比較対照 METAPOST & Asymptote 〜 第 9 回

ちょっと前回の補足。**Label** 値を生成する Label 関数の引数でラベルの文字列を与える **string** 値は「必須」と書きましたが，文法的には省略可です。ただ，この場合ラベルの文字列は "" つまり空文字列になるため，前回のテーマからいうと意味がありません。

あと，**string** 値引数と **align** 値引数の間に **position** 値引数を入れた

Label(**string** 値，**position** 値，**align** 値，**pen** 値，**embed** 値，**filltype** 値)

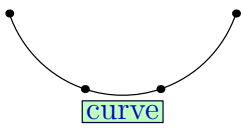
という形式も可能です。前回の使い方では，**position** 値引数は label 関数の **pair** 値引数で (型キャストの上) 上書きされるので，指定する意味がないのですが，今回紹介する使い方では意味があります (これについては以下を参照して下さい。)。

では今回のテーマです。今回はラベルに関する小ネタをいろいろ扱います。

1 パスにラベルを付ける

label 関数の「貼り付け先座標」を示す **pair** 値引数の部分を **path** 値で与えることができます。この場合，ラベルは引数で与えた ⟨ パス ⟩ のそばの適当な位置に配置されます。たとえば次のような感じです。

```
\begin{ASYpic}<1cm>(0,0)(3,-1.5)
  \sendASY{pair[] z = {(0,1),(1,0),(2,0),(3,1)};
    path p = z[0]..z[1]..z[2]..z[3]; draw(p); dot(p);
    label("curve",p,heavyblue,FillDraw(palegreen,darkblue));
  }
\end{ASYpic}
```

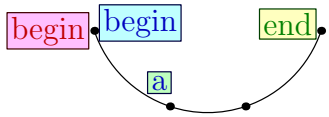


「適当な位置」とは書きましたが,「適当」かどうかは絵を描く人の主観によるので，実際には細かく位置を指定することが多いでしょう。以下，その点について補足します。

まず，曲線のどの辺にラベルを置くかについて。これは **Label** 値引数を (**string** 値の型キャストではなく) Label 関数で生成し，その際 **position** 値引数で位置を指定します。よく使うのは，曲線の始点を指定する BeginPoint , 中央を指定する MidPoint , 終点を指定する EndPoint です。上の例から分かるように，デフォルトは MidPoint です。さらに細かく指定したいときは，**position** 値を Relative 関数で生成します。これは曲線の始点を 0 , 終点を 1 とする相対座標 (**real** 値) を引数とし，曲線上の対応する点を **position** 値として返す関数で，たとえば先程の MidPoint は Relative(0.5) に相当します。

次に曲線のどちら側にラベルを置くかについて。これは前回の例と同じように label 関数の **align** 値引数で指定できるのですが，絶対座標で貼り付け先の西 = 左 (W) という指定方法の他に，曲線の接線方向を北 (N) と見た相対的な座標軸で西 (W) という指定方法も可能です。「相対的に西」の場合は，Relative(W) のように Relative 関数の引数に W を入れて指定します[1]。なお，上の例から分かるように，デフォルトは Relative(E) に相当するようです[2]。

以上を踏まえて次の例を見て下さい。




```
\begin{ASYpic}<1cm>(0,0)(3,1.2)
  \sendASY{pair[] z = {(0,1),(1,0),(2,0),(3,1)}; path p = z[0]..z[1]..z[2]..z[3];
    draw(p); dot(p);
    label(Label("begin",BeginPoint),p,Relative(W),heavyblue,FillDraw(palecyan,darkblue));
    label(Label("begin",BeginPoint),p,W,heavyred,FillDraw(palemagenta,darkred));
    label(Label("a",Relative(0.3)),p,Relative(W)),heavyblue,FillDraw(palegreen,darkblue));
    label(Label("end",EndPoint),p,Relative(W),deepgreen,FillDraw(paleyellow,darkgreen));
  }
\end{ASYpic}
```

[1] Relative 関数は先程同名の関数が出てきましたが，引数の種類 (こちらは **pair** 値) が違いますので，別の関数として共存しています。

[2] 本当のデフォルトは NoAlign なのですが，曲線上にラベルが来ることは通常望まれないので，内部で Relative(E) に読み替えているようです。NoAlign は「中央配置」ではなく「お任せ配置」と見るべきでしょうか。なお曲線上の「貼り付け先」にラベルの中央を合わせたいときは，Relative((0,0)) とします。

