PREPUBLICACIONES DEL DEPARTAMENTO DE ÁLGEBRA
DE LA UNIVERSIDAD DE SEVILLA

Estableciendo un puente entre la Geometría Dinámica y el Álgebra Computacional

(Texto de la conferencia impartida en el Seminario del Departamento de Álgebra el día 22-Noviembre-2001)

Eugenio Roanes-Lozano

Prepublicación nº 14 (5-Diciembre-2001)

Departamento de Álgebra. Universidad de Sevilla

ESTABLECIENDO UN PUENTE ENTRE LA GEOMETRÍA DINÁMICA Y El ÁLGEBRA COMPUTACIONAL

Eugenio Roanes-Lozano

Dept. Algebra, Univ. Complutense Madrid

Parcialmente subvencionado por el proyecto TIC2000-1368-C03-03
(Ministerio de Ciencia y Technología)

1 Introducción

1.1 Situación actual

• Los Sistemas de Cómputo Algebraico (CAS) aparecen en los '60 para poder afrontar problemas de Astronomía y Física de Altas Energías.

• Usan aritmética exacta y pueden manejar variables sin asignación (esto es, variables en el sentido *matemático*, no en el sentido usual en Computación).

• Los más conocidos son *Maple*, *Derive*, *Mathematica*, *Axiom*, *Macsyma*, *Reduce* y *MuPad*.

• Los CAS suelen incluyen muchas extensiones, como diferenciación simbólica e integración, resolución de sistemas lineales y no lineales, representaciones gráficas en 2D y 3D...

• Hay algunos de propósito específico, como *CoCoA* (especializado en calcular bases de Gröbner en anillos de polinomios sobre cuerpos finitos).

• Los Sistemas de Geometría Dinámica (DGS) se desarrollan desde el principio de los '90 con un propósito didáctico: permitir que el usuario "explore" la Geometría de la regla y el compás.

• El adjetivo *dinámica* proviene del hecho de que, una vez que la construcción está acabada, los primeros objetos dibujados (puntos) pueden ser arrastrados con el ratón, cambiando consecuentemente toda la construcción. Usualmente incorporan animación y trazado.

• Existen en la actualidad muchos DGS, como *The Geometer's Sketchpad*, *Cabri Geometry II*, *Cinderella*, *Euklid*, *Dr. Geo*, *WinGeom*, *Autograph*, *The Geometric Supposer*...

• *Cinderella* es el único que ofrece la posibilidad de manejar Geometrías no-Euclídeas.

• Además, es el único que realiza internamente los cálculos en $\mathbb{C}$ (lo que evita que el dibujo desaparezca en casos excepcionales).

• Desafortunadamente, los CAS y los DGS han evolucionado independientemente. Algunos CAS, como *Maple* incluyen potentes paquetes dedicados a la Geometría Euclídea, pero ningún CAS incluye capacidades de geometría Dinámica.

• Por otras parte, los DGS no pueden manejar (al menos desde el punto de vista del usuario) variables sin asignación. Por tanto, lo que puede salvarse desde un DGS es *sólo*

- un gráfico *vivo* (para ser interpretado por ese DGS),
- un *algoritmo geométrico*: (*script* or *macro*, que puede ser interpretado por ese DGS)
- un gráfico fijo (en uno de los formatos gráficos standard).

• La necesidad de esta cooperación ya se menciona en un libro de Tomás Recio (1998).

• Esta falta de cooperación es más sorprendente aún en casos como el de la calculadora *TI-92* de Texas Instruments, donde ambas tecnologías están disponibles simultáneamente.

1.2 Cooperación: las soluciones de otros

• Parece ser que cierto producto: *The Algebraic Geometer*, ofrecerá esta posibilidad. Pero aunque este producto ha sido anunciado hace ya tiempo, parece que no está a la venta.

• Recientemente, los profesores J.L. Valcarce y F. Botana (Univ. de Vigo) han implementado en visual-Prolog tres impresionantes DGS denominados *Lugares*, *Discovery y Rex*.

• Por el momento en versión beta, pueden comunicarse con CoCoA o Mathematica para determinar lugares geométricos o para producir demostraciones automáticas de teoremas geométricos y puede incluso completar teoremas geométricos (puede a veces encontrar hipótesis que faltan para que un teorema sea cierto), e.e., realizar *descubrimiento automático*.

• No obstante, el usuario no tiene control de lo que hace el CAS.

1.3 Nuestra solución

- Nuestra estrategia es completamente diferente: reutilizaremos software.

- Hemos elegido desarrollar un puente entre un DGS y un CAS ya existentes.

- En este primer estadio hemos elegido *The Geometer's Sketchpad 3* y *Maple 7*. Elegimos *The Geometer's Sketchpad 3* por ser muy amigable (especialmente sus *scripts*). Elegimos *Maple 7* pues es también muy amigable y su paquete sobre bases de Gröbner es potente y bien documentado.

- La idea clave es bien simple:

  i) usar la descripción (casi en lenguaje natural) de los *scripts* de *The Geometer's Sketchpad 3*

  ii) traducir los *scripts* a sintaxis de *Maple* usando un traductor escrito *ad hoc* en *Maple* (paquete `GeoTran`)

  iii) interpretar el código traducido con un nuevo paquete de Geometría para *Maple* que pueda manejar parámetros `ParamGeo`.

- El futuro desarrollo de esta línea de trabajo será implementar paquetes para distintos DGS y CAS. Ello es directo p.ej. entre *Cinderella* (cuyos *Construction Texts* son muy similares a los *scripts* de *Sketchpad*) y *Macsyma* o *MuPad*.

