Flow — LaTeX の picture 環境でフローチャートを描くプログラム

テリー・ブラウン

1992 年 11 月 25 日 – 2004 年 2 月 18 日

目 次

1　概要

Flow は、このドキュメントで説明されているフローチャート記述言語を解析し LaTeX の picture 環境へ翻訳する小さなプログラムです。

Flow は、次のように、フィルターとして機能します。

```
flow < myprog.flo > myprog.pic
```

ここで、myprog.flo はフローチャート記述言語を含むテキスト・ファイル、myprog.pic はフローチャートを描く LaTeX の picture 環境本体のテキスト・ファイルです。次のようにも書けます。

```
flow inputFile または flow inputFile outputFile
```

VMS ユーザーは後ろの方の「注意事項」をご覧ください。

myprog.pic は、テキスト・エディタで直接 LaTeX ファイルに取り込むか、LaTeX ファイルでは \input コマンドで参照しておき、LaTeX 起動時に動的に呼び込みます。

2　動作条件

特別な用意は不要です。Flow は標準的な C で書かれており、書き直すことなく、ほとんどの環境で動作するはずです。もし、うまくいかないときは、ソース・ファイルの始めの方にいくつかあるマクロ定義 (#define) をチェックしてみてください。

3　配布とバグ報告

Flow は GPL でカバーされるフリーソフトウェアです。詳細は、付属のファイル COPYING をご覧ください。バクなどはメールにてお知らせください。

4　フローチャート記述言語の文法

4.1　[indented text]

[indented text] は、0 行以上のテキストを受け付けるコマンドを必要とします。そして、これら行は字下げによって識別されます。あるコマンドに続く、一つの空白またはタブで始まる行は、すべて、そのコマンドのテキストと解釈されます。このような始まり方をしない最初の行があれば、次のコマンドと解釈されます。配布されているコード/実行形式は大文字小文字を区別していませんが、個々のコンパイルでは様々でしょう。

フローチャートは、常に、上下左右のいずれかの方向へ「進む」ものとします。最初の方向は下です。

フローチャート記述言語は、文法ミス、空行、または、ファイルの終わりによって終了します。キーワードの大文字小文字は本ドキュメントどおりにしてください。

flow の出力は LaTeX の picture 環境の内部に見えることでしょう。図の位置を調整するには、通常どおり、picture コマンドで二番目に指定する左下隅の座標を書き換えてください。

\unitlength は picture 環境用に指定してください。本ドキュメントの例では、すべて 2em を指定しています (つまり、この LaTeX ファイルの頭の方に \setlength{\unitlength}{2em} と書いてあります)。この値を小さくすればテキストを囲むボックスは狭くなり、大きくすれば広くなります。

4.2 コマンド

4.2.1 box コマンド

書き方

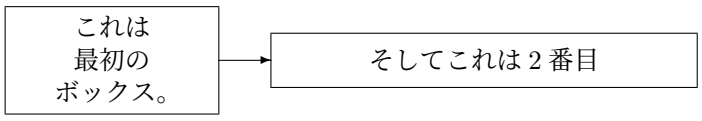
```
Box [x y]
  [indented text]
```

機能 現在位置に長方形の枠で囲まれたテキストを描きます。また、その時点での指定に従って、その枠へとつながる線または矢印も描きます。x、y が指定された場合、その単位は \unitlength で、そのボックスおよび以降のすべてのボックスの大きさに適用されます。省略すると、横 4, 縦 2 \unitlength。

入力例

```
Right
Box
    これは
    最初の
    ボックス。
Box 8 1
    そしてこれは 2 番目
```

出力例



4.2.2 Oval, Tilt, Text, Call, Drum コマンド

書き方

```
Oval [x y]
  [indented text]

Tilt [x y]
  [indented text]
```

```
Call [x y]
  [indented text]

Drum [x y]
  [indented text]

Text [x y]
  [indented text]
```

