overview]:

> OpenMath is an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. While the original designers were mainly developers of computer algebra systems, it is now attracting interest from other areas of scientific computation and from many publishers of electronic documents with a significant mathematical content.

A rough overview of the standard can be found in Figure 7.1. The 3 layers are explained as follows:

Language The OpenMath language defines the 'grammar'. It defines notions like Variables, Constants, Errors, and Functions.

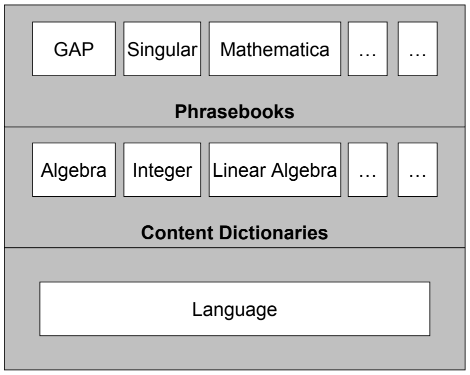

Figure 7.1: The OpenMath framework

Content Dictonaries A Content Dictionary (CD) is (or can be) defined for each area of Mathematics. For example the ‘arith1’ CD describes the notions of ‘minus’, ‘plus’, ‘power’, etc.

Phrasebooks A Phrasebook provides communication between OpenMath and another program. Phrasebooks exist for, for example, Mathematica and GAP. A specific Phrasebook consists of three parts:

- An *encoder* to encode OpenMath objects into commands that the program understands,
- A *decoder* to translate program output into OpenMath objects,
- The physical communication between the program and the Java (or C, or C++) program containing the OpenMath objects.

The interested reader is encouraged to have a look at `http://www.openmath.org` for an extensive overview of the OpenMath standard. The work done in this field is divided in two parts. Firstly, the Phrasebook for Singular, and secondly a Content Dictionary for geometric theorem proving.

7.2 The Singular Phrasebook

The connection between Singular and OpenMath consists of the following subsystems, all of which were implemented in the current project.

singular-link The connection between singular and Java. This program opens an Input- and Outputstream to Singular, and provides the possibility to execute Singular commands.

singular-service Opens a TCP socket with a singular-link, thus enabling the user to execute Singular commands over the internet.

singular-phrasebook Contains the CoDec’s, that translate OpenMath objects into Singular statements, thus implementing the different Content Dictionaries. This is presented to the end-user using the singular-service.

singular-shell Presents the singular-phrasebook to the user in a neat user interface. Enables the user to execute singular commands described in the OpenMath language.

It should be noticed that in this project only the encoding of OpenMath into Singular was realized. The other way is yet to be done. A list of functions from standard Content Dictionaries that were implemented for use in Singular can be found in Appendix C.

7.3 The Content Dictionary

An OpenMath Content Dictionary for polynomial equations with Gröebner Bases was extended, implementing a definition that can be used for geometric theorem proving. That definition is defined as follows:

```
<CDDefinition>
    <Name> extended_in </Name>
    <Description>
        This symbol is a function of at least 3 arguments. The first argument is a list of variables.
        The second and third argument are lists of polynomials in the variables from the first
        argument, C and T respectively.
        When applied to its arguments, it represents the boolean value of the assertion that all elements t
        in T can be written as t = f_1*c_1 + ... + f_n*c_n (c_i in C).
        If the optional 4th argument is 1, those f_i are returned.
    </Description>

    <Example>

        <OMOBJ>
            <OMA>
                <OMS name="extended_in" cd="polygb2"/>

                <OMA>
                    <OMS name="list" cd="list1"/>
                    <OMV name="x"/>
                    <OMV name="y"/>
                </OMA>

                <OMA>
                    <OMS name="list" cd="list1"/>
                    <OMA>
                        <OMS name="plus" cd="arith1"/>
                        <OMV name="x"/>
                        <OMV name="y"/>
                    </OMA>
                    <OMA>
                        <OMS name="plus" cd="arith1"/>
                        <OMV name="x"/>
                        <OMA>
                            <OMS name="times" cd="arith1"/>
                            <OMI> 2 </OMI>
                            <OMV name="y"/>
                        </OMA>
                    </OMA>
                </OMA>

                <OMA>
                    <OMS name="list" cd="list1"/>
                    <OMV name="y"/>
                </OMA>

                <OMI> 1 </OMI>

            </OMA>
        </OMOBJ>

    </Example>
</CDDefinition>
```

