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

(Version 0.99g+k-0.06 [2012/05/28])

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[1]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 ファイル内で \input コマンドで参照しておき、LaTeX 起動時に動的に呼び込むようにします。

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 (kurino@zaregoto.org)[2]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] の表記は、コマンドが 0 行以上のテキストを受け付ける事を示しています。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。

[1] sk:「かった」かもしれません。

[2] オリジナルの作者は、テリー・ブラウン氏ですが、+k-0.01 以後の版に関しては、栗野までご連絡ください。また、github{\slash}kurino{\slash}flow[https://github.com/kurino/flow] も一緒に御参照下さい。

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

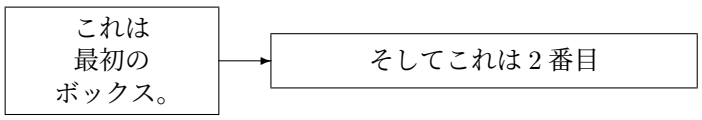
```
Box [x y]
  [indented text]
```

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。初期値は、横 4, 縦 2 \unitlength です。

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]
```

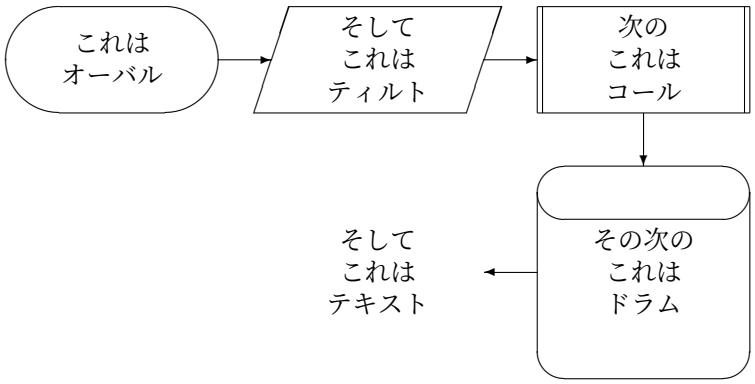
機能 Oval、Tilt, Call, Drum および Text は、Box と同じ機能ですが、テキストを囲む枠の形状が異なります。Oval は長円、Tilt は平行四辺形、Call は横枠が二重線、Drum は円筒形[3]、そして Text は枠なしです。

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
```

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

出力例



4.2.3 Choice コマンド

書き方

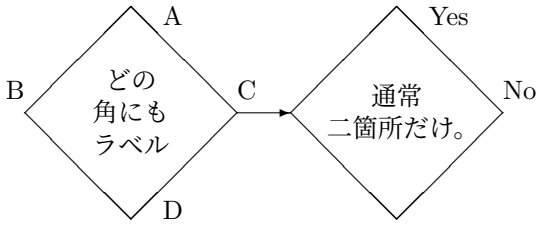
```
Choice A B C D [x y]
  [indented text]
```

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。初期値は、ともに 4 \unitlengh です。縦横比が LaTeX がサポートする直線の傾き (-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、を指定します。

書き方

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

機能 SetWidth [4]ボックス間の線を引く場合の線の太さを指定します。初期値は、0.8 em となっており、thick と指定すると、この標準の 0.8 em になります。thin と指定した場合は、0.4 em になり、少し細くなります。直接、数値を指定する事も可能ですが、余りに太いと、矢印に見えなくなってしまうようです。

thick でも thin でもなく、また数値と解釈できない[5]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 em) に戻してしまいます。

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]) と同じです。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となります。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) に置く文字列となります。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。

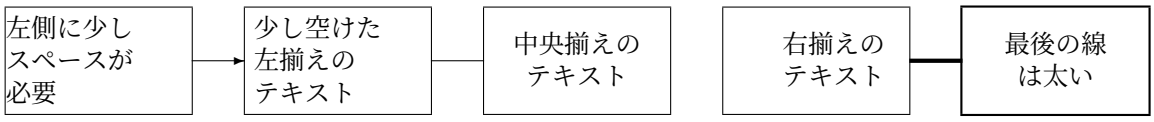
入力例

```
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
```