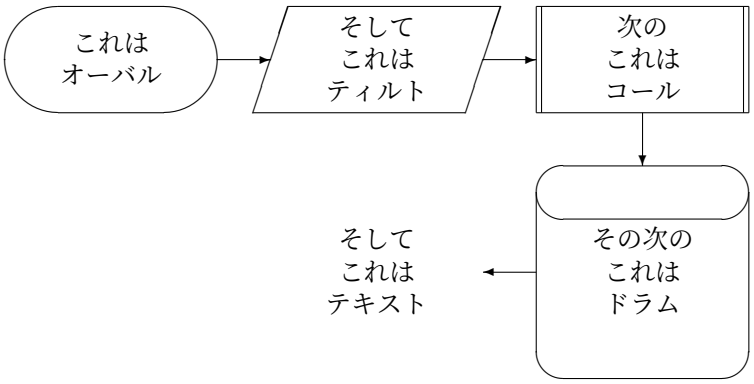
機能 Oval、Tilt, Call, Drum および Text は、Box と同じ機能ですが、テキストを囲む枠の形状が異なります。Oval は長円、Tilt は平行四辺形、Call は横枠が二重線、Drum は円筒形[1]、そして Text は枠なしです。

入力例

```
Right
Oval
    これは
    オーバル
Tilt
    そして
    これは
    ティルト
Call
    次の
    これは
    コール
Down
Drum
    その次の
    これは
    ドラム
Left
Text
    そして
    これは
    テキスト
```

出力例

[1] [s.k] Call と Drum は+k-0.01 からの拡張です。



4.2.3 Choice コマンド

書き方

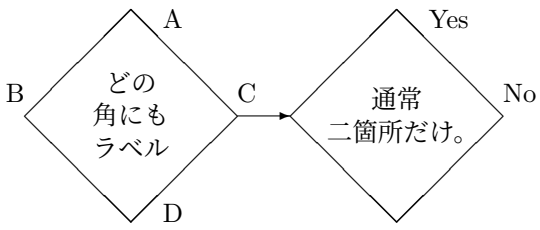
```
Choice A B C D [x y]
  [indented text]
```

機能 選択を意味する菱形を描きます。出力例の左側のように角ごとにラベルを付けられます。終止符 (.) は印字されません。オプションの x と y は菱形の大きさです。省略すると、ともに 4 \unitlengh。縦横比が LaTeX がサポートする直線の傾き (-6〜6：-6〜6, 整数のみ) と合わないと Flow はエラー・メッセージを出します。

入力例

```
Right
Choice A B C D
    どの
    角にも
    ラベル
Choice Yes . No .
    通常
    二箇所だけ。
```

出力例



4.2.4 SetTrack, TxtPos, SetWidth コマンド

書き方

```
SetTrack none | arrow | line
```

機能 ボックス間を線でつなぐかつながないか、つなぐならば矢印ありかなしか、を指定。

書き方

```
SetWidth thick | thin | #
```

機能 SetWidth [2]ボックス間の線を引く場合の線の太さを指定。標準は、0.8 となっており、thick と指定すると、この標準の 0.8 になる。thin と指定した場合は、0.4 になり、少し細くなる。直接、数値を指定する事も可能だが、余りに太いと、矢印に見えなくなってしまう。

thick でも thin でもなく、また数値と解釈できない (atof の仕様による) か、あるいは 0.0 以下の太さが指定された場合は、強制的に、標準値 (0.8) に戻してしまう。

書き方

```
TxtPos P1 P2 [B [A]]
```

機能 P1 も P2 も LaTeX の位置仕様 (例えば [c] や [l])。P1 は行ごとのテキストの位置仕様、P2 はボックス内のテキスト全体の位置仕様。B は各テキスト行の前 (before) に置く文字列 (空白は指定できない)、A は各テキスト行の後 (after) に置く文字列。次の例の二番目では B を指定してテキストをボックスの左側の辺から少し離しています。