The example tests if the theorem defined by the configuration polynomials $c_1 = x + y$ and $c_2 = x + 2y$ and the thesis polynomial $t = y$ holds. Furthermore, a Codec for use with the Singular library as defined in Chapter 6 was written. Using this codec, the result of the OpenMath-object above is:

```
Ring declaration: ring r_base = 0,(x,y),(c,dp)
Thesis holds:
 t1==
  c1 * (-1) +
  c2 * (1) .

Final tests: OK

1
```

Chapter 8

Conclusion

At first, this project mainly focused on Step 4 from Figure 1.1. The biggest problems arose with degenerations, and obtaining the human readable proof (as can be seen in Chapter 5). When those problems were out of the way the implementation of an easy-to-use Singular library was achieved rather easily, apart from the trivial problems that tend to occur when programming in a language in which you don't have much experience. The same holds for the work with OpenMath - once you understand how it works, the implementation is straightforward (which does not reduce the amount of work, however).

Countless possibilities exist for extension of this particular field of interest. For example:

- The translation of polynomials into lines, points, etc,
- Automatically 'fixing' degenerations,
- Manipulate the given polynomials before calculating the Gröbner basis, in such a way that the Gröbner basis-calculation is executed faster,
- Producing the certificate (human readable proof) when the thesis is not in the original ideal, but is in the radical ideal.

The next thing to be done is Steps 1 and 2 from Figure 1.1. This work will be conducted during a three-month internship at the Freie Universität Berlin, the home of Cinderella. The most important translation to be done is the one from Cinderella into OpenMath objects, preferably polynomials. When this is achieved, and the connection between the singular-phrasebook and Cinderella exists, it will be possible to put a theorem forward by clicking and pointing, and obtain a proof automatically.

Bibliography

[1] *The Interactive Geometry Software Cinderella*, `http://www.cinderella.de`.

[2] Arjeh M. Cohen, Hans Cuypers, Hans Sterk (Eds.), *Some Tapas of Computer Algebra*, Springer, 1999.

[3] Dan Roozemond, Stefan van Zwam, *Algebra 3 Eindopdracht, 'Algebraic numbers and the three-colorability of graphs*, 25th November 2002.

[4] *OpenMath, "A new, extensible standard for representing the semantics of mathematical objects"*, `http://www.openmath.org/`.

[5] *Singular - A Computer Algebra System for Polynomial Computations*, `http://www.singular.uni-kl.de/`.

[6] *Singular Manual - HTML User Manual for Singular Version 2-0-3*, University of Kaiserslautern, Department of Mathematics, Centre for Computer Algebra, January 2002

[7] Hans Sterk, *Algebra 3: algorithms in algebra*, 2002-2003.

Appendix A

An additional proof

This proof is taken from [3].

Theorem A.0.1 *Let I be an ideal in $k[X_1,\ldots,X_n]$, and let $f \in k[X_1,\ldots,X_n]$. Then the following equivalence holds:*

$$f \in \sqrt{I} \Leftrightarrow 1 \in (f_1,\ldots,f_k,zf-1) \tag{A.1}$$

where $(f_1,\ldots,f_k,zf-1)$ is an ideal in $k[X_1,\ldots,X_n,z]$.

Proof

- "$\Rightarrow$" Let $f \in \sqrt{I}$, so $f^m \in I$ for some $m \in \mathbb{N}$.