2 Lo que no proporcionan los CAS y DGS standard

2.1 Las carencias de los DGS standard

• Lo que haría falta del DGS es que proporcionara la posibilidad de manejar y exportar *datos paramétricos* del dibujo:

- coordenadas de puntos (permitiendo parámetros como coordenadas),
- ecuaciones de objetos (permitiendo parámetros como coeficientes),
- longitud de objetos (dependientes de parámetros),
- ...

0.0.1 Ejemplo.- *Sea ABC un triángulo rectángulo general (p.ej. $\widehat{A} = 90^0$). Podemos elegir, sin pérdida de generalidad : $A = (0,0)$, $B = (1,0)$, $C = (0,c)$. Como el punto C yace sobre el eje y, en un cierto instante, t_0, sus coordenadas pueden ser, p.ej., $(0,3)$, pero sus coordenadas paramétricas (generales) son $(0,c)$; $c \in \mathbb{R}$ (y $c \neq 0$ para que el triángulo no sea degenerado). O sea, el punto C es libre y una de sus coordenadas depende de un parámetro.*

En consecuencia, la ecuación general (dependiente de un parámetro) de la recta BC es

$$y + c \cdot x - c = 0$$

a pesar de que en un cierto instante t_0 sea, p.ej.,

$$y + 3x - 3 = 0 \ .$$

• Como se dijo más arrriba, ninguno de los DGS standard puede retornar coordenadas como $(c,0)$ o una ecuación como $y + c \cdot x - c = 0$.

2.2 Las carencias del paquete "Geometry" de Maple

• *Maple* incorpora un excelente paquete para Geometría Euclidea (denominado *Geometry*). Desafortunadamente (y sorprendentemente), este paquete no puede manejar parámetros (e.e., por ejemplo las coordenadas de los puntos que definen una recta deben ser todas numéricas).

0.0.2 Ejemplo.- *Veamos que ocurre:*

```
> with (geometry):
> point(A,[0,0]):
> point(B,[2,3]):
> line(r,[A,B]):
> Equation(r);
> enter name of the horizontal axis> x;
> enter name of the vertical axis> y;
                                -3 x + 2 y = 0
```

funciona perfectamente, pero el similar

```
> point(A,[a1,a2]):
> point(B,[b1,b2]):
> line(r,[A,B]):
    geometry/checkline: One of the following conditions
      must be satisfied    -b1 <> 0   -b2 <> 0
    Error, (in geometry/checkline) not enough information:
      the line is not defined
```

obviamente falla.

3 El paquete `ParamGeo` para Maple

• `ParamGeo` es el paquete de *Maple* de "Geometría Paramétrica" que hemos desarrollado. Es similar al paquete *Geometry* pero con las siguientes ventajas

- está disenado para traducir directamente los pasos de una construcción realizada con *The Geometer's Sketchpad*
- los puntos iniciales pueden tener coordenadas paramétricas y estos parámetros se arrastran durante los cálculos subsiguientes.

• La necesidad de implementar un paquete de este tipo se trató en una comunicación al 4th. Int. Derive Conf. (Liverpool, 2000). Una primera versión de este paquete fue presentada en el ACA'2001 y se menciona en el estudio de cooperación entre DGS y CAS presentado en el ICTMT-5 (2001).

4 Procedimientos en el paquete `ParamGeo`

- Los procedimientos implementados pueden clasificarse como sigue:
  - procedimientos declarativos: point, line, segment, circumference by centre and point, circumference by centre and radius
  - procedimientos constructivos: midpoint, parallel line, perpendicular line, intersection of two algebraic sets of points
  - procedimiento booleano: isIn (pregunta sobre una pertenencia)
  - procedimiento auxiliar: square of distance
  - procedimientos derivados: perpendicular bisector, angle bisector.
  - procedimiento declarativo de pertenencia: pointOnObject.

Algunos son triviales, como:

- `point` permite declarar puntos
  sintaxis: `point(x-coordinate, y-coordinate)`
  efecto: devuelve la lista de las coordenadas dadas.

otros no son tan directos:

- `segment` permite declarar segmentos
  sintaxis: `segment(name of endpoint1, name of endpoint2)`
  efecto: devuelve la lista: [ecuación de la recta por los puntos dados, coordenadas del extremo 1, coordenadas del extremo 2]

y en otros hay que considerar varios casos

- `intersection` calcula y permite declarar la intersección de exactamente dos objetos algebraicos (segmento se substituye por recta)
  sintaxis: `intersection(name of set of points 1, name of set of points 2)`
  efecto: se distinguen 4 casos:
  - si los dos objetos dados son disjuntos, escribe un mensaje de error (no solution) y no devuelve nada
  - si los dos objetos tienen un punto en común, devuelve las coordenadas del punto de intersección (como una lista de coordenadas)
  - si los dos objetos tienen dos punto en común, da un mensaje (two solutions) y devuelve una lista con los dos puntos de intersección (cada uno dado como una lista de coordenadas punto de intersección (como una lista)
  - si los dos objetos tienen infinitos puntos de intersección, da un mensaje (infinite No. of solutions) y no devuelve nada.

0.0.3 Observación.- *El caso de intersecar conjuntos semi-algebraicos tiene muchos más casos.*

• Nota: No hay discusión sobre los parámetros: el programa funciona correctamente si el sistema es fijo (en $\mathbb{C}$): Por ejemplo

$$(x-7)^2+(y-3)^2-16=0 \quad , \quad x=h$$

da que tiene dos soluciones dependientes de h.

Sin embargo:

$$x=h \quad , \quad x=7$$

da NO SOLUTION en lugar de una discusión sobre h, pues la propia compatibilidad del sistema es la que depende de h.

• El paquete `ParamGeo` puede ser usado directamente en Maple, siendo su interés principal comenzar con un *script* de *The Geometer's Sketchpad's*. Veamos un ejemplo sencillo de uso directo.

