# Une expérience METAPOST

ou

# comment j'ai repris le goût de dessiner

Denis ROEGEL

roegel@loria.fr

Journées GUTenberg

Mai 1997

## Généralités

## METAFONT et METAPOST

- METAFONT : création de polices (Donald Knuth)
- METAPOST : adaptation de METAFONT pour obtenir une sortie PostScript (John Hobby)

## Caractéristiques

- Langage de description de figures ;
- Adapté aux dessins techniques, géométriques, symétriques, ... ;
- Interfacé avec TEX ;
- Produit du PostScript ;
- Disponible dans le domaine public depuis 1995 ;
- Utilisé par Knuth pour ses livres.

## Choix d’un outil

Mieux vaut bien connaître un programme que d’en connaître superficiellement plusieurs.

Autres systèmes comparables :

- PSTricks
- DraTeX
- XYpic
- PiCTeX
- TeXdraw
- environnement picture et extensions epic, eepic

Avantages de METAPOST : facilité de manipulation des courbes (path), résolution automatique des équations linéaires, et donc écriture *naturelle* des dessins.

## Vertus inculquées par METAPOST

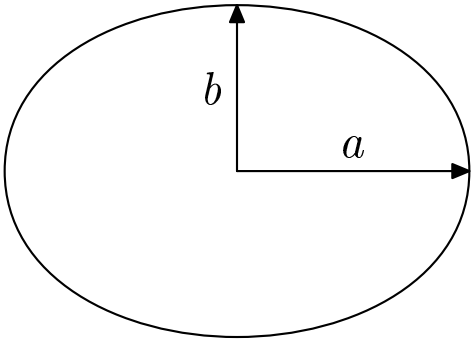
L'utilisation de METAPOST apprend à *observer*, analyser et décomposer les figures à (re)produire.

Lorsqu'on doit *réfléchir*, le dessin a tendance à être *mieux conçu*, à être plus *paramétrable* et plus facilement *réutilisable*.

**L'outil forge l'ouvrier**

# Exemples du manuel

## Ellipse




`beginfig(1);` *Numéro de fichier*

*Définition de deux valeurs numériques* $a$ *et* $b$ *:*

```
a=.7in; b=0.5in;
```

`z0=(0,0); z1=(a,0); z2=(0,b);` *Trois points*

*Définition de* $z3$ *et* $z4$ *(symétrie) :*

```
z0=.5[z1,z3]=.5[z2,z4];
```

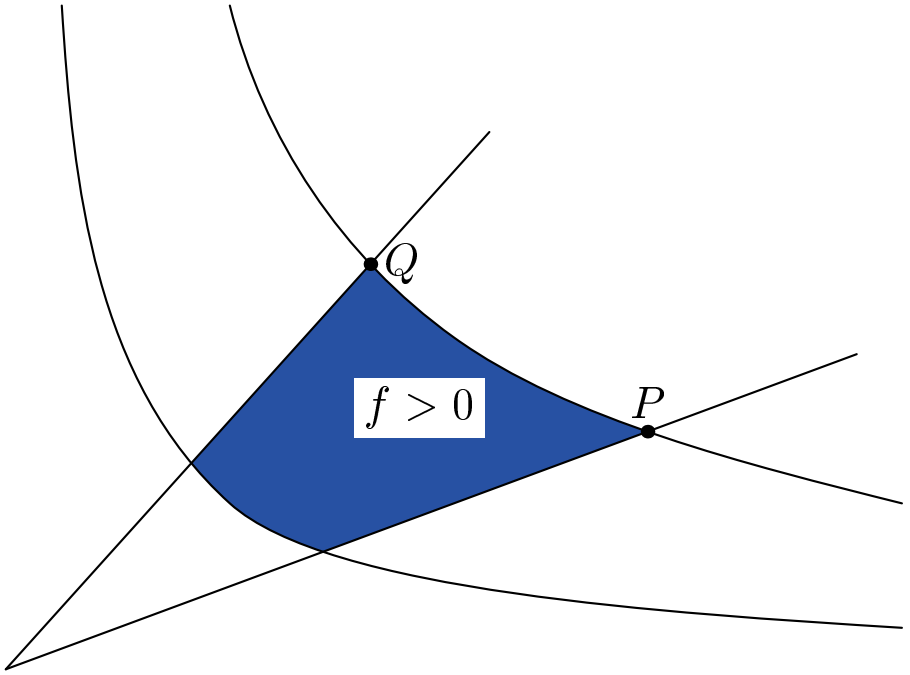
`draw z1..z2..z3..z4..cycle;` *Chemin*

`drawarrow z0..z1;drawarrow z0..z2;` *Arcs*

*Étiquettes :*

```
label.top(btex $a$ etex, .5[z0,z1]);
label.lft(btex $b$ etex, .5[z0,z2]);
endfig;
```

## Intersections




– Définition de quatre chemins

– construction du chemin fermé central avec `buildcycle` :

```
pp = buildcycle(q0.5, p2, q1.5, p4);
```

– coloriage

– construction de l'étiquette :

```
picture lab; lab=thelabel(btex $f>0$ etex, z0);
```

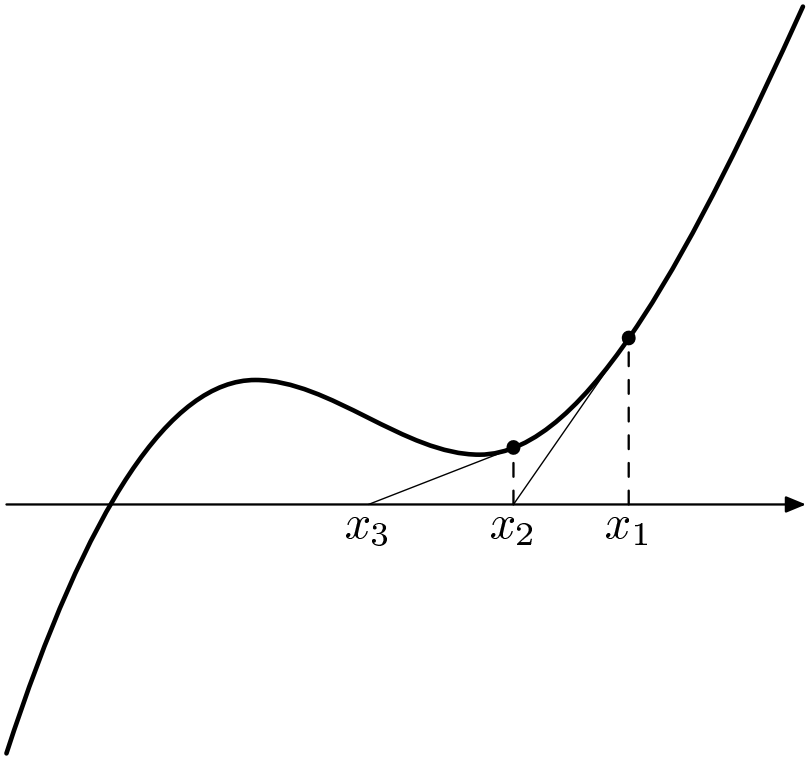
– tracé de l'étiquette après effacement :

```
unfill bbox lab; draw lab;
```

– Étiquettes aux intersections :

```
makelabel.top(btex $P$ etex,
      p2 intersectionpoint q0.5);
```

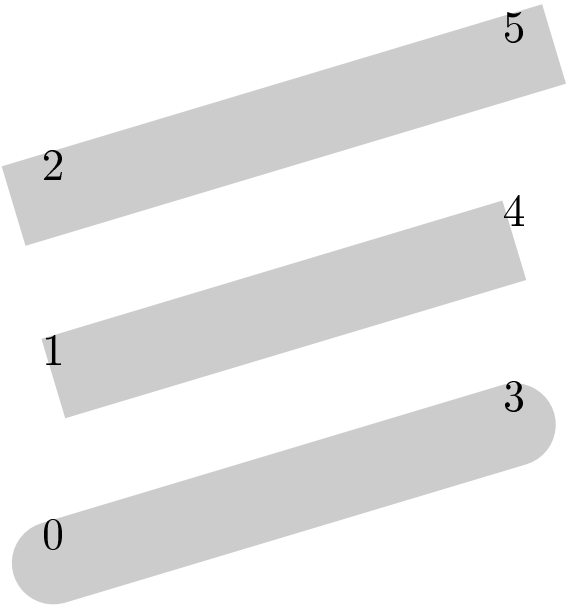
## Tangentes




Difficulté : trouver la direction d'une courbe en un point donné.