  –

  $$f^m \in I, \text{ so } f^m = \alpha_1 f_1 + \ldots + \alpha_k f_k, \text{ where } \alpha_i \in P \tag{A.2}$$

  From this we conclude

  $$z^m f^m = (z^m\alpha_1)f_1 + \ldots + (z^m\alpha_k)f_k \tag{A.3}$$

  and

  $$z^m f^m \in (f_1,\ldots,f_k) \subseteq (f_1,\ldots,f_k,zf-1) \subseteq \mathbb{C}[X_1,\ldots,X_n,z] \tag{A.4}$$

  –

  $$1 - f^m z^m = (1-fz)(1+fz+(fz)^2+\ldots+(fz)^{m-1}) \tag{A.5}$$

  Because $1+fz+(fz)^2+\ldots+(fz)^{m-1} \in \mathbb{C}[X_1,\ldots,X_n,z]$ we have:

  $$1 - f^m z^m \in (zf-1) \subseteq (f_1,\ldots,f_k,zf-1) \tag{A.6}$$

  From these two sections we conclude:

  $$1 = \underbrace{1 - f^m z^m}_{\in(f_1,\ldots,f_k,zf-1)} + \underbrace{f^m z^m}_{\in(f_1,\ldots,f_k,zf-1)} \in (f_1,\ldots,f_k,zf-1). \tag{A.7}$$

- "$\Leftarrow$" If $1 \in (f_1, \ldots, f_k, zf - 1) \subseteq \mathbb{C}[X_1, \ldots, X_n, z]$.

  Then:

$$1 = \alpha_1 f_1 + \ldots + \alpha_k f_k + \alpha(zf - 1) \tag{A.8}$$

  where $\alpha_i, \alpha \in \mathbb{C}[X_1, \ldots, X_n, z]$.

  We proceed to $\mathbb{C}[X_1, \ldots, X_n, z]$ and substitute: $z := \frac{1}{f}$. We obtain:

$$1 = \alpha'_1 f_1 + \ldots + \alpha'_k f_k \tag{A.9}$$

  Both sides are multiplied by f^t, where t is the maximum power of z that occurs in α_i, α. We obtain

$$f^t = \beta_1 f_1 + \ldots + \beta_k f_k \tag{A.10}$$

  where $\beta_i \in \mathbb{C}[X_1, \ldots X_n]$, so $f^t \in I$ and $f \in \sqrt{I}$.