0.0.4 Ejemplo.- *Las diagonales de un paralelogramo se cortan en su punto medio.*

```
> read "ParamGeo.mpl";
> A:=point(a1,a2):
> B:=point(b1,b2):
> C:=point(c1,c2):
> r:=line(A,B):
> s:=line(A,C):
> t:=parallel(s,B):
> u:=parallel(r,C):
> D:=intersection(t,u):
                              UNIQUE SOLUTION
> diag1:=line(A,D):
> diag2:=line(B,C):
> intersection(diag1,diag2);
                              UNIQUE SOLUTION
                    [1/2 c1 + 1/2 b1, 1/2 c2 + 1/2 b2]
```

5 Traducción del output del DGS al CAS

5.1 Los *scripts* en *The Geometer's Sketchpad 3*

• *The Geometer's Sketchpad 3* puede salvar figuras (denotadas *sketches*) y *algoritmos geométricos* (denotados *scripts*). Los *Scripts* pueden salvarse en el formato interno de *The Geometer's Sketchpad* para *scripts* `(.GSS)` o en formato legible `(.TXT)`.

0.0.5 Ejemplo.- *Consideremos el script generado por la construcción de la mediatriz de un segmento (que, una vez definido, permite construir la mediatriz de cualquier segmento, dados sus extremos). Si se guarda en modo texto, tiene la siguiente apariencia:*

```
Mediatri.gss

Given:
Point A
Point B
--------------
Steps:
1. Let [j] = Segment between Point A and Point B.
2. Let [C] = Midpoint of Segment [j].
3. Let [k] = Perpendicular to Segment [j] through Midpoint [C].
```

lo que está muy próximo al lenguaje matemático standard. Ahora debería transformarse a sintaxis Maple.

5.2 `GeoTran`: Traductor de *scripts* a sintaxis Maple (por Carl Devore, Univ. Newark)

• `GeoTran` está escrito en *Maple* (previamente realizamos una versión más primitiva implementada en lenguaje Logo).

• La idea es que el usuario sólo tiene que introducir los parámetros de los llamados *Given*. *Maple* realizará los cálculos subsiguientes (que el DGS presentó en formato geométrico) en forma algebraica.

0.0.6 Ejemplo.- *Si el input es el listado anterior, el output será:*

```
#Given:
A:=point(A_x,A_y);
B:=point(B_x,B_y);
#--------------
#Steps:
j:=segment(A,B);
C:=midpoint(j);
k:=perpendicular(j,C);
```

0.0.7 Sumario.- *Para obtener esta tradución de un script de The Geometers' Sketchpad 3's dado en formato* `.TXT`*, el usuario sólo tiene que:*

1) comenzar una sesión de Maple

2) cargar el paquete de Geometría Paramétrica `ParamGeo` *(que automáticamente carga el traductor de Sketchpad a Maple* `GeoTran`*):*

```
> read "paramGeo.mpl";
```

3) (opcional) sería recomendable poner `echo` *a* `2` *para que las líneas del archivo traducido pudieran ser vistas en pantalla mientras se leen:*

```
> interface(echo=2);
```

4a) ejecutar el traductor aplicado al `.TXT` *del script que describe la construcción, creándose el correspondiente* `.MPL`*:*

```
> `Sketchpad->Maple`(`filename.txt`);
```

4b) (opcional) si la traducción no es completamente satisfactoria, el usuario puede editar el `.MPL` *traducción del* `.TXT`*. Este puede ser el caso, p.ej., cuando haya dos puntos de intersección: el usuario tiene que precisar manualmente a cual de los dos se refiere.*

5) (opcional) dar una particularización de las coordenadas de los puntos dados, p.ej.:

```
> A_x:=1;
> A_y:=0;
```

6) leer el `.MPL` *traducción del* `.TXT` *(se ejecuta automáticamente línea a línea cuando se lee):*

```
> read(‘filename.mpl‘);
```

Entonces todas las coordenadas, ecuaciones,... de los objetos en el sketch estarán disponibles para el usuario (en Maple).

6 Ejemplos

- Se pueden encontrar ejemplos detallados en la worksheet de Maple adjunta como Apéndice II.

Agradecimientos

• Muchos colegas han ayudado directa o indirectamente en este trabajo. En orden alfabético:

- Francisco Botana y Jose Luis Valcarce (Universidad de Vigo, autores de Lugares, Discovery y Rex)
- Carl Devore (Newark University)
- Nicholas Jackiw, Steven Rasmussen y Scott Steketee (Key Curriculum Press)
- John Olive (Georgia University)
- Eugenio Roanes Macias (Univ. Complutense)
- Tomás Recio (Universidad de Cantabria).

Conclusiones

• Entendemos que esta colaboración entre DGS y CAS es muy interesante. El Álgebra es el lenguaje más potente para la Geometría, y esta colaboración permite aumentar sus posibilidades.

• Una aplicación muy importante es la posibilidad de tratar con el ordenador todo el proceso matemático de descubrimiento (o re-descubrimiento):

$$\text{Investigación} \longrightarrow \text{Conjetura} \longrightarrow \text{Comprobación} \longrightarrow \text{Demostración} \quad .$$

References

[1] `www.maplesoft.com`

[2] `www.keypress.com/catalog/products/software/Prod_GSP.html`

[3] U. Kortenkamp: *Foundations of Dynamic Geometry* (PhD. Thesis), Swiss Fed. Inst. Tech. Zurich, 1999.

[4] B. Kutzler: *Introduction to DERIVE for Windows* (Chapter 12). Ed . B. Kutzler, 1996.

[5] T. Recio: *Cálculo Simbólico y Geométrico*. Ed . Síntesis, 1998.