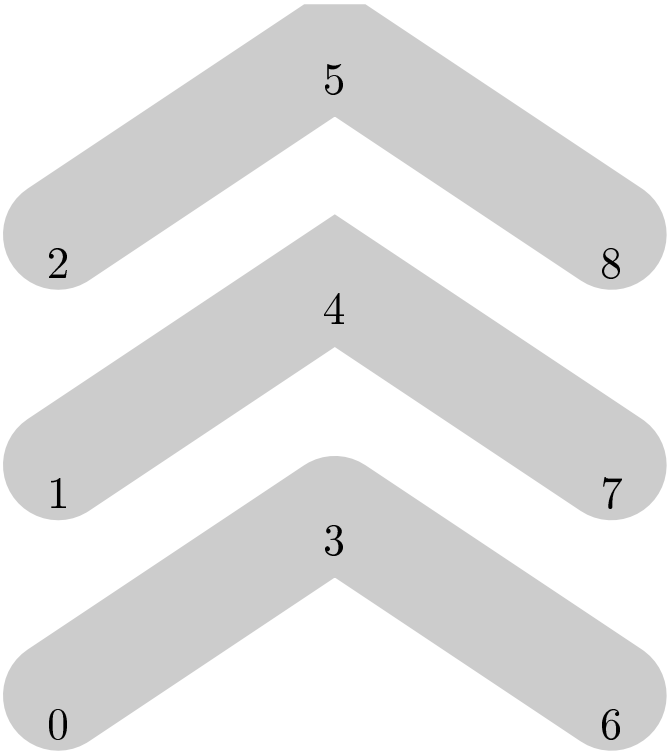
```
beginfig(3);

3.2scf = 2.4in;
path fun;
# = .1;  % Keep the function single-valued
fun = ((0,-1#)..(1,.5#){right}..(1.9,.2#){right}..{curl .1}(3.2,2#))
  scaled scf yscaled(1/#);

vardef vertline primary x = (x,-infinity)..(x,infinity) enddef;
primarydef f atx x = (f intersectionpoint vertline x) enddef;
primarydef f whenx x = xpart(f intersectiontimes vertline x) enddef;

z1a = (2.5scf,0);
z1 = fun atx x1a;
y2a=0; z1-z2a=whatever*direction fun whenx x1 of fun;
z2 = fun atx x2a;
y3a=0; z2-z3a=whatever*direction fun whenx x2 of fun;
...
```

## Formes des bouts de traits



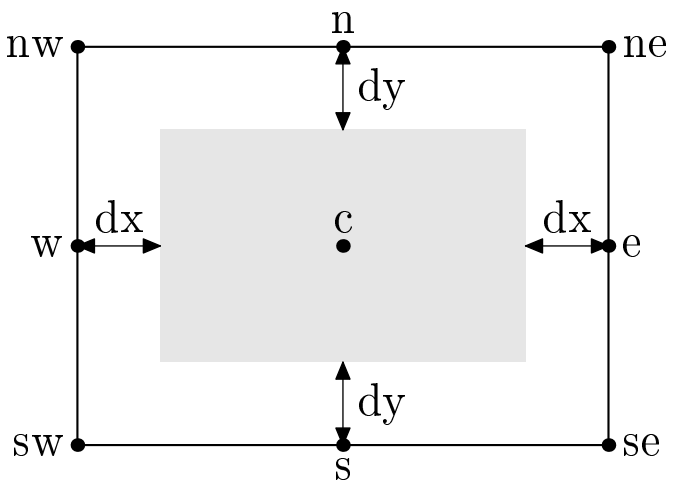

```
beginfig(4);
for i=0 upto 2:
  z[i]=(0,40i); z[i+3]-z[i]=(100,30);
endfor
pickup pencircle scaled 18;
def gray = withcolor .8white enddef;
draw z0..z3 gray;
linecap:=butt; draw z1..z4 gray;
linecap:=squared; draw z2..z5 gray;
labels.top(0,1,2,3,4,5);
endfig;
```

## Angles



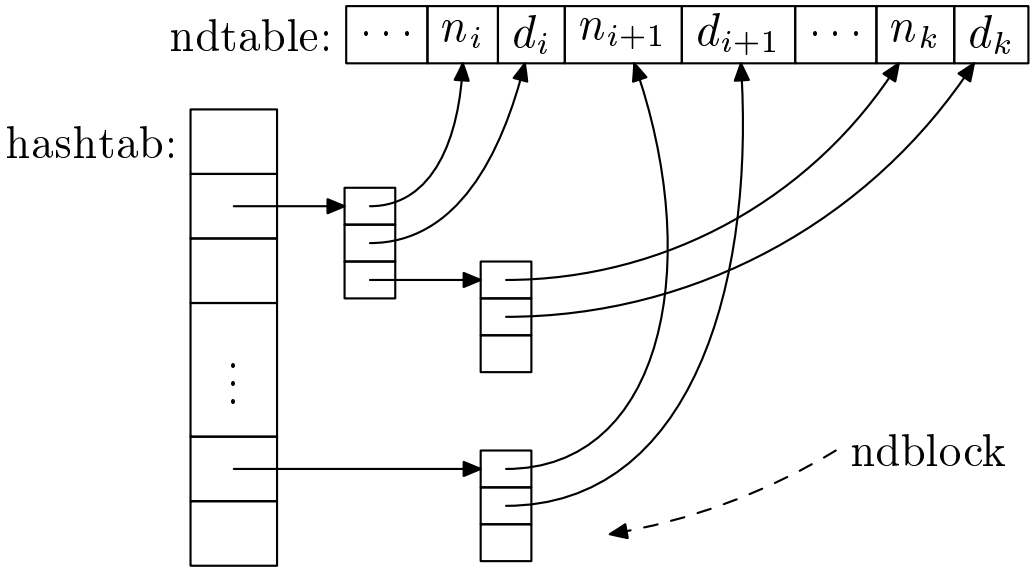

```
beginfig(5);
for i=0 upto 2:
  z[i]=(0,50i); z[i+3]-z[i]=(60,40);
  z[i+6]-z[i]=(120,0);
endfor
pickup pencircle scaled 24;
def gray = withcolor .8white enddef;
draw z0--z3--z6 gray; % linejoin=rounded (par defaut)
linejoin:=mitered; draw z1..z4--z7 gray;
linejoin:=beveled; draw z2..z5--z8 gray;
labels.bot(0,1,2,3,4,5,6,7,8);
endfig;
```

## Boîtes




```
beginfig(6);
fill unitsquare xscaled 1.1in yscaled .7in
         withcolor .9white;
boxit(currentpicture); % boite sans nom
dx = dy = .25in;
clearit; drawboxed();
forsuffixes $=n,c: makelabel.top(str $, $); endfor
makelabel.bot("s",s);
forsuffixes $=ne,e,se: makelabel.rt(str $, $); endfor
forsuffixes $=nw,w,sw: makelabel.lft(str $, $); endfor
pickup pencircle scaled .3bp;
vardef larrow@#(expr a, da, s) =
  drawdblarrow a..a+da; label@#(s,a+.5da); enddef;
larrow.rt(n, (0,-dy), "dy");
larrow.rt(s, (0,dy), "dy");
larrow.top(e, (-dx,0), "dx");
larrow.top(w, (dx,0), "dx");
endfig;
```

## Liens entre boîtes

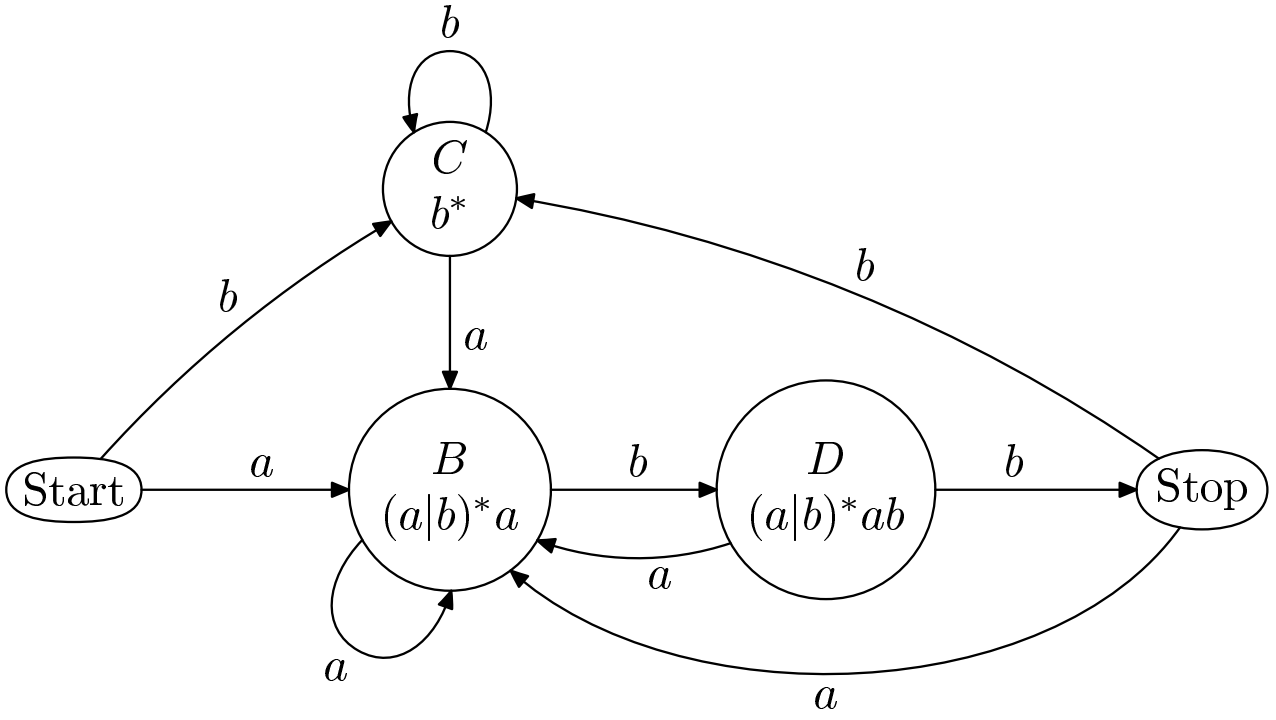



– Flèches vers le centre des boîtes

– Flèches pointillées interrompues

```
beginfig(7);
boxjoin(a.se=b.sw; a.ne=b.nw );
boxit.a(btex $\cdots$ etex);
boxit.ni(btex $n_i$ etex);
...
boxit.aa(pic_.a);

di.dy = 2;
drawboxed(a,ni,di,nii,dii,aa,nk,dk);
label.lft("ndtable:", a.w);

boxjoin(a.sw=b.nw; a.se=b.ne);
interim defaultdy:=7;
boxit.ba();
...
boxit.bd(btex $\vdots$ etex);
bd.dx = 8;
ba.ne = a.sw - (15,10);
drawboxed(ba,bb,bc,bd,be,bf);
label.lft("hashtab:", ba.w);
def ndblock suffix $ =
  boxjoin(a.sw=b.nw; a.se=b.ne);
  forsuffixes $$=$a,$b,$c:
    boxit$$(); ($$dx,$$dy)=(5.5,4);
  endfor;
enddef;
ndblock nda;
...
nda.a.c - bb.c = ndb.a.c - nda.c.c
               = (whatever,0);
xpart ndb.c.se = xpart ndc.a.ne
               = xpart di.c;
ndc.a.c - be.c = (whatever,0);
drawboxes(nda.a,nda.b,nda.c, ...);
drawarrow bb.c .. nda.a.w;
...
drawarrow nda.a.c{right}..{curl0}ni.c
          cutafter bpath ni;
...
x.ptr = xpart aa.c;
y.ptr = ypart ndc.a.ne;
drawarrow subpath (0,.7) of
    (z.ptr..{left}ndc.c.c) dashed evenly
label.rt(btex ndblock etex, z.ptr);
endfig;
```