$\square$

Appendix B

Singular Library

```
////////////////////////////////////////////////////////////////////////////////
version="$Id: AGTP.lib,v 0.1 2003/07/05$";
category="Miscellaneous";
info="
LIBRARY:  AGTP.lib  Automatic Geometric Theorem Proving
AUTHOR:   D.A. Roozemond (d.a.roozemond@student.tue.nl)

PROCEDURES:
 AGTP_Test (string vars, string c_poly, string t_poly [,1]); Geometric configuration
    described by c_poly, thesis described by t_poly.
";

////////////////////////////////////////////////////////////////////////////////

LIB "matrix.lib";

proc AGTP_Test (string vars, string c_string, string t_string, list #)
"USAGE:   AGTP_Test (string vars, string c_poly, string t_poly [,1])
RETURN:
@format
boolean b: Thesis t_poly in configuration c_poly holds.
    t_poly and c_poly are polynomials in vars.
@end format
NOTE:   If the argument vars is empty, it is tried, *in a very
            primitive way* to deduce the variables from c_string and t_string.
        If a fourth argument 1 is given, the procedure is verbose and tries
            to find an actual proof if the thesis holds.
SEE_ALSO:
KEYWORDS:
EXAMPLE:  example AGTP_Test; shows a few examples.
"
{
    int retval = 0;
    int be_verbose = (size(#) > 0); if (be_verbose) { be_verbose = (#[1] > 0); }

    /*
        This makes sure a proper ring is created, from the variables
        found in c_string and t_string if needed.
    */
    if (vars != "") {
        if (vars[1]=="[") {
            vars = "(" + vars[2,size(vars)-2] + ")";
        } else {
            vars = "(" + vars + ")";
        }
        string ringdecl = "ring r_base = 0," + vars + ",(c,dp)";
    } else {
        string ringdecl = AGTP_CreateBasering ("r_base", c_string, t_string, "c,dp", be_verbose);

    }
    if (be_verbose)
    {
        print("Ring declaration: " + ringdecl);
    }

    execute (ringdecl);
    if (c_string[1] != "[") { c_string = "[" + c_string + "]"; }
    execute ("vector c_poly = " + c_string + ";");
    if (t_string[1] != "[") { t_string = "[" + t_string + "]"; }
    execute ("vector t_poly = " + t_string + ";");

    ideal c_ideal = c_poly;
    ideal c_gb = groebner(c_ideal);
    vector rem_poly = reduce(t_poly,c_gb);

    if (rem_poly != 0) {
        if (be_verbose == 1) {
            //Only execute this part when we should be verbose
            print("Thesis does not hold. Remainders are as follows: ");
            for (int i=1; i <= nrows(t_poly); i = i + 1) {
                print(" t" + string(i) + ": " + string(rem_poly[i]));
            }
            print("");
        }
        retval = 0;
    } else {
        if (be_verbose == 1) {
            //Only execute this part when we should be verbose, and the user
            //wants to know more than just if the thesis holds.
            print("Thesis holds:");
            AGTP_RealProof(c_poly, t_poly);
            print("");
        }
        retval = 1;
    }

    return(retval);
}
example
{ "EXAMPLE:";
    echo = 2;

    //Simple example (holds)
    AGTP_Test ("","x+y,x^2+y","x^2+x+2*y,x^2+2*x+3*y",1);

    //Simple example (does not hold)
    AGTP_Test ("","x+y,x^2+y","x^2+x+2*y,x^2+3*x+3*y",1);

    //B3
    AGTP_Test ("","a1^2+a2^2-(2*s)^2,(2*b1-a1)^2 + (2*b2-a2)^2 - (2*s)^2",
        "b1*(b1-a1) + b2*(b2-a2)",1);

    //G
    AGTP_Test ("","p*a+b*q-2*a*b,b*p-a*q,a*b*y-1","(2*p-b)^2 + (2*q-a)^2 - a^2 - b^2",1);

    //Thales's Theorem
    AGTP_Test ("","q*u-p*v,q*(u-s)-v*(p-l),a*q-1",
        "(u^2+v^2)*l^2 - s^2*(p^2+q^2),((s-u)^2 + v^2)*l^2 - s^2*((p-l)^2 + q^2)",1);

    //Triangle's (OAB) Bissectrices through one point
    AGTP_Test ("","a1*b2-2*a1*q-a2*b1+2*a2*p+b1*q-b2*p,-a1*b2+a1*q+a2*b1-a2*p-2*b1*q+2*b2*p",
        "-a1*q+a2*p-b1*q+b2*p",1);

}

///////////////////////////////////////////////////////////////////////////////

proc AGTP_RealProof (vector c_poly, vector t_poly)
"USAGE:   AGTP_RealProof (vector c_poly, vector t_poly);
ASSUME:   Basering, where the c_i and t_i are in, is defined.
RETURN:
@format
string representing the proof.
@end format