[6] `www.saltire.com/paraprts.html`

[7] J.L. Valcarce, F. Botana: *Lugares. Manual de Referencia.* Universidad de Vigo, 2001.

[8] `rosalia.uvigo.es/sdge/`

[9] D. Cox, J. Little, D. O'Shea: *Ideals, varieties, and algorithms*. Springer-Verlag, 1992.

[10] B. Buchberger: *Applications of Gröbner Bases in non-linear Computational Geometry*. In: J.R. Rice (editor): *Mathematical Aspects of Scientific Software*. Springer-Verlag, 1988 (59-87).

[11] S.C. Chou: *Mechanical Geometry Theorem Proving*. Reidel, 1988.

[12] M. Kreuzer, L. Robbiano: *Computational Commutative Algebra*. Springer-Verlag, 2000.

[13] E. Roanes-M., E. Roanes-L: *Nuevas Tecnologías en Geometría*. Editorial Complutense, Madrid (1994).

[14] E. Roanes-L., E. Roanes-M.: Automatic Theorem Proving in Elementary Geometry with DERIVE 3. *The International DERIVE Journal*, **3(2)** (1996), 67-82.

[15] E. Roanes-L., E. Roanes-M.: *Mechanical Theorem Proving in Geometry with DERIVE-3*. In: B. Bärzel (editor): *Teaching Mathematics with Derive and the TI-92*. ZKL-Texte Nr.2 , 1996 (404-419).

[16] Wu Wen-Tsun: On the decision problem and the mechanization of theorem-proving in elementary Geometry. *A.M.S. Contemporary Mathematics*, **29** (1984), 213-234.

[17] Wu Wen-Tsun: Some recent advances in Mechanical Theorem-Proving of Geometries. *A.M.S. Contemporary Mathematics*, **29** (1984), 235-242.

[18] T. Recio, M.P. Vélez: Automatic Discovery of Theorems in Elementary Geometry. *Journal of Automated Reasoning*, **23** (1999), 63-82.

[19] E. Roanes-M., E. Roanes-L.: *Automatic Determination of geometric loci. 3D-extension of Simson-Steiner Theorem*. In: J. Campbell, E. Roanes-L. (editors): *Proceedings of AISC'2000*. Springer-Verlag, LNCS 1930, 2001 (157-173).

[20] –: *The Geometer's Sketchpad User Guide and Reference Manual v.3*. Key Curriculum Press, 1995.

[21] B.W. Char et al.: *Maple V Language Reference Manual*. Springer-Verlag, 1991.

[22] B.W. Char et al.: *Maple V Language Library Reference Manual*. Springer-Verlag, 1991.

[23] A. Heck: *Introduction to Maple*. Springer-Verlag, 1996.

[24] R.M. Corless: *Essential Maple*. Springer-Verlag, 1995.

[25] E. Roanes-M., E. Roanes-L.: *Cálculos Matemáticos con Maple V.5*. Ed. Rubiños, 1999.

[26] E. Roanes-L., E. Roanes-M.: *How Dinamic Geometry could Complement Computer Algebra Systems (Linking Investigations in Geometry to Automated Theorem Proving)*. Procs. of the Fourth Int. Derive/TI-89/TI-92 Conference, Liverpool, July 2000. Published in CD-ROM by BK-Teachware, 2000.

[27] E. Roanes-L.: *Boosting the Geometrical Possibilities of Dynamic Geometry Systems and Computer Algebra Systems through Cooperation*. Procs. of ICTMT-5, Klagenfurt (Austria), August 2001. To appear.