## Boîtes « circulaires »



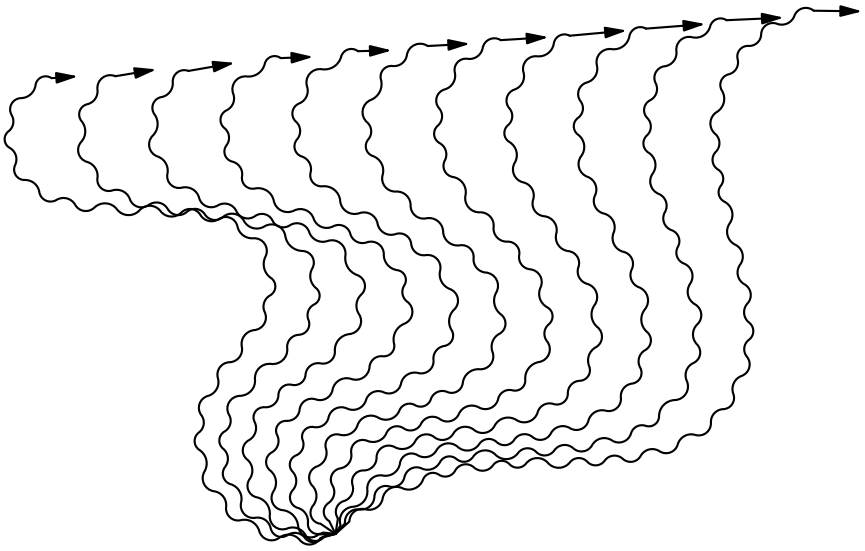
```
beginfig(9);
vardef cuta(suffix a,b) expr p =
  drawarrow p cutbefore bpath.a cutafter bpath.b;
  point .5*length p of p
enddef;

vardef self@# expr p =
  cuta(@#,@#) @#.c{curl0}..@#.c+p..{curl0}@#.c  enddef;

circleit.aa("Start"); aa.dx=aa.dy;
...

numeric hsep;
bb.c-aa.c = dd.c-bb.c = ee.c-dd.c = (hsep,0);
cc.c-bb.c = (0,.8hsep);
xpart(ee.e - aa.w) = 3.8in;
drawboxed(aa,bb,cc,dd,ee);

label.ulft(btex$b$etex, cuta(aa,cc) aa.c{dir50}..cc.c);
label.top(btex$b$etex, self.cc(0,30pt));
...
endfig;
```

## Expérience personnelle : petite galerie

Les exemples ne seront pas étudiés dans le détail, mais certains aspects clés seront isolés.

## Exemple d'analyse de dessin

Les principes sous-jacents à un dessin ne sont pas forcément visible.



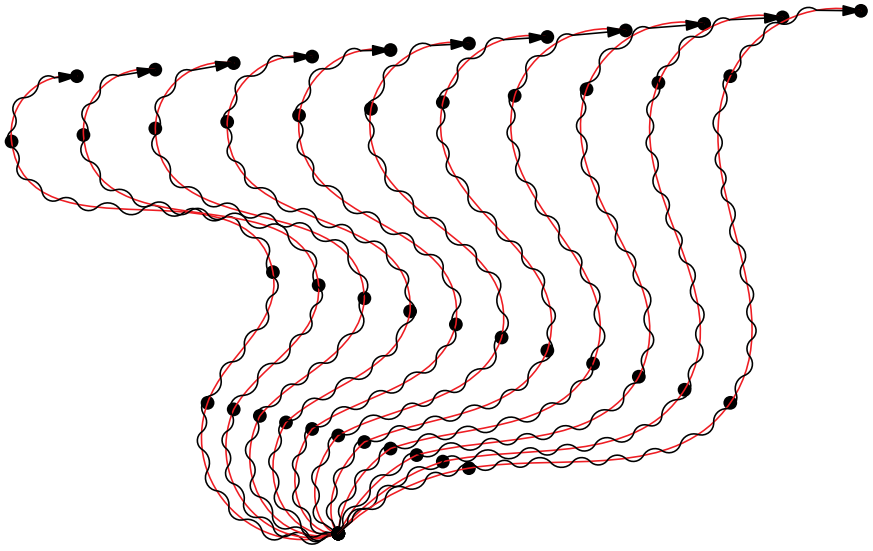
S'agit-il d'une suite de 11 graphes de fonctions ?

En réalité :

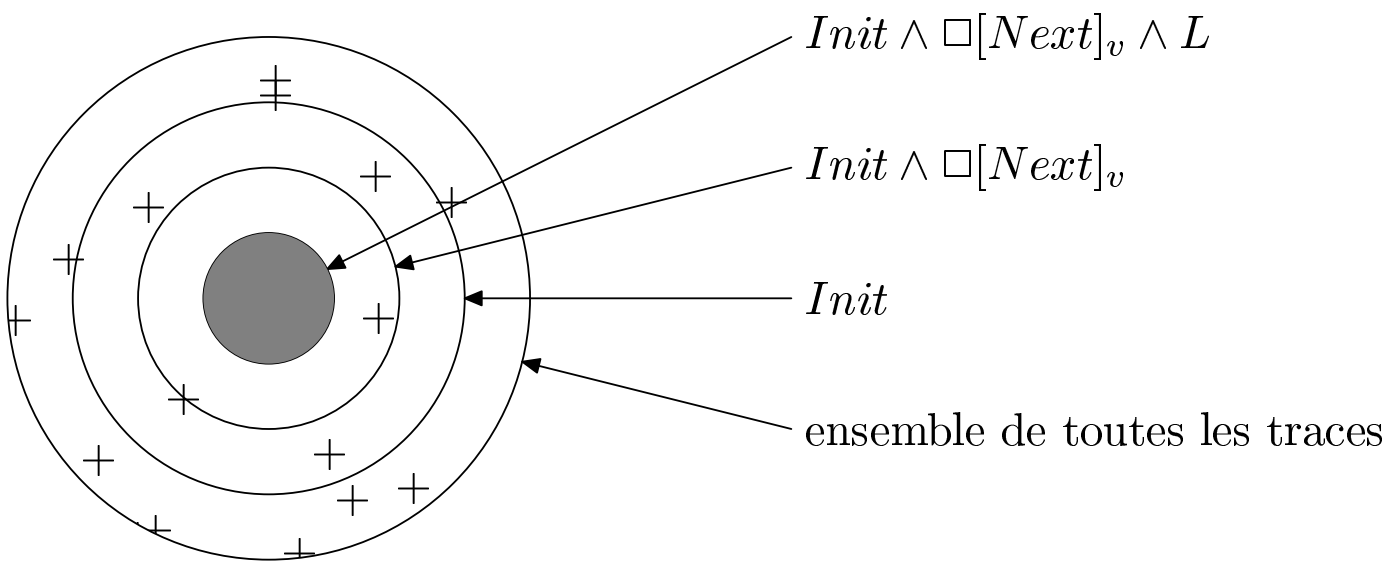
– deux courbes extrêmes, dont les interpolations donnent les neuf autres ;

– chaque courbe est perturbée, par introduction de nouveaux points ;

– les perturbations se font à angle et densité constantes.



## Textes et formules TEX



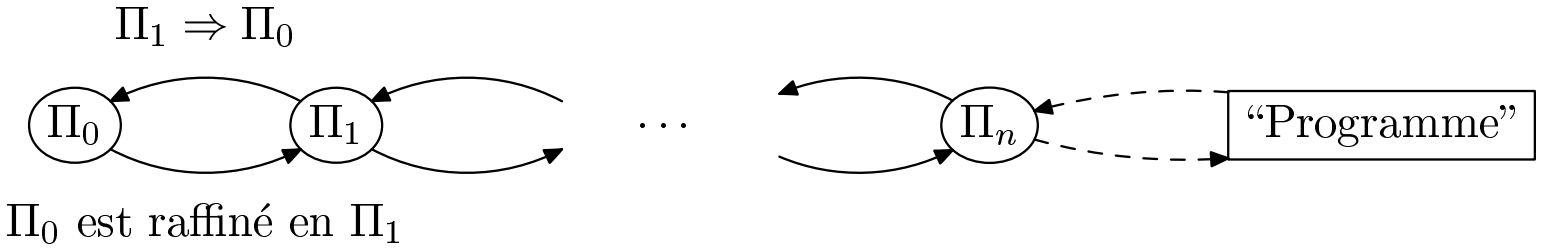

– croix aléatoires

– croix coupées

– flèches passant par le centre des cercles et *interrompues* aux cercles.

```
beginfig(1);
numeric u;
randomseed:=23;
u=1cm;
z0=origin;
x1-x0=4u;x2=x3=x4=x1;y1=2u;y2=u;y3=0;y4=-u;
path c[];
c0=fullcircle scaled u;
c1=fullcircle scaled 2u;
c2=fullcircle scaled 3u;
c3=fullcircle scaled 4u;
random_dots(5u,5u,30);
ran_dots:=ran_dots shifted (-2u,-2u);
draw c0;draw c1;draw c2;draw c3;
clip ran_dots to c3;
draw ran_dots;
label.rt(btex ensemble de toutes les traces etex,z4);
label.rt(btex $Init$ etex,z3);
label.rt(btex $Init\land\Box[Next]_v$ etex,z2);
label.rt(btex $Init\land\Box[Next]_v\land L$ etex,z1);
drawarrow z1--z0 cutafter c0;
drawarrow z2--z0 cutafter c1;
drawarrow z3--z0 cutafter c2;
drawarrow z4--z0 cutafter c3;
fill c0 withcolor (.5,.5,.5);
endfig;
```

