比較対照 METAPOST & Asymptote 〜 第 10 回

今回は第一部最後ということで，これまで保留にしてきた draw のオプション引数について説明しましょう。

1 draw の書式

第 4 回で draw や fill の書式を次のようにまとめました。

fill(**pair** 値，**picture** 値，**path** 値 (必須)，**pen** 値)　　(draw も同様。)

fill に関してはこれですべて[1]なのですが，draw にはこれ以外にも引数があります。

まずは draw の (基本の) 定義における引数をすべて書き上げてみます。

draw(**pair** 値，**picture** 値，**Label** 値，**path** 値 (必須)，**align** 値，**pen** 値，
arrowbar 値，**arrowbar** 値，**margin** 値，**Label** 値，**marker** 値)

Label 値と **arrowbar** 値が 2 つずつあります。2 つの **Label** 値はそれぞれ役割が異なるのですが，必須の **path** 値引数の前か後かで区別がつきます。2 つの **arrowbar** 値は役割が同じです。名前から想像できると思いますが，矢印やバーを付けることを指定する引数です。2 つあるのは矢印 (->) とバー (-|) を組み合わせて ->| のような記号を描かせるためです。

引数のうち，**pair** 値，**picture** 値，**path** 値，**pen** 値については，すでに第 4 回 (まで) で説明ずみです。

また，**align** 値と (**path** 値の前にある方の) **Label** 値に関しては，前回の「label 関数で 〈 パス 〉 にラベルを付ける」のと同様のことをするための引数です。したがって，これに関しても第 7 〜 9 回で説明済みです。

arrowbar 値，**margin** 値については前回すでに現れていますが，紙幅の都合でまだ具体的な説明をしていませんでした。今回これを詳しく説明します。

残り 2 つの引数のうち，**marker** 値は 〈 パス 〉 を構成する点 (〈 ペア 〉) などに記号をつける場合，その記号 (マーカー) を指定するのに使われます。また，**Label** 値 (**path** 値の後にある方) は凡例 (legend) を指定するのに使われます。「凡例」は複数のグラフを同時に描くとき，どのグラフが何に対応するのかを表す一覧表です。ここに指定されたラベルと，対応する 〈 パス 〉 を描くときに使った **pen** や **marker** は別個に記録され，legend() 命令によって凡例を表す適切な **frame** 値に変換されます (これを add で貼り付けます)。マーカーや凡例の使い方は，グラフの描き方のときにまた詳しく説明することにし，今回は簡単な実例だけ挙げます。

2 arrowbar 値

arrowbar 値としてとれる値は，デフォルトでは以下の通りになっています[2]。

矢印型 (その 1) : BeginArrow , EndArrow , Arrow , BeginArcArrow , EndArcArrow , ArcArrow
矢印型 (その 2) : MidArrow , Arrows , MidArcArrow , ArcArrows
バー型　　: BeginBar , EndBar , Bar , Bars
特殊型　　: None (デフォルト) , Blank

名前からおおよそ想像はつくと思いますが，Begin〜 は 〈 パス 〉 の始点に，Mid〜 は中点に，End〜 は終点に，〜s は両端に，それぞれ記号を付加します。矢印型で真ん中に Arc が入ったものは，より曲線向けにデフォルト値を調整[3]したものです。

これらは **arrowbar** 値の定数としてそのまま使えますが，関数として引数をとることもできます。

矢印型 (その 1) の引数は (**arrowhead** 値，**real** 値，**real** 値，**filltype** 値，**position** 値) です。

arrowhead 値引数は矢印の形状を指定します。DefaultHead , SimpleHead , HookHead , TeXHead から選べます。

1 つめの **real** 値引数は矢印のサイズを実寸で，2 つめの **real** 値引数は矢印の三角形の頂角の半分を度数で与えます。なお，1 つめの **real** 値引数を省略し，2 つめの **real** 値のみ指定したいときは angle=45 などとします。一般に Asymptote の関数の引数にはそれぞれ名前がついており，今回のようなケースでは名前と指定したい値を = で結ぶことで適切な引数に値を渡せます[4]。

filltype 値引数は矢印を塗りつぶすか枠線だけにするかなどを指定します。とれる値は以前説明したとおりです。

---

[1] 第 6 回で説明したように，Asymptote では同名で引数の違う関数が同居できますので，この形式以外の fill もあり得ますし，実際あります。そういったものをユーザーが勝手に定義することも可能です。そういう意味では「すべて」の引数を説明するのは不可能なのですが，ここでは「基本の定義における引数のすべて」ぐらいの意味にとって下さい。

[2] EndArrow と Arrow は同じものです。同様に EndArcArrow と ArcArrow , EndBar と Bar もそれぞれ同じものです。

[3] たとえば矢印の角度が大きくなっています。

[4] それどころか，ある程度なら引数の順序もこの方法で変更できます。

最後の **position** 値引数についてですが，実は BeginArrow は「⟨ パス ⟩ の始点」がデフォルトなだけであって，**position** 値引数を変更すれば ⟨ パス ⟩ 上の任意の位置に矢印を置けます。 EndArrow についても同様です。