Apéndice I: Código del paquete `paramGeo`

```
# ParamGeo: "PARAMETRIC GEOMETRY" PACKAGE FOR MAPLE:
#           SKETCHING, AUTOMATIC TH. PROVING AND
#           DISCOVERY AND COOPERATION WITH THE
#           GEOMETER'S SKETCHPAD MADE EASY
#
# Authors: Eugenio Roanes-Lozano, Univ. Complutense de Madrid
#
# October 2001
#
# Addresses: e-mail: eroanes@mat.ucm.es  ;  eroanes@fi.upm.es
#            snail-mail: Dept. Algebra, Desp. 305
#                        Fac. Educacion
#                        c/ Rector Royo Villanova s/n
#                        28040 - Madrid (Spain)

 unprotect(D);  D:='D':
 unprotect(O);  O:='O':
 unprotect(point);
```

```
LREL:=[]:                  #list of relations defined by poinOnObject

point:=proc(p1,p2)
  [p1,p2];
 end:

sqdist:=proc(P::list,Q::list)
  ((Q[1]-P[1])^2+(Q[2]-P[2])^2)
 end:

line:=proc(P::list,Q::list)
  if P[1]=Q[1] then x-P[1]=0
            elif P[2]=Q[2] then y-P[2]=0
                 else collect(collect(simplify(
                      (Q[1]-P[1])*(y-P[2])-(Q[2]-P[2])*(x-P[1])=0),x),y)
   fi;
 end:

segment:=proc(P::list,Q::list)
  [line(P,Q), P, Q];
 end:
```

```
circumCP:=proc(P::list,Q::list)
  (x-P[1])^2+(y-P[2])^2 - sqdist(P,Q)=0;
 end:

circumCR:= proc(P::list, r::list)          #r is a segment, not a distance
   (x-P[1])^2+(y-P[2])^2 - sqdist(r[2],r[3])=0;
 end:

midpoint:=proc(s::{equation,list})
  (s[2]+s[3])/2;
 end:

parallel:=proc(o::{equation,list},P::{equation,list})
  local ec;
  if nops(o)=3 then ec:=op(1,o)  #equation distinguishing
               else ec:=o        #if it is segment or line
   fi;
  if coeff(lhs(ec),x)=0 then y=P[2]
     elif coeff(lhs(ec),y)=0 then x=P[1]
          else coeff(lhs(ec),x)*(x-P[1])+coeff(lhs(ec),y)*(y-P[2])=0
   fi;
 end:
```

```
perpendicular:=proc(o::{equation,list},P::{equation,list})
  local ec;
  if nops(o)=3 then ec:=op(1,o)  #equation distinguishing
               else ec:=o        #if it is segment or line
   fi;
  if coeff(lhs(ec),x)=0 then x-P[1]=0
     elif coeff(lhs(ec),y)=0 then y-P[2]=0
          else coeff(lhs(ec),y)*(x-P[1])-coeff(lhs(ec),x)*(y-P[2])=0
   fi;
 end:

intersection:=proc(r1::{equation,list},r2::{equation,list})
  local r1r,r2r,sola,sol;
  if nops(r1)=3 then r1r:=op(1,r1)  #equation distinguishing
                else r1r:=r1        #if it is segment or line
   fi;
  if nops(r2)=3 then r2r:=op(1,r2)  #equation distinguishing
                else r2r:=r2        #if it is segment or line
   fi;
  sol:=solve({r1r,r2r},{x,y});
    if [sol]=[] then print(‘NO SOLUTION‘)   #if "solve" doesn’t find a solution
```

```
      else if whattype(sol)=set
              then sol:=allvalues(sol);              #if sol.expr. unique
              elif sol[1]=sol[2] then sol:=sol[1]    #if double sol
                   else  op(map(allvalues,[sol]));   #if sol.expr. not unique
               fi;
        if whattype(sol)=exprseq
           then print(`TWO SOLUTIONS`);
                [ intersection_aux([op(sol[1])],1) ,
                  intersection_aux([op(sol[2])],2) ];
           elif whattype(sol)=set
                then
                   if member(rhs(op(1,sol)),[x,y])
                      or
                      member(rhs(op(2,sol)),[x,y])
                      then print(`INFINITE No. OF SOLS.`)
                      else sol=[op(sol)];
                           print(`UNIQUE SOLUTION`);
                           intersection_aux(sol,0);
                    fi;
         fi;
    fi;
end:
```

```
intersection_aux:=proc(sol::{set,list},n::integer)
  local solu;
  solu:=sol;
  if lhs(op(1,solu))=y
     then solu:=[op(2,solu),op(1,solu)];
   fi;                              #check order for x,y
  [rhs(op(1,solu)),rhs(op(2,solu))];
 end:

isIn:=proc(P::list,o::{equation,list})
  local ec;
  if nops(o)=3 then ec:=op(1,o)    #equation distinguishing
               else ec:=o          #if it is segment or
   fi;                             # (line or circf.)
  evalb(simplify(subs(x=P[1],y=P[2],lhs(ec)))=0);
 end:

perpBis:=proc(s::list)
  perpendicular(s,midpoint(s));
 end:
```

```
angleBis:=proc(P1::list,P2::list,P3::list)
  local P4,d12,d23,P4_1,P4_2;
  if (P1[1]-P2[1])*(P3[2]-P2[2]) = (P3[1]-P2[1])*(P1[2]-P2[2])
     then perpendicular(line(P2,P1),P2);          #if angle=180deg
     else d12:=sqrt(sqdist(P1,P2));
          d23:=sqrt(sqdist(P2,P3));
          P4_1:=P1[1]+(d12/(d12+d23))*(P3[1]-P1[1]);
          P4_2:=P1[2]+(d12/(d12+d23))*(P3[2]-P1[2]);
          line(P2,point(P4_1,P4_2));
   fi;
 end:

pointOnObject:=proc(Px::name,Py::name,o)          #::{equation,list})
  local eq,o_aux;
  global LREL;
  if nops(o)=3 then o_aux:=op(1,o)
               else o_aux:=o
   fi;                                  #The case "segment" is distinguished
  eq:=subs(x=Px,y=Py,o_aux);
  LREL:=[op(LREL),lhs(eq)];
  [Px,Py];
 end:
```

```
read `geo_tran.mpl`:
```

Apéndice II: "Worksheet" con ejemplos

EXAMPLE OF USE OF

"paramGeoN" PARAMETRIC GEOMETRY PACKAGE

AND

Carl Devore's geoTran package

FOR THE COOPERATION OF

"The Geometer's Sketchpad 3" AND "Maple 7"

by

Eugenio Roanes-Lozano

Dept Algebra, Universidad Complutense de Madrid (Spain)

```
> currentdir("C:/sevilla/codigo/"):
```

Remark: Observe that I,x,y are reserved words of Maple (I) and paramGeo (x,y). If used by Sketchpad, they will be substituted by I_,x_,y_ (respectively).

Example 0: Test

STEP 1: The user should begin with a "restart" followed by loading the package paramGeoN, that automatically loads the package geoTran.

The "echo=2" level makes it easier to follow the definitions of the geometrical objects.

```
> restart;
> read "paramGeo.mpl";
> interface(echo=2);
```

STEP 2: The user can then ask the translator to translate a .TXT version of a Sketchpad script (.GSS). The output file will have Maple syntax and will be prepared to use paramGeoN package. The final name is by default the original one and the final extension is .MPL by default. Ayway they can be specified.

```
> `Sketchpad->Maple`(`prueba.txt`);
```

"prueba.mpl"

```
> #`Sketchpad->Maple`(`prueba.txt`, `prueba.mpl`);
```

Remark: Observe that there should be no file with the final name and extension.

File prueba.mpl is ready to be used.