SEE_ALSO:
KEYWORDS:
EXAMPLE:  example AGTP; shows a few exampleoots.
"
{
    matrix mod_matrix = transpose(concat(c_poly,-unitmat(nrows(c_poly))));
    module mod_module = mod_matrix;
    vector coeff_poly = 0; int i = 0; int j = 0; string delim = "";
    int finaltest = 1; poly tmp_poly = 0;
    string opt = "";

    for (i=1; i <= nrows(t_poly); i++) { //For each polynomial in t_poly:
        coeff_poly = reduce(t_poly[i]*gen(1),std(mod_module));

        if (coeff_poly[1] == 0) {
            opt = opt + " t" + string(i) + "== " + newline;
            tmp_poly = 0;
            for (j=1; j <= nrows(c_poly); j++) { //For each polynomial in c_poly:
                if (j == nrows(c_poly)) { delim = "."; } else { delim = "+"; }
                opt = opt + "  c" + string(j) + " * (" +
                    string(coeff_poly[j+1]) + ") " + delim + newline;
                tmp_poly = tmp_poly + c_poly[j]*coeff_poly[j+1];
            }

            //A final check if what we suspect really is correct
            if (t_poly[i] != tmp_poly) {
                ERROR("The above is incorrect!!");
                finaltest = 0;
            }
        } else {
            ERROR("Error finding coefficients.");
        }
    }

    if (finaltest) {
        opt = opt + newline + "Final tests: OK";
    }

    return(opt);
}

////////////////////////////////////////////////////////////////////////////////

proc AGTP_CreateBasering (string ringname, string c, string t, string ordering)
{
    /*
    Returns a string, representing the command to make ringname
     a basering from the used variables in c and t.
    Note: Using xy means xy is a special character. Use x*y instead.
    */
    string vars = ","; string ct = c + "," + t;
    string tstr = "";

    for(int i=1; i <= size(ct); i++)
    {
        if (!IsDelim(ct[i]))
        {
            tstr = tstr + ct[i];
        }
        else
        {
            if (!IsNumber(tstr) && (tstr != ""))
            {
                if (!find(vars, "," + tstr + ","))
                {
                    vars = vars + tstr + ",";
                }
            }
            tstr = "";
        }
    }

    vars = vars[2,size(vars)-2];
    string cmd="ring " + ringname + " = 0,(" + vars + "),(" + ordering + ");";

    return(cmd);
}


////////////////////////////////////////////////////////////////////////////////
/////////////////  NEEDED BUT NOT INTERESTING FUNCTIONS.... /////////////////
////////////////////////////////////////////////////////////////////////////////

proc IsDelim(string c) {
    /*
    Input: character c.
    Returns: TRUE if
        - c is a 'delimiter' i.e. a +, *, -, ^, (, ),
        - or c is a ,
        - or c is a space character.
    */
    return ((c == "+") || (c == "*") || (c == "-") || (c == "^") || (c == ",") ||
        (c == " ") || (c == ")") || (c == "(") || (c == "]") || (c == "["));
}

proc IsNumber(string c) {
    /*
    Input: string c.
    Returns: TRUE if c is a number, i.e. all characters in c are between "0" and "9".
    */
    int i = 1; int r = 1;
    while (r && (i <= size(c))) {
        r = r && ("0" <= c[i]) && (c[i] <= "9");
        i++;
    }
    return(r);
}
```

Appendix C

Singular Phrasebook

The following standard objects where implemented for use in Singular:

- `alg1.one`
- `alg1.zero`
- `arith1.abs`
- `arith1.divide`
- `arith1.gcd`
- `arith1.lcm`
- `arith1.minus`
- `arith1.plus`
- `arith1.power`
- `arith1.times`
- `arith1.unary_minus`
- `integer1.factorof`
- `integer1.quotient`
- `integer1.remainder`
- `interval1.integer_interval`
- `linalg1.determinant`
- `linalg1.matrix_selector`
- `linalg1.outerproduct`
- `linalg1.scalarproduct`
- `linalg1.transpose`

- `linalg1.vector_selector`
- `linalg1.vectorproduct`
- `linalg2.matrixrow`
- `linalg2.vector`
- `list1.list`
- `logic1.and`
- `logic1.equivalent`
- `logic1.false`
- `logic1.implies`
- `logic1.not`
- `logic1.or`
- `logic1.true`
- `relation1.approx`
- `relation1.eq`
- `relation1.geq`
- `relation1.gt`
- `relation1.leq`
- `relation1.lt`
- `relation1.neq`