[4] [sk] SetWidth も+k-0.01 から追加。
[5] atof の仕様による

```
  テキスト
SetTrack line
TxtPos [c] [c]
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
```

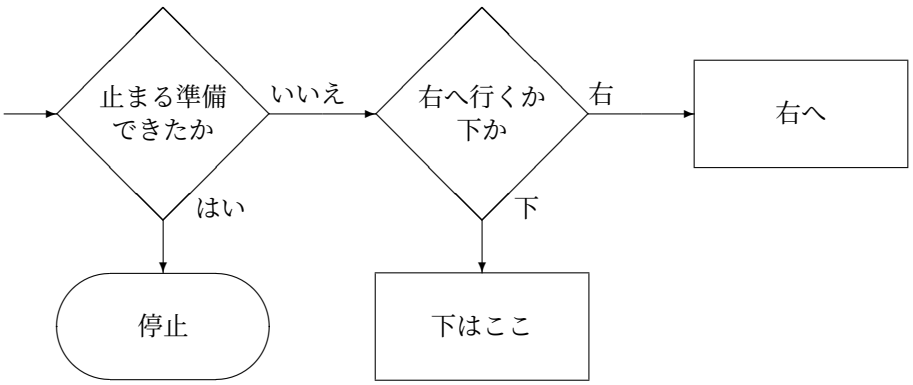
機能 Tag は最後に描いたオブジェクトの位置と大きさをあるスタックに格納し、ToTag で、その位置に戻り、アイテムをそのスタックから削除 (ポップアップ) します。スタックするアイテムは何でもかまいませんが、特に、菱形の Choice から二番目の分岐を出すときに便利です。もし、Tag より多くの ToTag が指定されていると Flow は警告しますが、終了時、Tag がスタックに残っていても無視します。

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 の時には線が引かれ、none の時には空きができます。線または空きの長さは d で指定します。初期値は、Box、Oval、Tilt, Call, Drum および Text が横 4、縦 2 \unitlength です。Choice

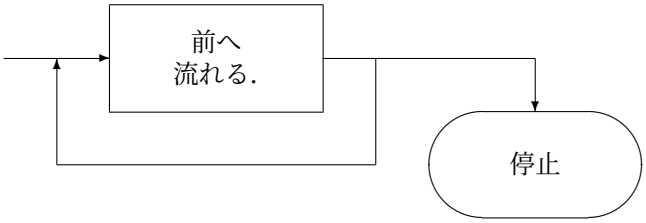
は縦横とも 4 \unitlength です。オブジェクトをつなぐ矢印は 1 \unitlength です。よって、現在の方向が下向きの場合、ボックスを一つ描くには 2 + 1 = 3 \unitlength の高さが必要です。

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 Turn コマンド

書き方

```
Turn [ (right|left) [d [*]] ]
```

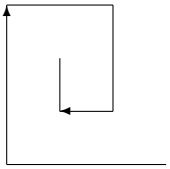
機能 Turn [6]は Up, Down, Left, Right コマンドの相対版であり、Up などが、現在の向きとは無関係に、次の向きを絶対的に変更するのに対し、Turn は現在の向きに相対的に変更します。

入力例

[6][sk] Turn は k-0.05 から利用できます。

```
Turn right 3
Turn right 3 *
Turn right 2
Turn right 2
Turn right 1 *
Turn right 1
```