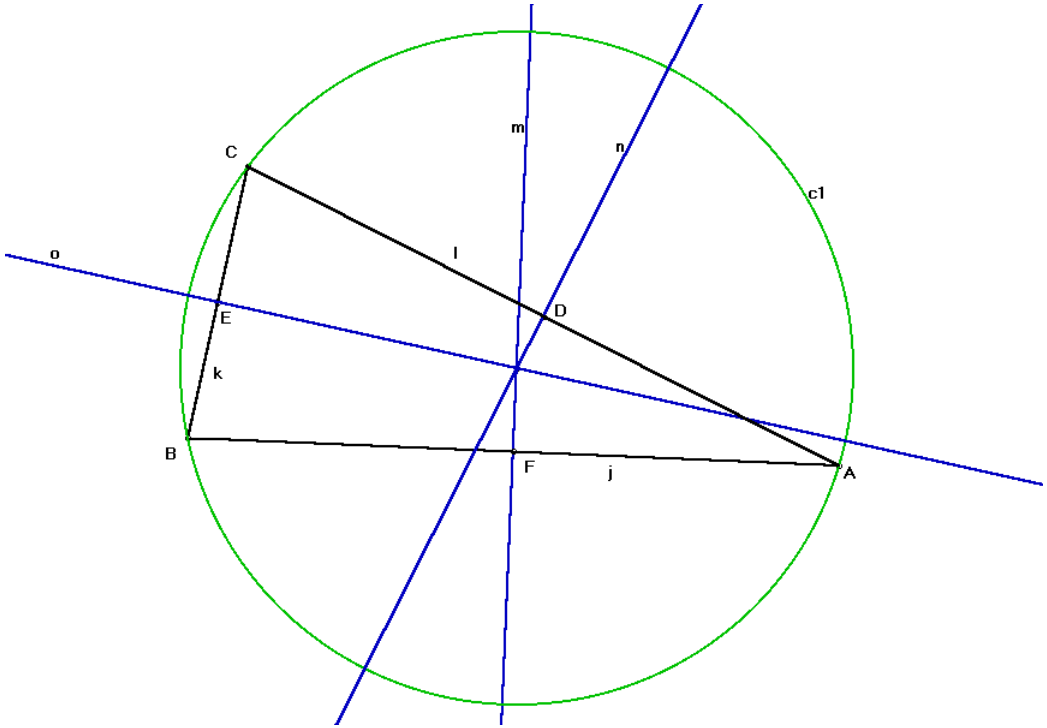
Example 1: Existence of circumcentre

STEP 2: The user can then ask the translator to translate a .TXT version of a Sketchpad script (.GSS).

```
> `Sketchpad->Maple`(`circum.txt`);
```

"circum.mpl"

Remark: Observe that there should be no file with the final name and extension.



STEP 3: The new .MPL can be read with Maple's "read" command, and read lines will be executed while reading. Therefore, any asigning to the parameters of the "Given" points should be done before reading the translation. Otherwise the coordinates of the given points will be their names concatenated with _x and _y. For example, in this case, A=(0,0) and B=(1,0).

```
> A_x:=0:
> A_y:=0:
> B_x:=1:
> B_y:=0:
```

STEP 4: Let us read the translated file

```
> read(‘circum.mpl‘);
```

```
# Sketchpad to Maple automatic translation by Carl Devore
# circum.txt -> circum.mpl
# circum.gss
#
#Given:
> A:=point(A_x, A_y);
```

$$A := [0,\ 0]$$

```
> B:=point(B_x, B_y);
```

$$B := [1,\ 0]$$

```
> C:=point(C_x, C_y);
```

$$C := [C\_x,\ C\_y]$$

```
#---------------
#Steps:
> j:=segment(A, B);
```

$$j := [y = 0,\ [0,\ 0],\ [1,\ 0]]$$

```
> k:=segment(B, C);
```

$$k := [(C\_x - 1)\, y - C\_y\, x + C\_y = 0,\ [1,\ 0],\ [C\_x,\ C\_y]]$$

```
> l:=segment(C, A);
```

$$l := [-y\, C\_x + C\_y\, x = 0,\ [C\_x,\ C\_y],\ [0,\ 0]]$$

```
> D:=midpoint(l);
```

$$\mathrm{D} := [\frac{1}{2}\, C\_x,\ \frac{1}{2}\, C\_y]$$

```
> E:=midpoint(k);
```

$$E := [\frac{1}{2}\, C\_x + \frac{1}{2},\ \frac{1}{2}\, C\_y]$$

```
> F:=midpoint(j);
```

$$F := [\frac{1}{2},\ 0]$$

```
> m:=perpendicular(j, F);
```

$$m := x - \frac{1}{2} = 0$$

```
> n:=perpendicular(l, D);
```

$$n := -C\_x\,(x - \frac{1}{2}\, C\_x) - C\_y\,(y - \frac{1}{2}\, C\_y) = 0$$

```
> o:=perpendicular(k, E);
```

$$o := (C\_x - 1)\,(x - \frac{1}{2}\, C\_x - \frac{1}{2}) + C\_y\,(y - \frac{1}{2}\, C\_y) = 0$$

```
> G:=intersection(n, o);
```

UNIQUE SOLUTION

$$G := [\frac{1}{2},\ \frac{1}{2}\, \frac{-C\_x + C\_x^2 + C\_y^2}{C\_y}]$$

```
> c1:=circumCP(G, A);
```

$$c1 := (x - \frac{1}{2})^2 + (y - \frac{1}{2}\,\frac{-C\_x + C\_x^2 + C\_y^2}{C\_y})^2 - \frac{1}{4} - \frac{1}{4}\,\frac{(-C\_x + C\_x^2 + C\_y^2)^2}{C\_y^2} = 0$$

STEP 5: Let's go on with the algebraic process (this is a specially simple case).