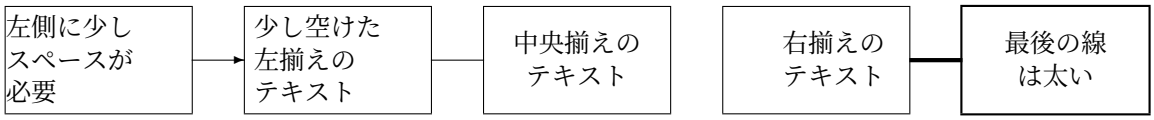
入力例

```
Right
SetTrack arrow
TxtPos [l] [l]
Box 3.5 2
    左側に少し
    スペースが
    必要
TxtPos [l] [l] ~
Box
    少し空けた
    左揃えの
    テキスト
SetTrack line
TxtPos [c] [c]
```

[2][sk] SetWidth も+k-0.01 から追加。

```
Box
    中央揃えの
    テキスト
SetTrack none
TxtPos [r] [r] ~ \hspace*{1zh}
Box
    右揃えの
    テキスト
SetTrack line
TxtPos [c] [c]
SetWidth 1.6
Box
最後の線
は太い
```

出力例



4.2.5 Tag, ToTag コマンド

書き方

```
Tag

ToTag
```

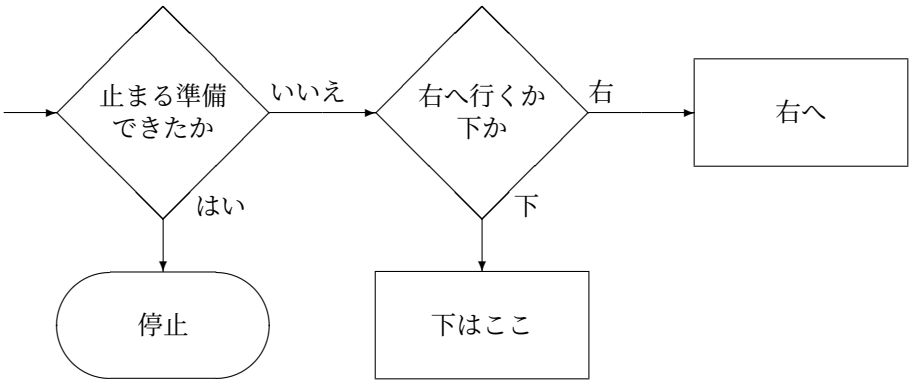
機能 Tag は最後に描いたオブジェクトの位置と大きさをあるスタックに格納し、ToTag でその位置に戻り、アイテムをそのスタックから削除 (ポップアップ) します。スタックするアイテムは何でもかまいませんが、特に、菱形の Choice から二番目の分岐を出すときに便利です。もし、Tag より多くの ToTag が指定されていると Flow は警告しますが、終了時、Tag がスタックに残っていても無視します。

入力例

```
Right 0
Choice . . いいえ はい
    止まる準備
    できたか
Tag
Right 1
Choice . . 右 下
```

```
    右へ行くか
    下か
Tag
Right 1
Box
    右へ
ToTag
Down
Box
    下はここ
ToTag
Down
Oval
    停止
```

出力例



4.2.6 Up, Down, Left, Right コマンド

書き方

```
Up    [d [*]]
Down  [d [*]]
Left  [d [*]]
Right [d [*]]
```

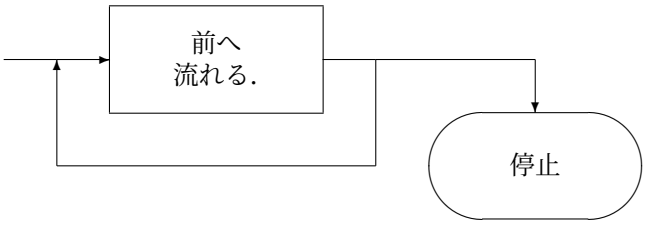
機能 これらのコマンドは、オプション・パラメータの有無にかかわらず、現時点でのフローチャートの向きを変えます。オプション・パラメータを指定すると、SetTrack が line **または**arrow のときは線が引かれ、none のときは空きができます。線または空きの長さは d。省略すると Box、Oval、Tilt, Call, Drum および Text は横 4、縦 2 単位長。Choice は縦横とも 4 単位長。オブジェクトをつなぐ矢印は 1 単位長。よって、現在の方向が下向きの場合、ボックスを一つ描くには $2 + 1 = 3$ 単位長の高さが必要です。