## Macros



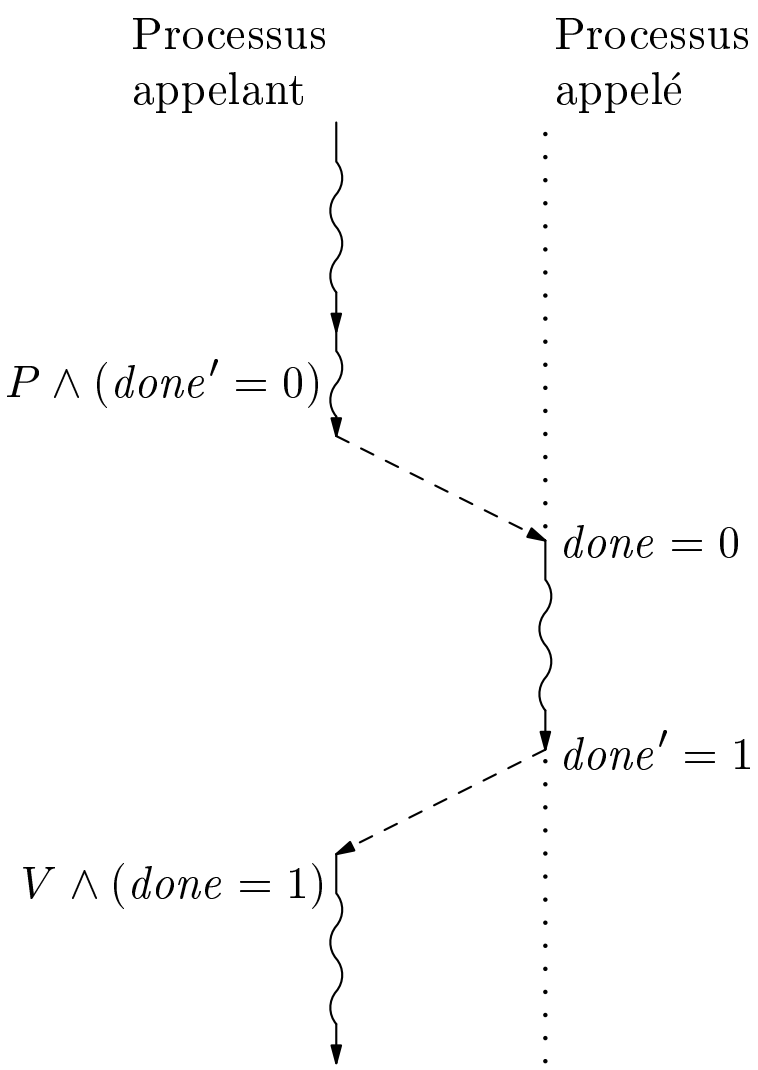

– interruption des arcs sur des ellipses invisibles ;

– macro `link` facilitant la mise en place des couples de flèches.

```
def link(suffix a,b)(expr aa)(text ext)=
    drawarrow a.c{dir (-aa)}..{dir aa}b.c
                          cutbefore bpath a cutafter bpath b ext;
    drawarrow b.c{dir (180-aa)}..{dir (180+aa)}a.c
                          cutbefore bpath b cutafter bpath a ext;
  enddef;

beginfig(2);
numeric d;
d=5mm;
circleit.spec1(btex $\Pi_0$ etex);spec1.dx=spec1.dy;
circleit.spec2(btex $\Pi_1$ etex);spec2.dx=spec2.dy;
```

```
circleit.spec3(btex $\Pi_3$ etex);spec3.dx=spec3.dy;
circleit.spec.pn(btex $\Pi_{n-1}$ etex);spec.pn.dx=spec.pn.dy;
circleit.spec.n(btex $\Pi_n$ etex);spec.n.dx=spec.n.dy;
boxit.prog(btex ‘‘Programme’’ etex);
spec1.c=origin;
spec2.c-spec1.c=(2cm,0);
spec3.c-spec2.c=(2cm,0);
spec.pn.c-spec3.c=(1cm,0);
spec.n.c-spec.pn.c=(2cm,0);
prog.c-spec.n.c=(3cm,0);
drawboxed(spec1,spec2,spec.n,prog);
link(spec1,spec2)(40)();
label.bot(btex $\Pi_0$ est raffin\’e en $\Pi_1$ etex,
          1/2[spec1.c,spec2.c]-(0,d));
label.top(btex $\Pi_1 \Rightarrow \Pi_0$ etex, 1/2[spec1.c,spec2.c]+(0,d));
link(spec2,spec3)(40)();
link(spec.pn,spec.n)(40)();
link(spec.n,prog)(20)(dashed evenly);
picture lab;
lab=thelabel(btex $\ldots$ etex, 1/2[spec3.c,spec.pn.c]);
draw lab;
endfig;
```

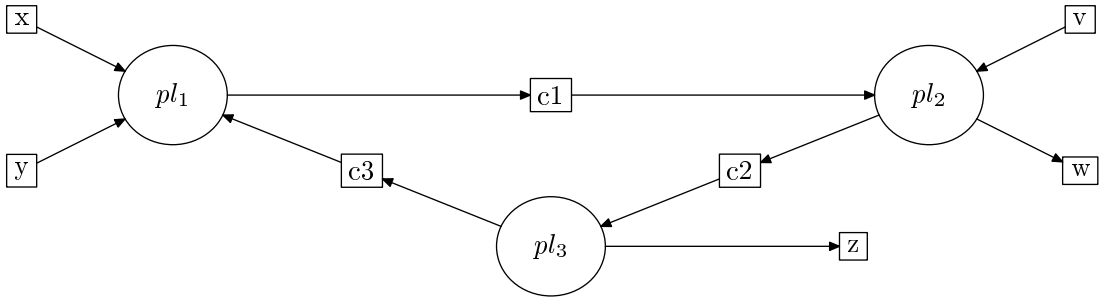
## Processus




– macro `drawcoil`

```
beginfig(2);
numeric v;
v=.8u;
z0=origin;
x0=x1=x2=x5=x6;x3=x4=x7;y0=y7;
y0-y1=y3-y4=y5-y6=2(y1-y2)=2(y2-y3)=2(y4-y5)=2v;
x7-x0=2v;
drawcoil(z0,z1)(btex etex)(5mm,1mm,40);
drawcoil(z1,z2)(btex etex)(5mm,1mm,40);
drawcoil(z3,z4)(btex etex)(5mm,1mm,40);
drawcoil(z5,z6)(btex etex)(5mm,1mm,40);
drawarrow z2--z3 dashed evenly;
drawarrow z4--z5 dashed evenly;
label.ulft(btex \vbox{\hbox{Processus}
                      \hbox{appelant}}etex,z0);
label.urt(btex \vbox{\hbox{Processus}
                     \hbox{appel\'e}}etex,z7);
label.lft(btex $P \land ({\it done}'=0)$ etex,
          .5[z1,z2]);
label.rt(btex ${\it done}=0$ etex,z3);
label.rt(btex ${\it done}'=1$ etex,z4);
label.llft(btex $V \land ({\it done}=1)$ etex,z5);
pickup pencircle scaled 1pt;
draw z7--z3 dashed withdots;
z8-z6=z7-z0;
draw z4--z8 dashed withdots;
endfig;
```