出力例



4.2.8 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

4.2.9 Skip コマンド

書き方

```
Skip x0 y0 x1 x1
```

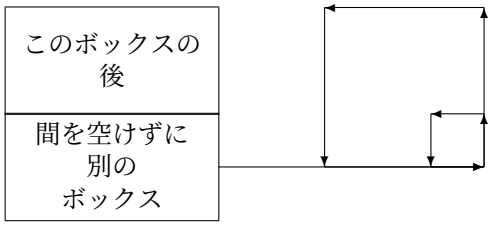
機能

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

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



4.2.10 Figure, Draw コマンド

書き方

```
Figure name [ size:(x,y) [ hasText [ params ] ] ]
  draw commands
defaults:
      size : 4 2
      hasText : True
      params : 0
```

機能 利用者が定義する新しい図形 (Figure) を追加します[7]。図形の形の指定は、Figure 命令の後に続くインデントされた命令列で指定します。Figure で定義された新しい図形は、Draw 命令 (この Draw 命令の”Draw”は省略可能なので、省略すると、あたかも新しい命令が追加され、利用できるようにみえます。) を利用して、描画する事ができます。

定義中で、引数を参照する場合は「%# (#=1〜9)」を指定する事により、「#番目の引数」を参照する事ができます。

書き方

```
[Draw] name [ params .. ]
  [indented text]
```

機能 Figure で新たに定義された図形を描画します。HasText が真の時には、更に、[indented text] を引数として扱います。

入力例

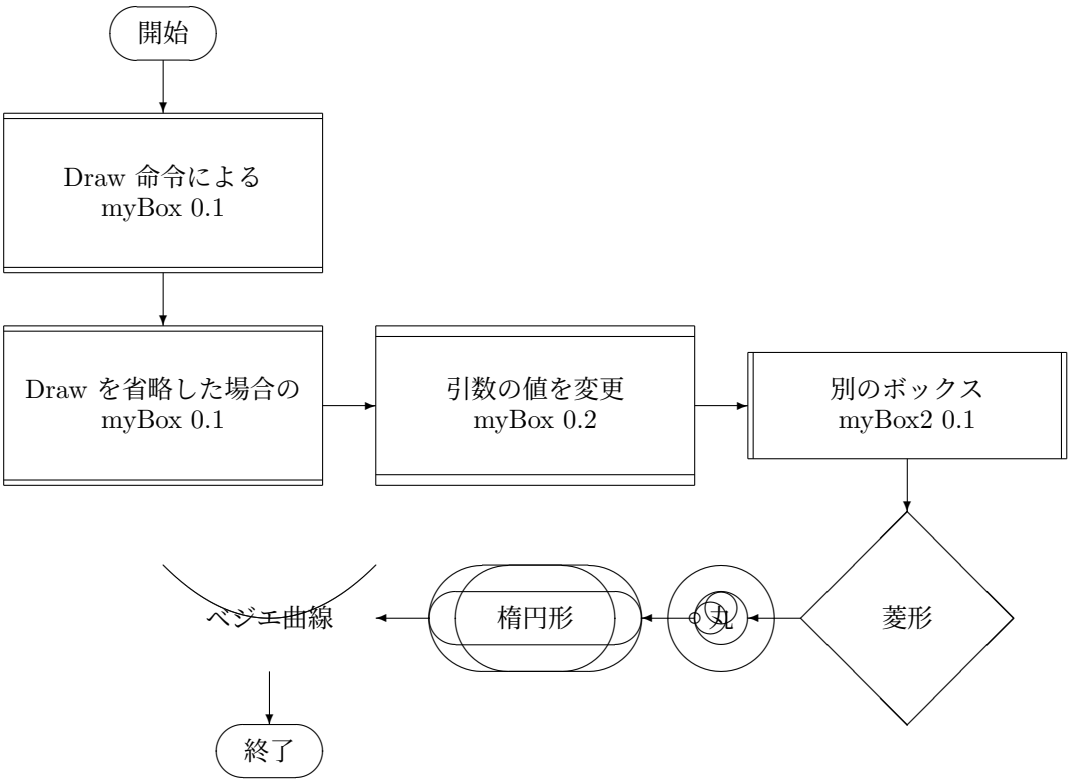
```
Figure  myBox  6  3  True  1
  line  s  s  s  e
  line  s  s  e  s
  line  s   %1  e   %1
  line  s  e  e  e
  line  e  s  e  e
  line  s  -%1  e  -%1
Figure  myBox2  6  2  True  1
  line  s  s  s  e
  line   %1  s   %1  e
  line  s  s  e  s
  line  s  e  e  e
  line  e  s  e  e
  line  -%1  s  -%1  e
Figure  myChoice  4  4  True
```