オブジェクトを指定しないで直線を矢印で終わらせるには、“d” の次に空白を一つ置いて “*” を指定します。

入力例

```
Right 1
Box
  前へ
  流れる.
Right 1
Tag
Down 2
Left 6
Up 2 *
ToTag
Right 3
Down
Oval
  停止
```

出力例



4.2.7 Scale コマンド

書き方

```
Scale x y
```

機能 次のアイテムの縦横比を指定値に変更する。

入力例

```
Right
Oval
    普通の
    オーバル
Scale 2 1
Box
    STOP 用には小さいオーバル
Scale 0.5 0.5
```

```
Oval
    STOP
```

出力例

4.2.8 Skip コマンド

書き方

```
Skip x0 y0 x1 x1
```

機能

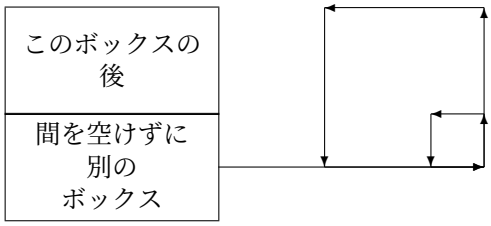
x0	横方向のボックス間の空き、省略時は 1 \unitlength
y0	縦方向のボックス間の空き、省略時は 1 \unitlength
x1	Left および Right コマンド用の倍率、省略時は 1
y1	Up および Down コマンド用の倍率、省略時は 1

入力例

```
Skip 0 0 1 1
SetTrack none
Box
    このボックスの
    後
Box
    間を空けずに
    別の
    ボックス
SetTrack arrow
Right 5
Up 1 *
Left 1 *
Down 1 *
Right 1 *
Skip 0 0 3 3
Up 1 *
Left 1 *
```

```
Down 1 *
Right 1 *
```

出力例



5　VMS ユーザー向けの注意事項

VMS には詳しくないのですが、以下は有効な方法の一つです。普通に、コンパイル、リンクした後、まず

```
flow :== $$1$DIA3:[brownt1.usr.flow]flow.exe
```

と入力します。ここで、ボックス内の記号は作業しているドライブの名前、[brownt1.usr.flow] は適当なパスです。次に

```
flow infile outfile
```