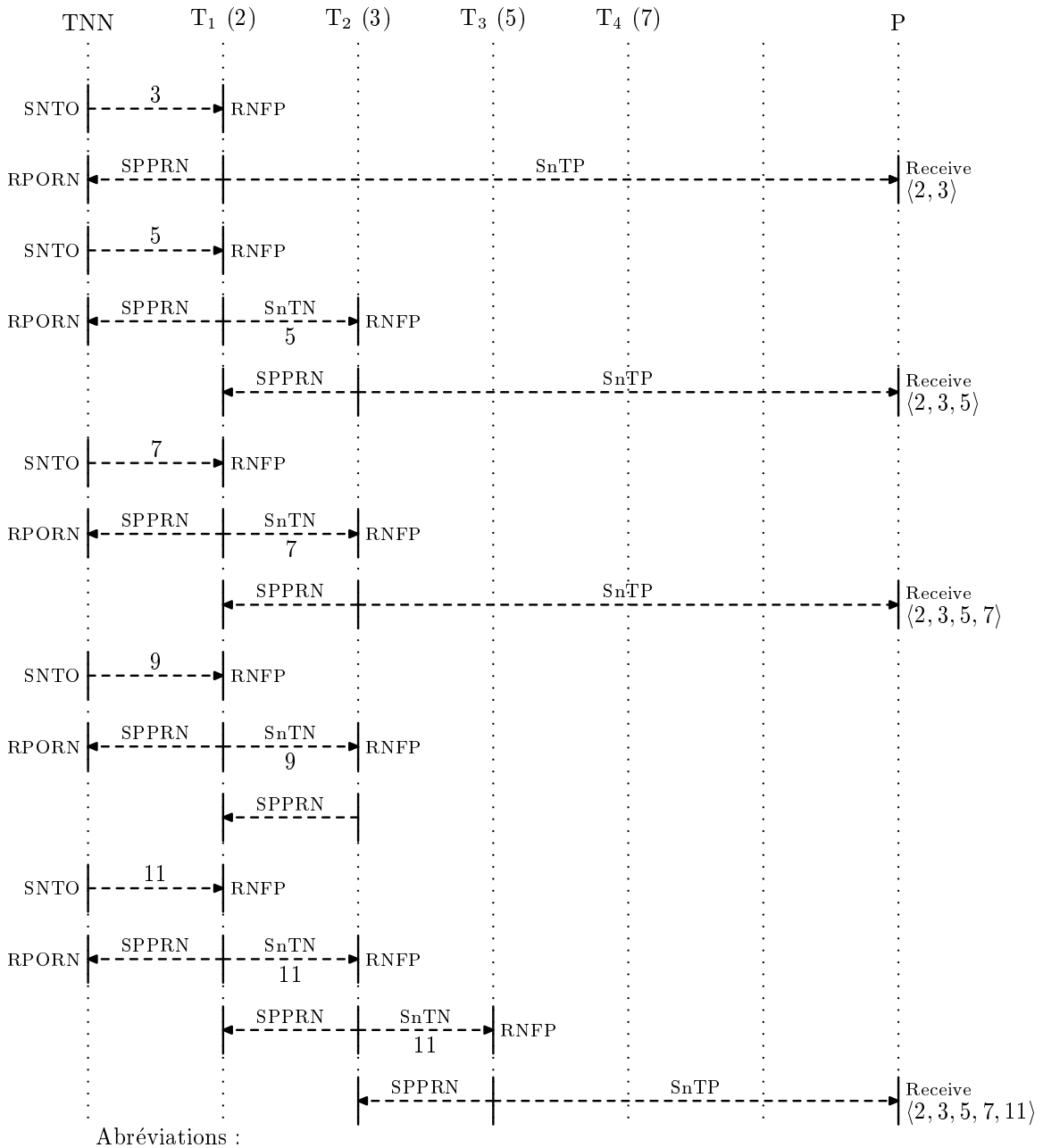
## Réseaux de Rabinovich




– macro `link` pour les flèches réapparaissant souvent

```
beginfig(1);
numeric u;
u=1cm;
circleit.pl1(btex $pl_1$ etex);
    pl1.c=origin;pl1.dx=pl1.dy=.5u;
circleit.pl2(btex $pl_2$ etex);
    pl2.c-pl1.c=(10u,0);pl2.dx=pl2.dy=.5u;
circleit.pl3(btex $pl_3$ etex);
    pl3.c=(5u,-2u);pl3.dx=pl3.dy=.5u;
drawboxed(pl1,pl2,pl3);
z1-pl1.c=(-2u,u);z2-pl1.c=(-2u,-u);z3=.5[pl1.c,pl2.c];
z4-pl2.c=(2u,u);z5-pl2.c=(2u,-u);z6-pl3.c=(4u,0);
z7=.5[pl2.c,pl3.c];z8=.5[pl1.c,pl3.c];
boxit.x(btex x etex);x.c=z1;x.dx=x.dy;
boxit.y(btex y etex);y.c=z2;y.dx=y.dy;
boxit.c1(btex c1 etex);c1.c=z3;c1.dx=c1.dy;
boxit.v(btex v etex);v.c=z4;v.dx=v.dy;
boxit.w(btex w etex);w.c=z5;w.dx=w.dy;
boxit.a(btex z etex);a.c=z6;a.dx=a.dy;
boxit.c2(btex c2 etex);c2.c=z7;c2.dx=c2.dy;
boxit.c3(btex c3 etex);c3.c=z8;c3.dx=c3.dy;
drawboxed(x,y,c1,v,w,a,c2,c3);
link(x,pl1);link(y,pl1);link(pl1,c1);link(c1,pl2);
link(v,pl2);link(pl2,w);link(pl2,c2);link(c2,pl3);
link(pl3,a);link(pl3,c3);link(c3,pl1);
endfig;
```

# Diagrammes temporels




Abréviations :

RNFP : ReceiveNumberFromPrevious

SPPRN : SendPreviousProcessReadyNotification

SnTP : SendnToPrimes

TNN : TryNewNumber

SnTN : SendnToNext

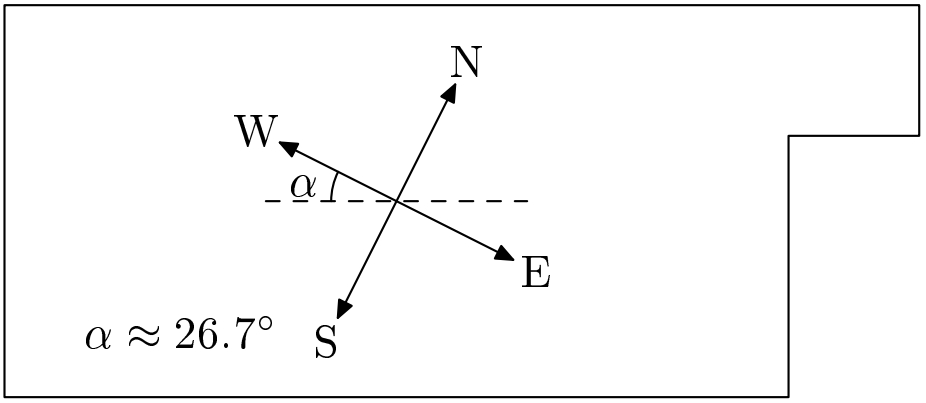
– diagramme construit du haut vers le bas, chaque commande déplaçant un « curseur » vers le bas.

```
def SendReady(expr yy,za,zb)=
    pair zx,zy;
    zx=za+yy*ShiftD;zy=zb+yy*ShiftD;
    draw (zx+(0,.5u))--(zx+(0,-.5u));
    draw (zy+(0,.5u))--(zy+(0,-.5u));
    drawarrow zy--zx dashed evenly;
    label.top(btex\eightrm SPPRN etex,.5[zx,zy]);
  enddef;
...

beginfig(2);
numeric u,DeltaX,DeltaY,yy;
pair ShiftD;
u=7mm;
DeltaX=2cm;DeltaY=1.5u;
ShiftD=down*DeltaY;
z0=origin;
z1-z0=z2-z1=z3-z2=z4-z3=z5-z4=z6-z5=(DeltaX,0);
pickup pencircle scaled 1pt;
for i:=0 upto 6:
    draw z[i]--(z[i]-(0,16cm)) dashed withdots;
endfor;
label.top(btex TNN etex,z0);
label.top(btex T$_1$ (2) etex,z1);
label.top(btex T$_2$ (3) etex,z2);
label.top(btex T$_3$ (5) etex,z3);
label.top(btex T$_4$ (7) etex,z4);
label.top(btex P etex,z6);
```

```
SendNumberToOne(1,z0,z1,btex $3$ etex);
SendNToPrimes(2,z0,z1,z6,btex \pr{2,3} etex,1);
SendNumberToOne(3,z0,z1,btex $5$ etex);
SendNToNext(4,z0,z1,z2,btex $5$ etex,1);
SendNToPrimes(5,z1,z2,z6,btex \pr{2,3,5} etex,0);
SendNumberToOne(6,z0,z1,btex $7$ etex);
SendNToNext(7,z0,z1,z2,btex $7$ etex,1);
SendNToPrimes(8,z1,z2,z6,btex \pr{2,3,5,7} etex,0);
SendNumberToOne(9,z0,z1,btex $9$ etex);
SendNToNext(10,z0,z1,z2,btex $9$ etex,1);
SendReady(11,z1,z2);
SendNumberToOne(12,z0,z1,btex $11$ etex);
SendNToNext(13,z0,z1,z2,btex $11$ etex,1);
SendNToNext(14,z1,z2,z3,btex $11$ etex,0);
SendNToPrimes(15,z2,z3,z6,btex \pr{2,3,5,7,11} etex,0);
label.rt(btex Abr\'eviations~: etex,z0+15.5*ShiftD);
label.rt(btex RNFP~: ReceiveNumberFromPrevious etex,
         .5[z0,z1]+16*ShiftD);
label.rt(
  btex SPPRN~: SendPreviousProcessReadyNotification etex,
         .5[z0,z1]+16.5*ShiftD);
label.rt(btex SnTP~: SendnToPrimes etex,
         .5[z0,z1]+17*ShiftD);
label.rt(btex TNN~: TryNewNumber etex,
         .5[z0,z1]+17.5*ShiftD);
label.rt(btex SnTN~: SendnToNext etex,
         .5[z0,z1]+18*ShiftD);
endfig;
```