2　ラベルに矢印を付ける

ラベルの貼り付け先が混んでいる場合，矢印を引いてそこにラベルを配置することもあるでしょう。矢印つきのラベルは次の命令（関数）で描けます。

arrow(**picture** 値，**Label** 値，**pair** 値，**pair** 値，**real** 値，**align** 値，**pen** 値，**arrowbar** 値，**margin** 値)

ここで必須な引数は 2 つの **pair** 値とその次の **real** 値で，順に「矢印の指し示す点の座標」「矢印の向き[3]」「矢印の長さ」を表します。

picture 値引数は矢印とラベルを描き込む〈ピクチャー〉を指定します。省略すればいつもと同じく currentpicture です。

Label 値引数はもちろん貼り付けるラベルです。省略するとラベルは無くなり矢印のみになります。**string** 値の型キャストも可能です。

align 値は，label 関数のときと同じくラベルの配置される方向を指定しますが，arrow 関数ではデフォルトで「矢印の延長上」に配置されますので特に指定しなくてもいいでしょう。

pen 値は矢印の太さや色を指定します。ラベルやラベルの枠の色は **Label** 値引数を Label 関数で生成し，その引数として指定します[4]。

arrowbar 値，**margin** 値は今回初めて出てきましたが，これは話が長くなるので詳細は次回に送ります。**arrowbar** 値引数は矢印の形状を指定します。また，矢印は〈ペン〉の太さに応じて「指し示す先」や「ラベル」からちょっと離した方が見映えがいいことが多いのですが，その場合矢印の長さを縮める必要が出てきます。これを指定するのが **margin** 値引数です。今回これらはデフォルト値を使うことにして，特に指定しないことにします。

サンプルはこのページの最後，次項の例とまとめて挙げます。

3　ラベルに枠を付ける

長方形の枠は **filltype** 値引数で指定できますが，他にも楕円や角の丸い長方形を枠に持つラベルを描けます。ただこれは「ラベル」というより「文字列の入った図形」といった扱いのようで，**align** 値引数がありません。書式はこうです。

draw(**picture** 値，**Label** 値，**envelope** 値，**pair** 値，**real** 値，**real** 値，**pen** 値，**filltype** 値，**bool** 値)

picture 値，**Label** 値引数の意味は arrow 関数のときと同様です。

枠の形状を指定するのが **envelope** 値引数で，長方形（box），角の丸い長方形（roundbox），楕円（ellipse）があります。

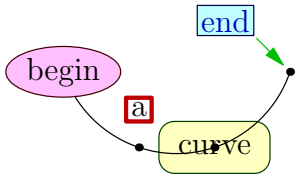
pair 値引数は貼り付け先の座標を指定します。ラベルの中央がこの座標に来るよう貼り付けられます。

2 つの **real** 値引数はラベルと枠の間隔を示し，出現順に「横方向の間隔」「縦方向の間隔」です。「横方向の間隔」は省略すると 0，「縦方向の間隔」は省略すると「横方向の間隔」と同じ値と解釈されます。また，**pen** 値引数は枠の色を，**filltype** 値で枠線を描く・枠内を塗りつぶすなどの指定ができます。なお，**filltype** 値で FillDraw などを指定する場合，FillDraw の引数で「間隔」「色（**pen** 値）」を指定できますが，「間隔」は先程の draw の引数の「間隔」に累積され，「色」は draw の引数の「色」を上書きで無視するようです。

最後の **bool** 値引数は，この枠つきのラベルをこれまでに描いた図の上に描くか否かを指定します。つまり true ならこれまでの図の上に描き，false なら下に描きます。

以上を踏まえたサンプルは次のようになります。

```
\begin{ASYpic}<1cm>(0,0)(3,-2.5)
  \sendASY{pair[] z = {(0,1),(1,0),(2,0),(3,1)};
    path p = z[0]..z[1]..z[2]..z[3]; draw(p); dot(p);
    draw("begin",ellipse,z[0],FillDraw(palemagenta,darkred));
    draw("a",box,z[1]+0.5N,heavyred+1.5pt);
    draw("curve",roundbox,z[2],2mm,2mm,FillDraw(paleyellow,darkgreen),false);
    arrow(Label("end",heavyblue,FillDraw(palecyan,darkblue)),z[3],dir(135),5mm,heavygreen);}
\end{ASYpic}
```



[3]矢印の向きと書きましたが，実際には「矢印の指し示す点からラベルのある方角」への向きですから，矢印の向きのちょうど逆方向を指定することになります。

[4]label 関数のときは，Label 関数の引数にラベルの色などを指定しても label 関数の引数で上書きされましたが，arrow 関数の **pen** 値引数は矢印の色などを指定するものでラベルとは無関係ですから，上書きは起こりません。