In this case it is straightforward: we can either directly ask (by hand) if G (intersection of perp. bis. n and o) is in the third perpendicular bisector (m):

```
> G;
```

$$[\frac{1}{2},\ \frac{1}{2}\,\frac{-C\_x + C\_x^2 + C\_y^2}{C\_y}]$$

```
> m;
```

$$x - \frac{1}{2} = 0$$

```
> isIn(G,m);
```

$$true$$

and directly check (by hand) that the distances to the three vertices are equal:

```
> simplify(sqdist(G,A)-sqdist(G,B));
```

$$0$$

```
> simplify(sqdist(G,A)-sqdist(G,C));
```

$$0$$

and check that the the circumscribed circumference (the circumference of center G that passes through A, denoted c1 by Sketchpad) passes through the other two vertices:

```
> isIn(B,c1);
```

$$true$$

```
> isIn(C,c1);
```

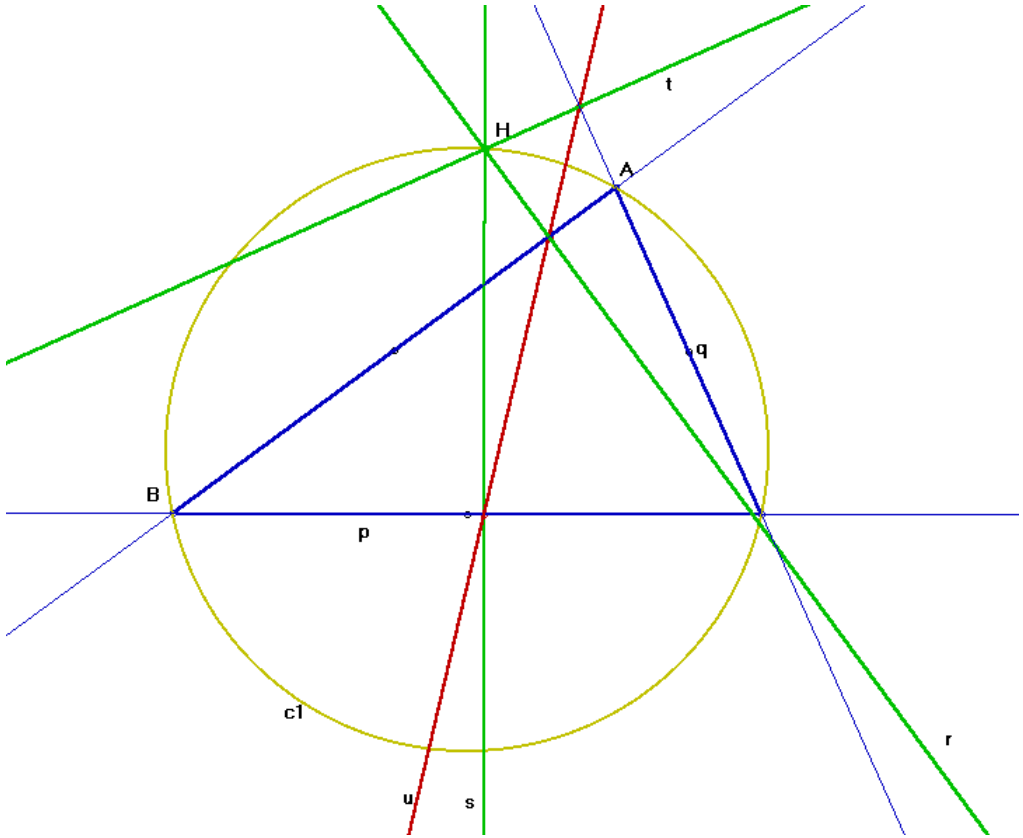
$$true$$

Example 2: Simson's Theorem

STEP 2: The user can ask the translator to translate the .TXT version of the Sketchpad script (.GSS).

```
> `Sketchpad->Maple`(`simson.txt`);
> #variable I was renamed (by hand)
> #I_ before translating
```

"simson.mpl"



STEP 3: Particular assignation of coordinates:

```
> A_x:=0:
> A_y:=0:
> B_x:=1:
> B_y:=0:
```

STEP 4: Let us read the translated file

```
> read(‘simson.mpl‘);
```

```
# Sketchpad to Maple automatic translation by Carl Devore
# simson.txt -> simson.mpl
# Simson.gss
#
#Given:
> A:=point(A_x, A_y);
```

$$A := [0,\ 0]$$

```
> B:=point(B_x, B_y);
```

$$B := [1,\ 0]$$

```
> C:=point(C_x, C_y);
```

$$C := [C\_x,\ C\_y]$$

```
#---------------
#Steps:
```

```
> j:=segment(A, B);
```

$$j := [y = 0,\ [0,\ 0],\ [1,\ 0]]$$

```
> k:=segment(B, C);
```

$$k := [(C\_x - 1)\, y - C\_y\, x + C\_y = 0,\ [1,\ 0],\ [C\_x,\ C\_y]]$$

```
> l:=segment(C, A);
```

$$l := [-y\, C\_x + C\_y\, x = 0,\ [C\_x,\ C\_y],\ [0,\ 0]]$$

```
> D:=midpoint(l);
```

$$\mathrm{D} := [\frac{1}{2}\, C\_x,\ \frac{1}{2}\, C\_y]$$

```
> E:=midpoint(k);
```

$$E := [\frac{1}{2}\, C\_x + \frac{1}{2},\ \frac{1}{2}\, C\_y]$$

```
> F:=midpoint(j);
```

$$F := [\frac{1}{2},\ 0]$$

```
> m:=perpendicular(j, F);
```

$$m := x - \frac{1}{2} = 0$$

```
> n:=perpendicular(l, D);
```

$$n := -C\_x\, (x - \frac{1}{2}\, C\_x) - C\_y\, (y - \frac{1}{2}\, C\_y) = 0$$

```
> G:=intersection(n, m);
```