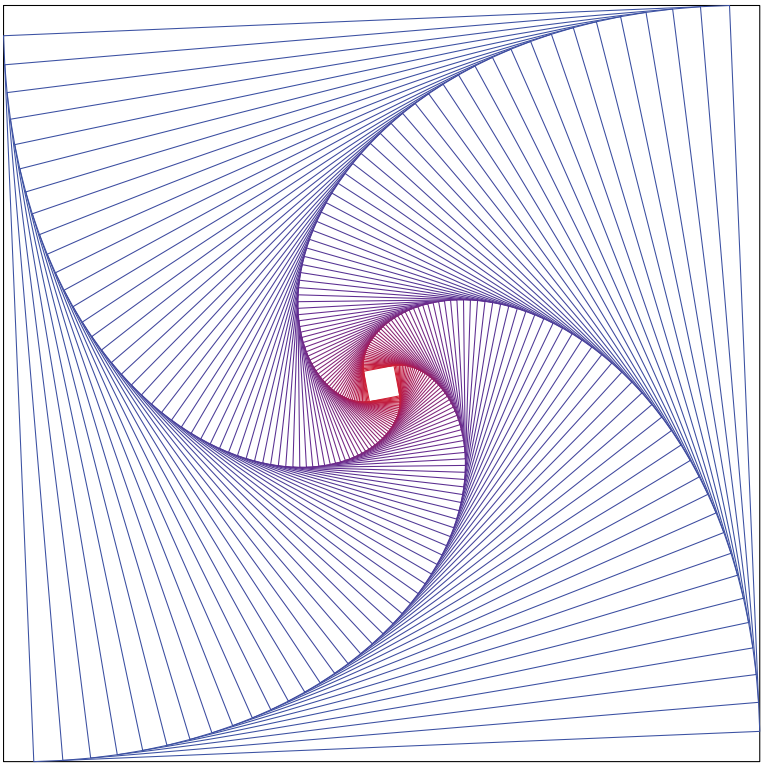
## Rose des vents

Orientation du bureau B219



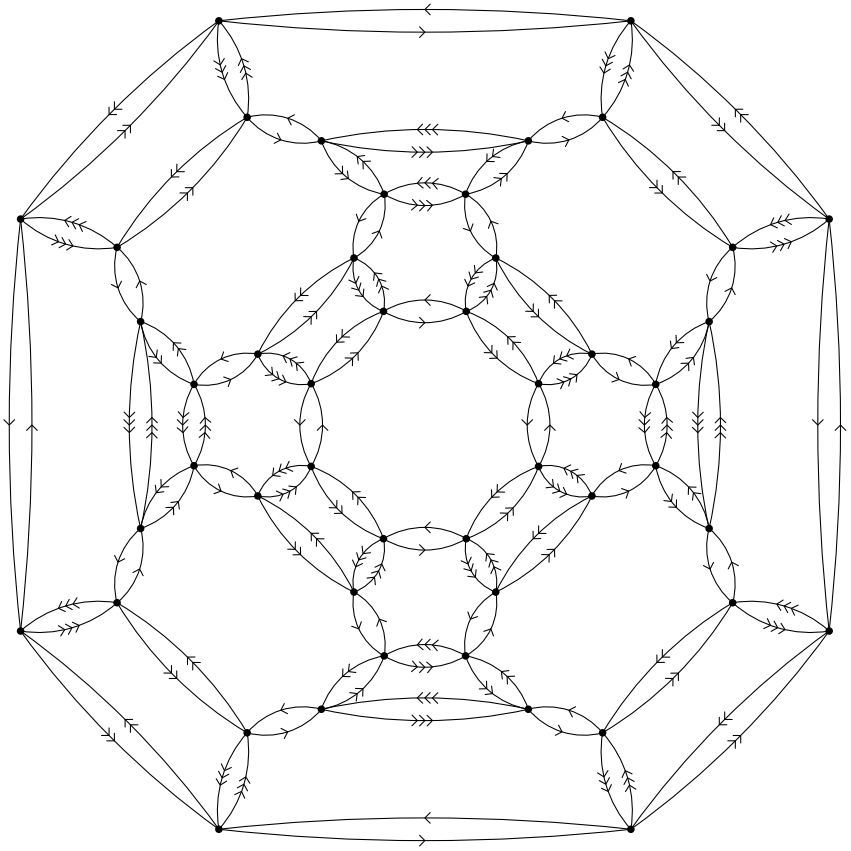
```
def ar(expr a,l) =
    drawarrow (p rotated a shifted z6);
    label(l,z6+ ((1.2u,0) rotated (a-26.7)));
  enddef;

beginfig(1)
u:=1cm;
z0=origin;
z1-z0=(6u,0);z2-z1=(0,2u);
z3-z2=(u,0);z4-z3=(0,u);z4-z5=(7u,0);
draw z0--z1--z2--z3--z4--z5--cycle;
y6=.5[y0,y5];x6=.5[x0,x1];
path p;
p=(origin--(u,0)) rotated -26.7;
ar(90,btex N etex);
ar(180,btex W etex);
ar(270,btex S etex);
ar(0,btex E etex);
draw (z6-(u,0))..(z6+(u,0)) dashed evenly;
draw (z6-(.5u,0)){up}
      ..{dir (90-26.7)}(z6+.5u*(dir (180-26.7)));
label.lft(btex $\alpha$ etex,z6+.5u*dir (180-13));
label.rt(btex $\alpha\approx26.7^\circ$ etex,
         z0+(.5u,.5u));
label.top(btex Orientation du bureau B219 etex,
         .5[z5,z4])
endfig;
```

# Rotations



```
beginfig(1);
path squares[];
numeric side;
side=10cm;
squares0=unitsquare scaled side;
draw squares0;
y1=0;x2=side;
x1=side/25;y2=x1;
d=angle(z2-z1);
r=abs(z2-z1)/side;
for i:=1 upto 80:
squares[i]:=squares[i-1] shifted (-side/2,-side/2)
rotated d scaled r shifted (side/2,side/2);
draw squares[i] withcolor (i/100)[blue,red];
endfor
endfig;
```

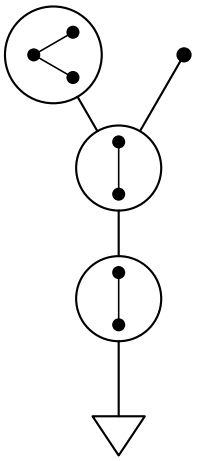
## Symétries : graphes de Cayley



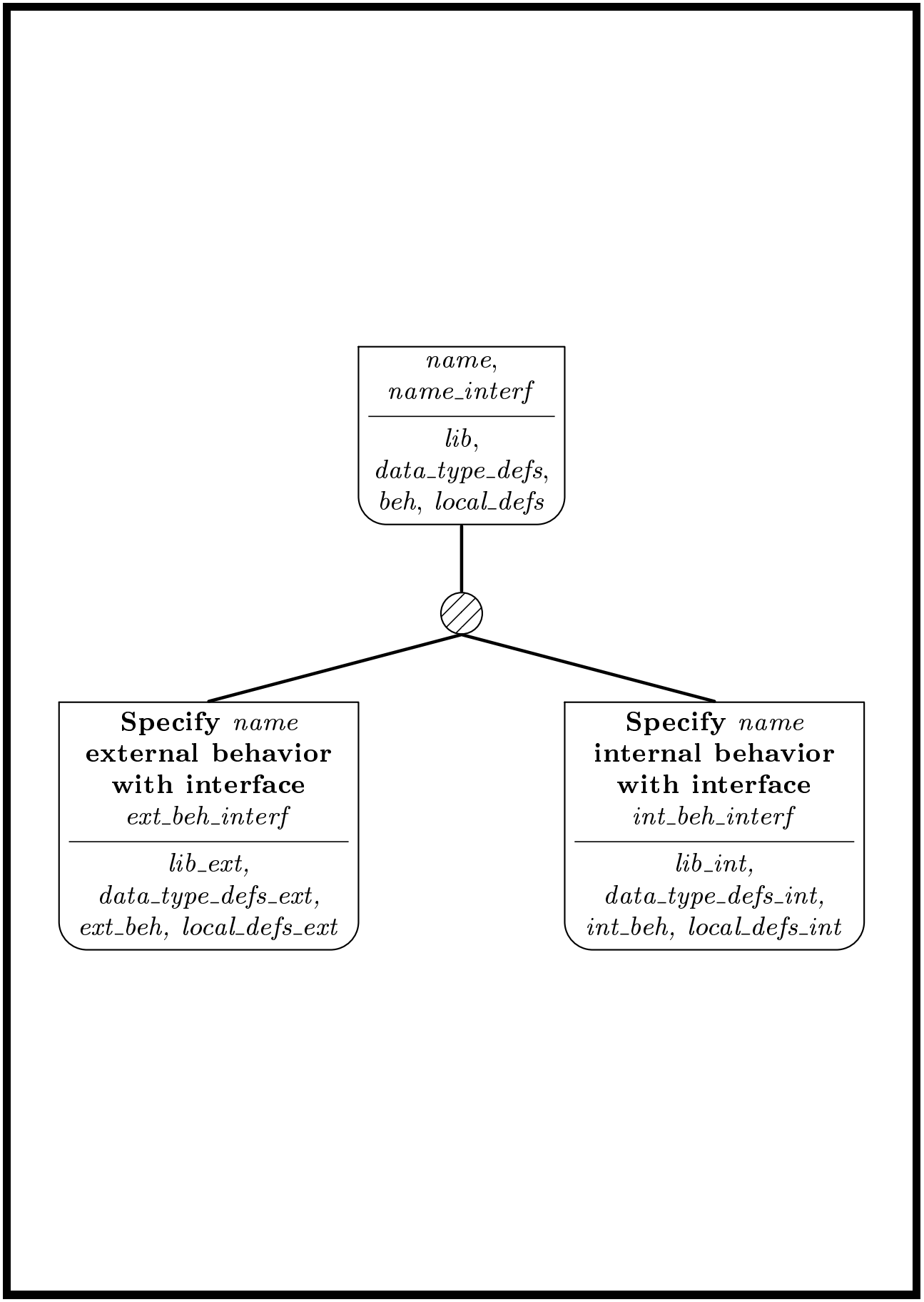
– nombreuses symétries : on ne dessine qu'un huitième du dessin

```
beginfig(1);
numeric u;u=1cm;z0=origin;
z1-z0=polar(1.6u,70);
z2-z0=polar(2.4u,67);
z3-z0=polar(3.1u,80);
z4-z0=polar(4u,70);
z5-z0=polar(4.7u,60);
z6-z0=polar(6u,63);
for i:=1 upto 6:
  z[i+10]=z[i]
     reflectedabout(z0,(100u,100u));
  endfor;
for i:=1 upto 6:
  z[i+20]=z[i] rotatedaround(z0,-90);
  z[i+30]=z[i+10]
             rotatedaround(z0,-90);
     ...
  z[i+70]=z[i+10] rotatedaround(z0,90);
  endfor;
for i:=0 upto 7:
  link(3,z[i*10+1],z[i*10+2]);
  link(1,z[i*10+2],z[i*10+3]);
  link(2,z[i*10+3],z[i*10+4]);
  link(1,z[i*10+4],z[i*10+5]);
  link(3,z[i*10+5],z[i*10+6]);
  endfor;
for i:=0 step 2 until 6:
  link(2,z[i*10+1],z[(i+1)*10+1]);
  link(2,z[i*10+2],z[(i+1)*10+2]);
  link(2,z[i*10+5],z[(i+1)*10+5]);
  link(2,z[i*10+6],z[(i+1)*10+6]);
  endfor;
for i:=1 step 2 until 7:
  link(1,z[i*10+1],
         z[((i+1) mod 8)*10+1]);
  link(3,z[i*10+3],
         z[((i+1) mod 8)*10+3]);
  link(3,z[i*10+4],
         z[((i+1) mod 8)*10+4]);
  link(1,z[i*10+6],
         z[((i+1) mod 8)*10+6]);
  endfor;
pickup pencircle scaled 1mm;
for i:=1 upto 6:
  for j:=0 upto 7: drawdot(z[i+j*10]);
    endfor;endfor;
endfig;
```

