Metapost definitions

(also called Macros)

Taco Hoekwater

September 8, 2020

```
def ⟨symbolic token⟩ =
    ⟨replacement text⟩
enddef;
```

```
def -- = {curl 1}..{curl 1} enddef;
```

Argument types:

`expr` – expression values
`suffix` – variable names
`text` – arbitrary tokens

```
def mymac (expr a) =  ... enddef;
def mymac (suffix a) =  ... enddef;
def mymac (text a) =  ... enddef;
```

```
def mymac (expr ,) =
  , = 5
enddef;
```

```
def mymac (expr a1) =  % Error
  a1 = 5
enddef;
```

```
tracingall;
def mymac (expr a) =
  a = 5
enddef;

mymac(b);
```

```
mymac(EXPR0)->(EXPR0)=5
(EXPR0)<-b
{(b)=(5)}
## b=5
```

```
def mymac (expr a) =
  quote a = a
enddef;
```

```
def mymac (expr a) =
  b = a
enddef;

mymac(5);
```

Equivalent:

```
mymac(2+3);
```

```
mymac(5b);
```

Equivalent:

```
mymac((2+1)*b +2b);
```

```
def mymac (expr a) =
  a := 5 % Error
enddef;
```

```
def mymac (expr a) =
  a = 5
enddef;

mymac(5b);
```

```
def mymac(expr a) =
  a
enddef;

5b = mymac(5);
```

```
def mymac(suffix a) =
  a = 5
enddef;

mymac(b);
```

```
def mymac(suffix a) =
  pair a
enddef;

mymac(b);
```

```
def mymac(suffix a) =
  pair c.a
enddef;

mymac(b);
```

Becomes:

```
pair c.b;
```

```
def mymac(suffix a) =
  a := 5
enddef;

mymac(b);
```

```
def mymac (text a) =
    a = 5
enddef;

mymac(b);
```

```
def mymac (text a) =
    c = 5;
    a = c
enddef;

mymac(b);
```

```
def mymac (text a) =
    c = a c
enddef;

mymac(5; a =);
```

```
def mymac (text a) =
    a + 5
enddef;

mymac(b = (3 + 4));
```

`def` ⟨symbolic token⟩ ⟨delimited part⟩ `=`
  ⟨replacement text⟩
`enddef;`
⟨delimited part⟩ → ⟨empty⟩
  | ⟨delimited part⟩ `(` ⟨parameter type⟩ ⟨parameter tokens⟩ `)`
⟨parameter type⟩ → `expr` | `suffix` | `text`
⟨parameter tokens⟩ → ⟨symbolic token⟩
  | ⟨parameter tokens⟩ `,` ⟨symbolic token⟩

```
def mymac =
def mymac(expr a) =
def mymac(expr a,b) =
def mymac(expr a)(expr b) =
def mymac(suffix a)(expr b, c) =
def mymac(suffix a)
         (expr b, c)
         ( suffix d )
         (text e) =
```

```
mymac (a,b,c,d,e);
mymac (a)(b)(c)(d)(e);
mymac (a,b)(c,d,e);
```

```
mymac (a,b);          % bad 1
mymac a;              % bad 1
mymac (a)(b)()()(e);  % bad 2
mymac (a,b,,,e);      % bad 2
mymac (a,b)c(d)(e);   % bad 3
mymac (a,b)(c,d) e;   % bad 3
```

```
def helper =
  (d)(e)
enddef;

mymac (a,b)(c) helper;
```

```
def e =
  (f)
enddef;

mymac (a,b)(c,d) e;
```

```
def e =
  (f)
enddef;

mymac (a,b)(c,d)(e);
```

```
def e =
  (f
enddef;

mymac (a,b)(c,d) e);
```

```
def e =
  ,f
enddef;

mymac (a,b)(c,d e);
```

```
def c =
  f)
enddef;

mymac (a,b)(c(d,e);
```

```
def e =
  f)
enddef;

mymac (a,b)(c,d)(e; % Error
```

```
def mymac (text e,f) =
    show e; show f;
enddef;
```

```
mymac(g,h);
```

```
mymac (g)(h);
```

```
mymac (g; i; j; k; l)(h);
```