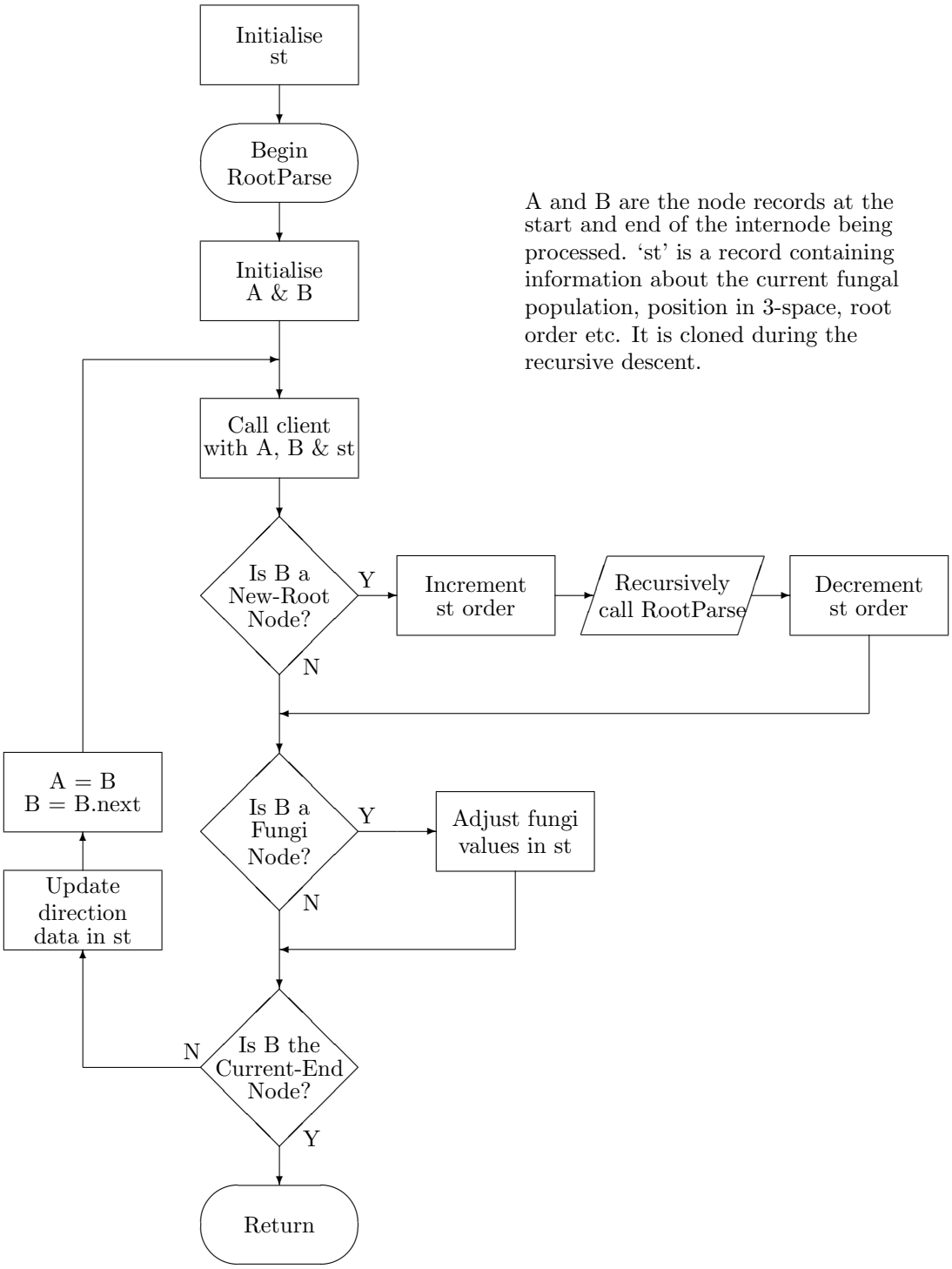
と入力します。リダイレクトは機能しないようです。

6 例

このフローチャートを描くコマンドは、コメント行として flowdoc.tex に入っています。フローチャートの説明文も picture 環境の一部であることにご注意ください。(Text コマンドを使っています)

```
% THIS IS THE FLOW DATA FOR THE EXAMPLE AT THE END
Box
  Initialise
  st
Oval
  Begin
  RootParse
Tag
Box
  Initialise
  A \& B
Down 1
Box
  Call client
  with A, B \& st
Choice . . Y N
  Is B a
  New-Root
  Node?
Tag
Down 1
Choice . . Y N
  Is B a
  Fungi
  Node?
Tag
Down 1
Choice . N . Y
  Is B the
  Current-End
  Node?
Tag
Oval
  Return
ToTag
Left 3
Up 2
Box
```

```
  Update
  direction
  data in st
Box
  A = B
  B = B.next
Up 10
Right 5 *
ToTag
Right 1
Box
  Adjust fungi
  values in st
Down 2
Left 6 *
ToTag
Right
Box
  Increment
  st order
Tilt
  Recursively
  call RootParse
Box
  Decrement
  st order
Down 2
Left 15 *
ToTag
SetTrack none
Down 2
Right 8
TxtPos [l] [c]
Text
  A and B are the node records at the
  start and end of the internode being
  processed.  'st' is a record containing
  information about the current fungal
  population, position in 3-space, root
  order etc.  It is cloned during the
  recursive descent.
```



A and B are the node records at the start and end of the internode being processed. 'st' is a record containing information about the current fungal population, position in 3-space, root order etc. It is cloned during the recursive descent.