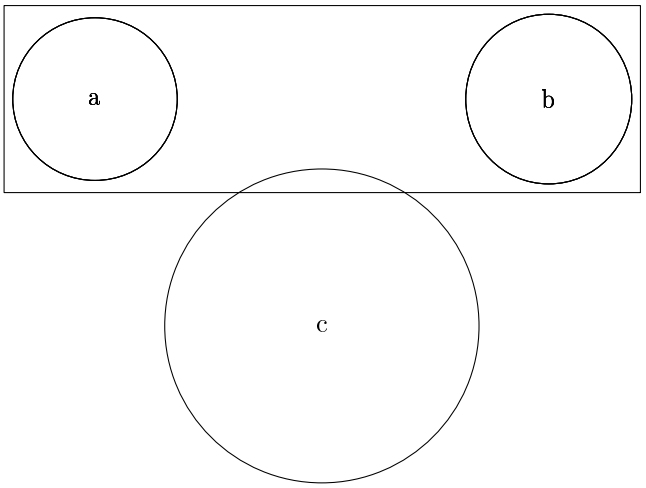
## Arbres, nœuds



```
beginfig(1);
path t; % triangle

% definition des centres des cercles et du triangle
z0=origin;
z1-z0=z2-z1=(0,u);
z3-z2=u*dir(120);z4-z2=u*dir(60);

circleit.un(deuxpoints);un.c=z1;
circleit.deux(deuxpoints);deux.c=z2;
circleit.trois(troispoints);trois.c=z3;
t=triangle shifted z0;
circleit.quatre(unpoint);
      quatre.c=z4;quatre.dx=quatre.dy=0;

drawboxed(un,deux,trois,quatre);draw t;
link(un,deux);link(deux,trois);link(deux,quatre);
draw un.c -- (center t) cutbefore bpath un cutafter t;

endfig
```

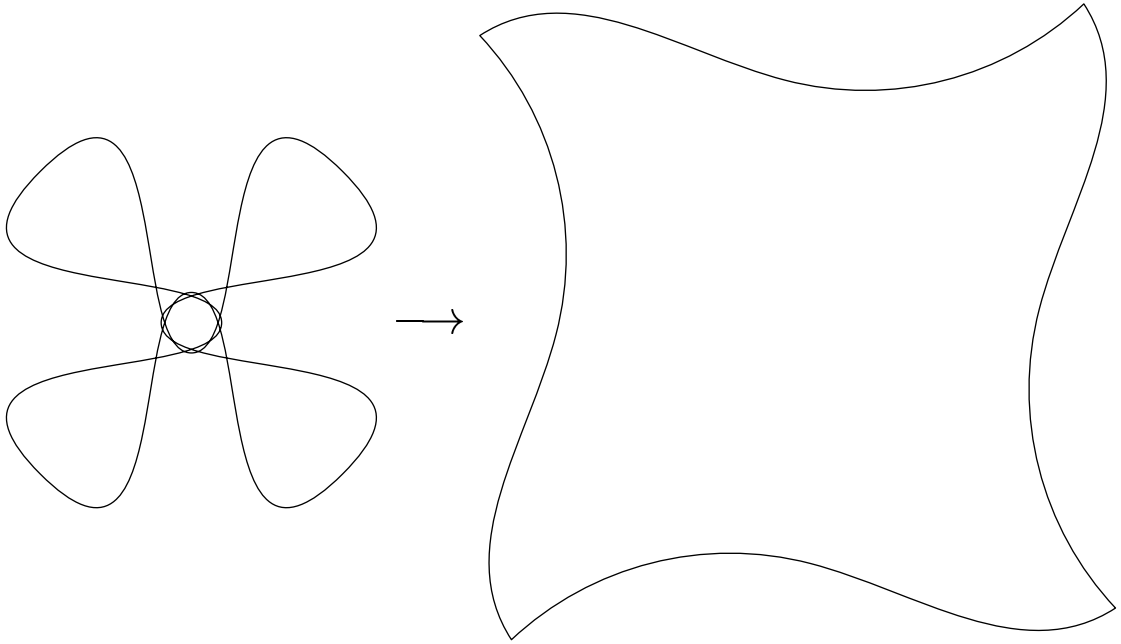

name,
name_interf
lib,
data_type_defs,
beh, local_defs
Specify name
external behavior
with interface
ext_beh_interf
lib_ext,
data_type_defs_ext,
ext_beh, local_defs_ext
Specify name
internal behavior
with interface
int_beh_interf
lib_int,
data_type_defs_int,
int_beh, local_defs_int

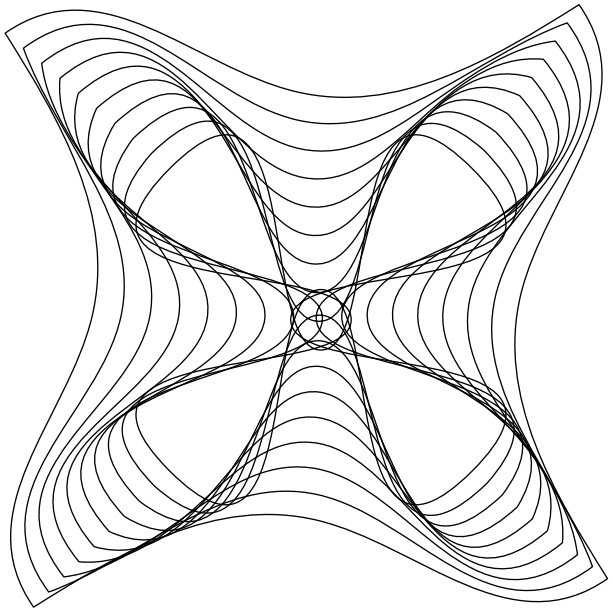
## Entrer dans des objets...




```
beginfig(1);
picture sav_pic;
circleit.a(btex a etex);
circleit.b(btex b etex);
circleit.c(btex c etex);
a.dx=a.dy=1cm;
b.dx=b.dy=1cm;
c.dx=c.dy=2cm;
c.c=origin;
a.c-c.c=(b.c-c.c) rotated 90;
xpart(a.c+b.c)=0;
ypart(a.c)=3cm;
drawboxed(a,b,c);
sav_pic:=currentpicture;
currentpicture:=nullpicture;
drawboxed(a,b);
boxit.d(currentpicture);
drawboxed(d);
draw sav_pic;
endfig;
```

## *Morphing*

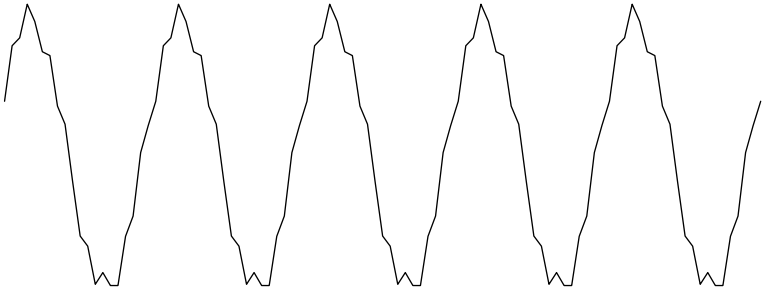




– emploi de `interpath`

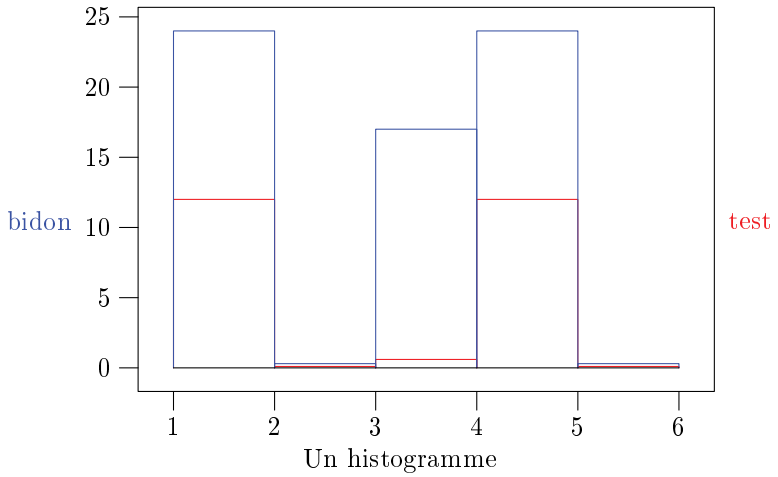
```
beginfig(1);
path p[];
u=2cm;
p1=(0,0){dir-45}..(1,1.2)..{dir45}(2,0);
p2=p1..(p1 rotated 90 shifted (2,0));
p3=p2..(p2 rotated 180 shifted (2,2))..cycle;
p3:=p3 scaled u;
p4:=(0,0)..(1,0.2)..{dir 30}(2,0);
p4:=p4..(p4 rotated 90 shifted (2,0));
p4:=p4..(p4 rotated 180 shifted (2,2))..cycle;
p4:=p4 rotated 3 shifted (-0.5,-0.5) scaled 2u;
for i:=0 step 0.1 until 1:
  draw interpath(i,p3,p4);
endfor;
endfig;
```