UNIQUE SOLUTION

$$G := [\frac{1}{2},\ \frac{1}{2}\, \frac{-C\_x + C\_x^2 + C\_y^2}{C\_y}]$$

```
> c1:=circumCP(G, A);
```

$$c1 := (x - \frac{1}{2})^2 + (y - \frac{1}{2}\, \frac{-C\_x + C\_x^2 + C\_y^2}{C\_y})^2 - \frac{1}{4} - \frac{1}{4}\, \frac{(-C\_x + C\_x^2 + C\_y^2)^2}{C\_y^2} = 0$$

```
> o:=line(A, B);
```

$$o := y = 0$$

```
> p:=line(B, C);
```

$$p := (C\_x - 1)\, y - C\_y\, x + C\_y = 0$$

```
> q:=line(C, A);
```

$$q := -y\, C\_x + C\_y\, x = 0$$

```
> H:=pointOnObject(H_x, H_y, c1);
```

$$H := [H\_x,\ H\_y]$$

```
> r:=perpendicular(o, H);
```

$$r := x - H\_x = 0$$

```
> I_:=intersection(o, r);
```

$$\textit{UNIQUE SOLUTION}$$

$$I\_ := [H\_x,\ 0]$$

```
> s:=perpendicular(p, H);
```

$$s := (C\_x - 1)\,(x - H\_x) + C\_y\,(y - H\_y) = 0$$

```
> J:=intersection(p, s);
```

$$\textit{UNIQUE SOLUTION}$$

$$J := [\frac{C\_x^2\, H\_x - 2\, C\_x\, H\_x + C\_x\, C\_y\, H\_y + H\_x - C\_y\, H\_y + C\_y^2}{C\_y^2 + C\_x^2 - 2\, C\_x + 1},$$
$$\frac{C\_y\,(C\_x\, H\_x - H\_x - C\_x + C\_y\, H\_y + 1)}{C\_y^2 + C\_x^2 - 2\, C\_x + 1}]$$

```
> t:=perpendicular(q, H);
```

$$t := -C\_x\,(x - H\_x) - C\_y\,(y - H\_y) = 0$$

```
> K:=intersection(q, t);
```

UNIQUE SOLUTION

$$K := [\frac{(C\_x\, H\_x + C\_y\, H\_y)\, C\_x}{C\_x^2 + C\_y^2}, \frac{C\_y\, (C\_x\, H\_x + C\_y\, H\_y)}{C\_x^2 + C\_y^2}]$$

```
> u:=line(K, I_);
```

$$u := -\frac{C\_y\, (-C\_y\, H\_x + C\_x\, H\_y)\, y}{C\_x^2 + C\_y^2} - \frac{C\_y\, (-C\_x\, H\_x - C\_y\, H\_y)\, x}{C\_x^2 + C\_y^2}$$
$$- \frac{C\_y\, (C\_x\, H\_x^2 + H\_y\, C\_y\, H\_x)}{C\_x^2 + C\_y^2} = 0$$

STEP 4: Let's have a look at the relations among variables automatically derived by paramGeo from "pointOnObject" declarations:

```
> LREL;
```

$$[(H\_x - \frac{1}{2})^2 + (H\_y - \frac{1}{2}\, \frac{-C\_x + C\_x^2 + C\_y^2}{C\_y})^2 - \frac{1}{4} - \frac{1}{4}\, \frac{(-C\_x + C\_x^2 + C\_y^2)^2}{C\_y^2}]$$

There is only one in this case, that will be the hypothesis.

STEP 5: Let's go on with the algebraic process.

Let us apply standard Aut. Th. Proving Methods:

```
> with (Groebner):
```

Let us give the hypothesis and thesis polynomials:

```
> hyp_pol:=op(1,LREL):                          #H is on circle c1
> thesis_pol:=subs(x=op(1,J),y=op(2,J),lhs(u));   #J is on line u
```

$$thesis\_pol := -\frac{C\_y^2\, (-C\_y\, H\_x + C\_x\, H\_y)\, (C\_x\, H\_x - H\_x - C\_x + C\_y\, H\_y + 1)}{(C\_x^2 + C\_y^2)\, (C\_y^2 + C\_x^2 - 2\, C\_x + 1)} -$$
$$C\_y\, (-C\_x\, H\_x - C\_y\, H\_y)$$
$$(C\_x^2\, H\_x - 2\, C\_x\, H\_x + C\_x\, C\_y\, H\_y + H\_x - C\_y\, H\_y + C\_y^2)/((C\_x^2 + C\_y^2)$$
$$(C\_y^2 + C\_x^2 - 2\, C\_x + 1)) - \frac{C\_y\, (C\_x\, H\_x^2 + H\_y\, C\_y\, H\_x)}{C\_x^2 + C\_y^2}$$

METHOD of equal bases (the thesis_pol doesn't add anything to the hypothesis,):

```
> gbasis([thesis_pol,hyp_pol],tdeg(H_x,H_y));
```

$$[C\_y\,H\_x^2 - C\_y\,H\_x + C\_y\,H\_y^2 + C\_x\,H\_y - H\_y\,C\_x^2 - C\_y^2\,H\_y]$$

```
> gbasis([hyp_pol],tdeg(H_x,H_y));
```

$$[C\_y\,H\_x^2 - C\_y\,H\_x + C\_y\,H\_y^2 + C\_x\,H\_y - H\_y\,C\_x^2 - C\_y^2\,H\_y]$$

METHOD of basis = <1>:

```
> gbasis([1-alpha*(thesis_pol),hyp_pol],tdeg(H_x,H_y,alpha));
```

$$[1]$$

FIN