矢印型 (その 2) の引数は **position** 値が無いことを除いて arrow 型 (その 1) と同じです[5]。

バー型の引数は (**real** 値) でバーのサイズを実寸で与えます。

特殊型は引数をとりません。なお，None は矢印やバーを付けないだけですが，Blank は ⟨ パス ⟩ そのものも描きません[6]。

3 margin 値

前回少し触れたように，⟨ パス ⟩ に矢印を付けるときは，矢印の指し示す点から少し離した方が見映えがよくなることがあります。また，太い線を引いたときペンの半径の分だけ線がはみ出すのが目立ちます。これもそのはみ出しを消す方が見映えがよいときがあります。こういった場合に ⟨ パス ⟩ をどれだけ縮めるか指定するのが **margin** 値引数です[7]。

デフォルトは NoMargin で縮めません。実寸で指定するときは TrueMargin(**real** 値，**real** 値) です。引数は順に始点，終点における縮み幅です。その他の値は，目的に応じて 3 系統に分類されます。

- フォントサイズに依存するタイプ — ラベルに付ける矢印などに向いています。

  BeginMargin , EndMargin , Margin , Margins , Margin(**real** 値，**real** 値)

現在の ⟨ ペン ⟩ のフォントサイズの labelmargin 倍 (デフォルトは 0.3 倍) を単位として，たとえば BeginMargin なら始点側を 1 単位縮めます。EndMargin[8] なら終点側，Margins なら両端を縮めます。始点側を 2 単位，終点側を 0.5 単位縮めたいのであれば Margin(2,0.5) とします。

- ⟨ ペン ⟩ の太さ (直径) に依存するタイプ (その 1) — ペンの半径分のはみ出しを抑えるときなどに便利です。

  BeginPenMargin , EndPenMargin , PenMargin , PenMargins , PenMargin(**real** 値，**real** 値)

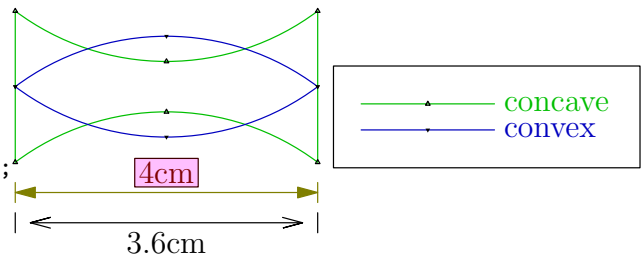
現在の ⟨ ペン ⟩ の太さを単位とする以外は上の系統と同様です。ただし，PenMargin() の引数に指定する値が 0 のときちょうどはみ出さないように値が 0.5 だけ補正されていますので,「マージン無し」は -0.5 です[9]

- ⟨ ペン ⟩ の太さ (直径) に依存するタイプ (その 2)

  BeginDotMargin , EndDotMargin , DotMargin , DotMargins , DotMargin(**real** 値，**real** 値)

現在の ⟨ ペン ⟩ の太さの dotfactor 倍 (デフォルトは 6 倍) を単位とする以外は上の系統と同様です。dot で ⟨ パス ⟩ の両端にドットを付けたとき，そのドットとかぶらないようにするのに役立ちます。

```
\begin{ASYpic}<2cm>(0,0)(4,-1)
  \sendASY{pair[] z = {(0,0),(1,1/3),(2,0)};
    path p=z[0]..z[1]..z[2];
    path[] pth={p--(shift(N)*yscale(-1)*reverse(p))--cycle,
                shift(0.5N)*p--(shift(0.5N)*yscale(-1)*reverse(p))--cycle,
                shift(0.2S)*(z[0]--z[2]), shift(0.4S)*(z[0]--z[2])};
    Label L=Label("4cm",MidPoint,deepred,FillDraw(palemagenta,darkred));
    Label[] Leg={Label("concave",heavygreen),Label("convex",heavyblue)};
    draw(pth[0],heavygreen,Leg[0],Mark[1]);
    draw(pth[1],heavyblue,Leg[1],MarkFill[4]);
    draw(L,pth[2],Relative(W),olive,Arrows,Bars);
    draw("3.6cm",pth[3],Relative(E),
         Bars,Arrows(SimpleHead),TrueMargin(2mm,2mm));
    add(legend(),point(E),2mm*E);
  }
\end{ASYpic}
```



今回で一応一区切りです。次回からは応用ってことで，グラフを描いたり文法の詳細を説明したりと，もう少し高度な内容にしていく予定です。

[5]つまりこちらは位置が固定です。
[6]ただし，arrow 型と併用すると (arrow 型の仕様上) ⟨ パス ⟩ の描画が行われてしまうため，Blank の意味が無くなってしまいます。
[7]なお，**margin** 値引数を指定してもバーの位置は変化しません。
[8]Margin は EndMargin と同じです。以下の系統でも同様です。
[9]したがってたとえば BeginPenMargin=PenMargin(0.5,-0.5) です。— 個人的には PenMargin(0,-0.5) の方がいいような気がするんですが。。。