`def` ⟨symbolic token⟩ ⟨delimited part⟩ ⟨undelimited part⟩ `=`
⟨replacement text⟩
`enddef;`
⟨undelimited part⟩ → ⟨empty⟩
| ⟨parameter type⟩ ⟨parameter⟩
| ⟨precedence level⟩ ⟨parameter⟩
| `expr` ⟨parameter⟩ `of` ⟨parameter⟩
⟨precedence level⟩ → `primary` | `secondary` | `tertiary`

⟨equation⟩ → ⟨expression⟩ **=** ⟨right-hand side⟩
⟨assignment⟩ → ⟨variable⟩ **:=** ⟨right-hand side⟩
⟨right-hand side⟩ → ⟨expression⟩ | ⟨...⟩
⟨expression⟩ → ⟨numeric expression⟩ | ⟨...⟩
⟨numeric expression⟩ → ⟨numeric tertiary⟩
⟨numeric tertiary⟩ → ⟨numeric secondary⟩
| ⟨numeric tertiary⟩ **+** | **-** ⟨numeric secondary⟩ | ⟨...⟩
⟨numeric secondary⟩ → ⟨numeric primary⟩
| ⟨numeric secondary⟩ * | **/** ⟨numeric primary⟩
⟨numeric primary⟩ **=** ⟨numeric atom⟩
| ⟨numeric atom⟩ **[** ⟨numeric expression⟩ **,** ⟨numeric expression⟩ **]**
| ⟨...⟩
⟨numeric atom⟩ → ⟨numeric token⟩
| **(** ⟨numeric expression⟩ **)** | ⟨...⟩

```
4*(a+1) - b / 2[4,8]
```

Four ⟨primary⟩'s

```
4
```

```
(a+1)
```

```
b
```

```
2[4,8]
```

Two ⟨secondary⟩'s

```
4*(a+1)
```

```
b / 2[4,8]
```

One ⟨tertiary⟩ (also the ⟨expression⟩)

```
4*(a+1) - b / 2[4,8]
```

⟨string expression⟩ → ⟨string tertiary⟩
| ⟨string expression⟩ & ⟨string tertiary⟩
⟨string tertiary⟩ → ⟨string secondary⟩
⟨string secondary⟩ → ⟨string primary⟩
⟨string primary⟩ → ⟨string variable⟩
| `char` ⟨numeric primary⟩
| ⟨...⟩

```
def mymac primary arg =
 arg
enddef;
res = mymac 4*(a+1) - b / 2[4,8];
```

Argument is a

```
def mymac secondary arg =
 arg
enddef;
res = mymac 4*(a+1) - b / 2[4,8];
```

Argument is 4a+4

```
def mymac expr arg =
 arg
enddef;
res = mymac 4*(a+1) - b / 2[4,8];
```

Argument is `-0.08333b+4a+4`

```
def mymac tertiary arg =
 arg
enddef;
res = mymac 4*(a+1) - b / 2[4,8];
```

Argument is `-0.08333b+4a+4`

⟨macro definition⟩ → ⟨macro heading⟩ **=**
⟨replacement text⟩
`enddef`
⟨macro heading⟩ →
`primarydef` ⟨parameter⟩ ⟨symbolic token⟩ ⟨parameter⟩
| `secondarydef` ⟨parameter⟩ ⟨symbolic token⟩ ⟨parameter⟩
| `tertiarydef` ⟨parameter⟩ ⟨symbolic token⟩ ⟨parameter⟩

```
primarydef a mult b =
  a * b
enddef;
```

Need to find the right precedence

```
def stuff =
  fill unitsquare
enddef;

vardef stuff =
  fill unitsquare
enddef;
```

```
begingroup
  fill unitsquare
endgroup
```

