The mkpath package

Here are the routines defined in the file **mkpath.inc**. Roughly speaking, the entire package is concerned with building paths in $2D$.

ROUTINE: **mkgraph**
ARGUMENTS: $a\ f\ x_0\ x_f\ N$
RETURNS: Adds a sequence of Bezier curves to the current path

The path added has N Bezier curves, along the graph of $y = f(x)$ from x_0 to x_f. The initial array contains parameters passed to the routine f. It is the *name* of the routine f which is passed as an argument (flagged by **/** in **PostScript**). The routine f has a fixed interface: it has two arguments, an array of parameters and a single number x, and returns an array $[f(x)\ f'(x)]$. The array can be empty, but it must be used in the call to **mkgraph**, and it must be removed from the stack in the routine f even if it is not used.

To draw the graph of $y = 2x^4$ from $x = -1$ to $x = 1$ we might have (among other things)

```
/quartic {
3 dict begin
/x exch def
/pars exch def
/c pars 0 def
[
c x mul x mul x mul x mul
2 c mul x mul x mul x mul
]
end
} def

newpath
[2] /quartic -1 1 8 mkgraph
stroke
```

The number N is to be chosen by experience.

ROUTINE: **mkpath**
ARGUMENTS: $a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of Bezier curves to the current path

Here the routine f amounts to a parametrization of a path. It has arguments **[...] t** as for **mkgraph**, and it is to return data of the form

$$[[x(t)\ y(t)][x'(t)\ y'(t)]]$$

This routine has a feature the graphing routine does not: it adds the path constructed to the current path by drawing a line from the current point if it exists to the start of the path being built. In this it acts like the command **arc** in **PostScript**.

```
/lissajous {
4 dict begin
/t exch 180 mul 3.1416 div def
/pars exch def
/m pars 0 get
/n pars 1 get
[
[t m mul cos t n mul cos]
[t m mul sin neg t n mul sin neg]
```

```
]
end
} def

newpath
[2 3] /lissajous 0 6.28 8 mkpath
stroke
```

ROUTINE: **mkpolypath**
ARGUMENTS: $a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of line segments to the current path

This has exactly the same calling setup as **mkpath**, but ignores the derivative in order to build a sequence of straight segments from one position to the next. Its purpose is to help you debug your parametrization routine, to separate the derivative calculation from the position calculation.