## Fonctions



```
numeric h,i;
path p;
def plot (text f)(expr a,b,n) =
  h:=(b-a)/n;
  x:=a;
  p:= (a,f)
  for x:=a+h step h until b:
    ---(x,f)
  endfor;
enddef;
beginfig(1);
plot(sind(x)+.5cosd(x+27)+.05(x mod 5))(0,360*5,100);
draw p xscaled (10cm/(360*5)) yscaled 2cm; % withcolor green;
endfig;
```

# Histogrammes

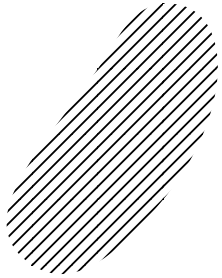



– Données rangées dans un fichier :

```
1       12      24
2       0.1     0.3
3       0.6     17
4       12      24
5       0.1     0.3
6       0.6     17
```

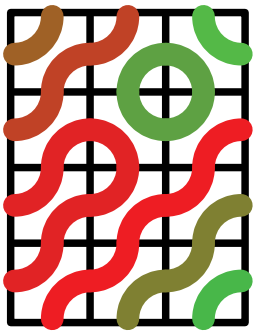
```
input graph

vardef drawhistogram(expr p,c)=
    numeric l;path q;pair zz[];
    l=length p;
    for j:=1 upto l:
      zz0:=point (j-1) of p;
      zz1:=point j of p;
      q:=(xpart zz0,0)--zz0
          --(xpart zz1,ypart zz0)--(xpart zz1,0);
      gdraw q withcolor c;
      endfor;
  enddef;

vardef drawbottomline(expr p)=
    gdraw (xpart point 0 of p,0)
          --(xpart point length p of p,0) ;
  enddef;

beginfig(1);
draw begingraph(3in,2in);
  path p[];
  gdata("test.d",$,
        augment.p1(scantokens $1,scantokens $2););
  gdata("test.d",$,
        augment.p2(scantokens $1,scantokens $3););
  drawhistogram(p1,red);drawhistogram(p2,blue);
  drawbottomline(p1);
  glabel.bot(btex Un histogramme etex,OUT);
  glabel.lft(btex bidon etex,OUT) withcolor blue;
  glabel.rt(btex test etex,OUT) withcolor red;
  endgraph;
endfig;
```

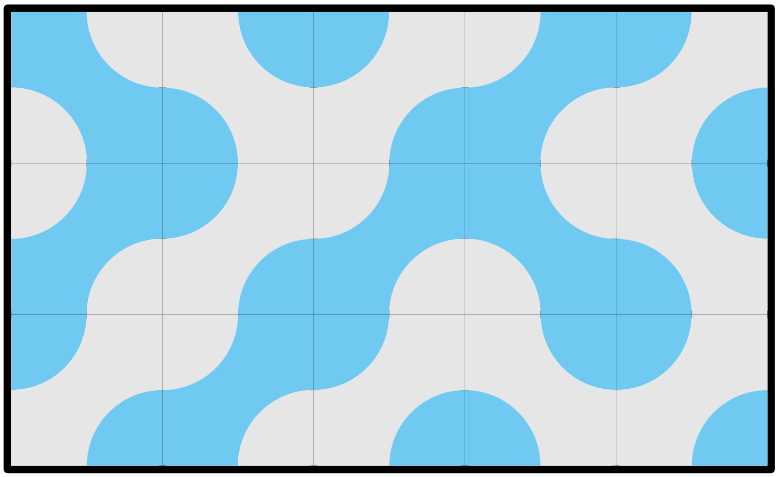
## Textures



```
% texture1 : traits a 45 degres
currentpicture:=nullpicture;
pickup pencircle scaled .4pt;
for i:=1 upto 10: draw (i*1mm,0)--(1cm+i*1mm,1cm); endfor;
clip currentpicture to (5mm,0)--(1cm,0)--(1cm,5mm)--(5mm,5mm)--cycle;
texture1=currentpicture shifted (-5mm,0);

path p;
p=(0,0)..(1cm,5mm)..(1cm,2cm)..(5mm,15mm)..cycle;

beginfig(1);
fillt(p,texture[1]);
endfig;
```

## Pavages

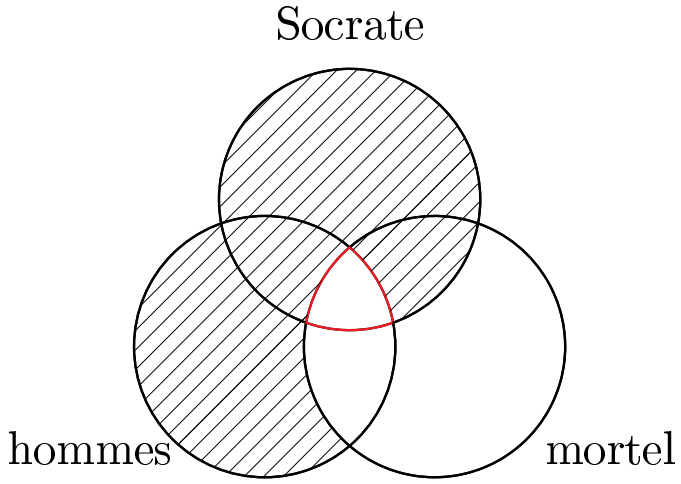




```
beginfig(2);
TilingUnit:=2cm;
TruchetTilingFill:=true;
TileColors((.4,.8,1),(.9,.9,.9));
TruchetTiling(5,3);
endfig;
```

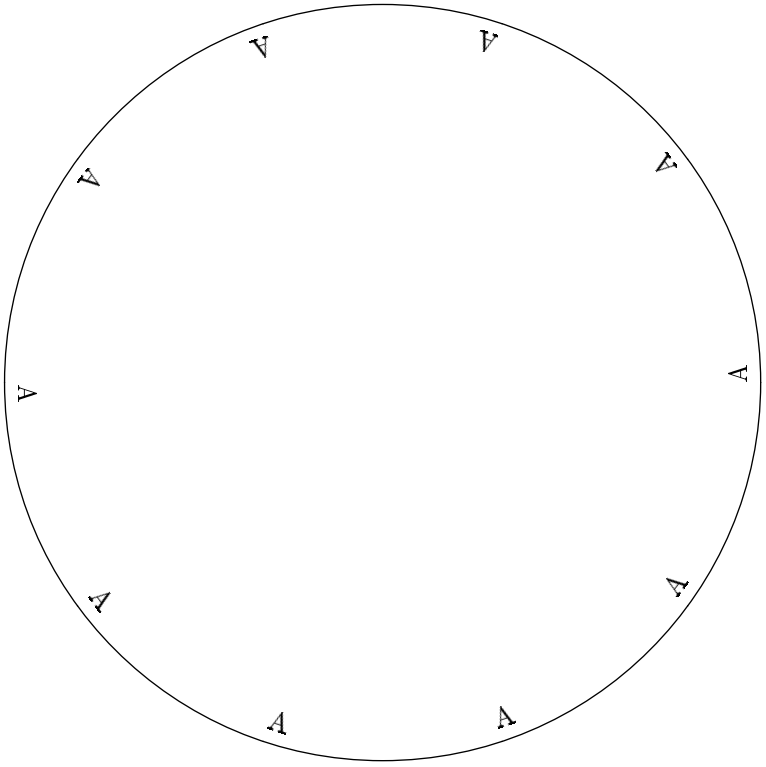
Voir sur CTAN dans `graphics/metapost`.

## Diagrammes de Venn




```
circle_labels("hommes","mortel","Socrate",1cm);

beginfig(1)
  numeric homme,mortel,Socrate;homme=1;mortel=2;Socrate=3;
  init;
  all[homme]is[mortel];
  all[Socrate]is[homme];
endfig;
```

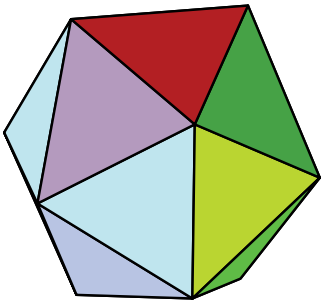
# Écriture circulaire

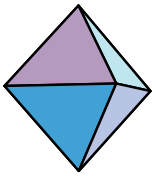


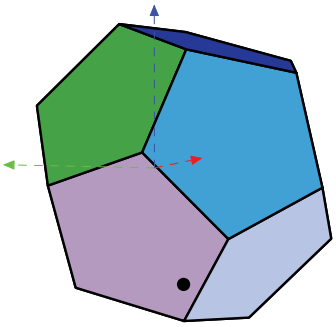
```
beginfig(2);
path p[];
pair dp;
transform t;
picture pp;
p5=(0,0){down}..(1,-1){right}..(2,0){up}..(1,1){left}..cycle;
p5:=p5 scaled 5cm;
draw p5;
n=length(p5);
for i=0 step n until 9n:
  dp:=direction i/10 of p5;
  drawdot(point (i/10) of p5);
  d:=angle (dp);
  pp:=btex A etex rotated d;
  t:=identity shifted (point (i/10) of p5) shifted (5*unitvector dir(d+90));
  draw pp transformed t;
endfor;
endfig;
```

Université Henri Poincaré

## Polyèdres







Voir sur CTAN dans `graphics/metapost`.

# Conclusions

## Avantages

- Langage puissant, agréable à utiliser
- Produit du PostScript
- Interfaçage avec TEX ou LATEX
- Une figure ne doit être compilée qu'une fois (inclusion rapide)

## Inconvénients

- Encore peu de bibliothèques
- Dessin dans un fichier séparé
- Pas de prise en compte du contexte TEX