⟨macro definition⟩ → ⟨macro heading⟩ **=**
⟨replacement text⟩
`enddef`
⟨macro heading⟩ →
`vardef` ⟨declared variable⟩ ⟨delimited part⟩ ⟨undelimited part⟩
| `vardef` ⟨declared variable⟩ **@#** ⟨delimited part⟩ ⟨undelimited part⟩

```
vardef mymac[]arr =
  4
enddef;
```

```
vardef z@#=
  (x@#,y@#)
enddef;
```

```
vardef z suffix v =
  (x.v,y.v)
enddef;
```

```
vardef mymac @# suffix v =
  (x@#v,y@#v)
enddef;
```

```
origin = mymac1right;
```

differs from

```
origin = mymac1(right);
```

```
vardef mymac @# (suffix v) =
  (x@#v,y@#v)
enddef;
```

Before (always present): #@

```
vardef p[]dir=
  (#@dx,#@dy)
enddef;
```

```
(p5dx,p5dy)
```

```
p5dir = up;
```

```
if p5dir = up: .... fi
```

At (always present): @

```
vardef a[] =
  if odd @: message("odd")
  else:     message("even")
  fi
enddef;
a1;  % prints "odd"
a20; % prints "even"
end.
```

After (only with @#): @#

```
vardef a@# =
  if str @# = "o": message("odd")
  else:            message("even")
  fi
enddef;
a.o; % prints "odd"
a.e; % prints "even"
```

```
pair p[]dir.target; % Error
```

⟨statement⟩ → ⟨equation⟩ | ⟨assignment⟩ | ⟨declaration⟩
| ⟨definition⟩ | ⟨title⟩ | ⟨command⟩ | ⟨empty⟩
| `begingroup` ⟨statement list⟩ ⟨statement⟩ `endgroup`

⟨numeric atom⟩ → ⟨numeric token⟩
| **(** ⟨numeric expression⟩ **)**
| `begingroup` ⟨statement list⟩ ⟨numeric expression⟩ `endgroup`
| ⟨...⟩

```
vardef stuff =
  fill unitsquare
enddef;
stuff withcolor green;
```

```
fill stuff withcolor green;
```

```
vardef stuff =
  unitsquare % earlier 'fill' deleted
enddef;
```

```
def hide(text t) =
   gobble begingroup t; endgroup
enddef;
def gobble primary g =
enddef;
```

```
def bfour =
  def b = 4 enddef
enddef;
```

```
def defbfour =
  def b = 4 % Error
enddef;
```

```
def defbfour =
  quote def b = 4
enddef;
```

```
defbfour *4 enddef;
```

⟨macro definition⟩ → ⟨macro heading⟩ **=**
⟨replacement text⟩
`enddef`
⟨macro heading⟩ →
`def` ⟨symbolic token⟩ ⟨delimited part⟩ ⟨undelimited part⟩
| `vardef` ⟨declared variable⟩ ⟨delimited part⟩ ⟨undelimited part⟩
| `vardef` ⟨declared variable⟩ **@#** ⟨delimited part⟩ ⟨undelimited part⟩
| ⟨binary def⟩ ⟨parameter⟩ ⟨symbolic token⟩ ⟨parameter⟩
⟨delimited part⟩ → ⟨empty⟩
| ⟨delimited part⟩ **(** ⟨parameter type⟩ ⟨parameter tokens⟩ **)**
⟨parameter type⟩ → `expr` | `suffix` | `text`
⟨parameter tokens⟩ → ⟨parameter⟩ | ⟨parameter tokens⟩ **,** ⟨parameter⟩

⟨parameter⟩ → ⟨symbolic token⟩

⟨undelimited part⟩ → ⟨empty⟩

| ⟨parameter type⟩ ⟨parameter⟩

| ⟨precedence level⟩ ⟨parameter⟩

| `expr` ⟨parameter⟩ `of` ⟨parameter⟩

⟨precedence level⟩ → `primary` | `secondary` | `tertiary`

⟨binary def⟩ → `primarydef` | `secondarydef` | `tertiarydef`