[7] [sk] Figure/Draw は+k-0.06 より追加。

```
  line  m  t  r  m
  line  r  m  m  b
  line  m  b  l  c
  line  l  c  m  t
Figure  myCircle  1  1  True
  circle
  circle -0.2 -0.2 +0.0
  circle -0.2 +0.0 -0.2
  circle +0.1 -0.5 +0.0
  circle 2
Figure  myOval  4  2  True
  oval
  oval 1.5
  oval . 0.5
Figure  myBezier  4  2  True
  bezier
Scale 0.5 0.5
Oval
  開始
Down
Draw myBox 0.1
  Draw 命令による
  myBox 0.1
myBox  0.1
  Draw を省略した場合の
  myBox 0.1
Right
myBox  0.2
  引数の値を変更
  myBox 0.2
Right
myBox2  0.1
  別のボックス
  myBox2 0.1
Down
myChoice
  菱形
Left
myCircle
  丸
Left
myOval
  楕円形
```

```
Left
myBezier
  ベジエ曲線
Down
Scale 0.5 0.5
Oval
  終了
```

出力例



制限 Draw で指定する図形名は、Draw が表れる前に Figure で定義して置く必要があります。

4.3 描画命令

ここでは Figure コマンド[8]で利用できる描画命令について説明します。

4.3.1 座標系

座標系は、左上を原点とし、左右は右が正で左が負、上下は下が正で上が負となります。

なお、基本的な描画領域は、Figure の第二 (xs)、第三引数 (ys) で指定した矩形のエリアサイズ内 ((0,0) - (xs,ys)) を想定しています。以下では、(xs,ys) を終点と呼ぶ事にします。

その上で、座標表現に対しては、次のような意味付けを行います。

[8][sk] Figure コマンドは k-0.06 より追加

正の数値 [(+) **符号無し浮動小数点数**] 原点からの相対値とします。xs, ys より大きな数値を指定した場合は、矩形内をはみ出して描画を行います。符号として「+」を付けてもかまいません

負の数値 [**-符号無し浮動小数点数**] 終点からの相対値とします。-xs, -ys より小さな数値を指定した場合は、矩形内をはみ出して描画を行います。

s, t, l いずれも、「+0」と同じ意味になり、原点の座標を意味します。

e, b, r いずれも、「-0」と同じ意味になり、終点の座標を意味します。

m, c いずれも、中心 (xs/2, ys/2) を表します。

. 「この引数の値を省略値にする」という事を意味します。図形命令は、引数を省略した場合には省略値が利用されるので、全く引数を指定しなくても、なんかしらの描画を行います。

4.3.2 line

書き方 `line sx sy ex ey`

機能 (sx,sy) を始点、(ex,ey) を終点とする直線をひきます。

省略値 sx : s / sy : s / ex : e / ey : e (左上から右下への斜線になります)

例 これは、Box と同じになります。

```
line s s e s
line s s s e
line s e e e
line e s e e
```

制限 LaTeX の制限から傾きに関しては、制限があります。

4.3.3 circle

書き方 `circle r cx cy`

機能 (cx,cy) を中心とし、半径 r の円を書きます。

なお、cx, cy の所に数値を指定した場合は (cx,cy) を原点として扱います。また、r の所に数値を指定した場合は、正の場合は、そのまま、負の場合は省略値からの offset になります。

省略値 cx : 0 / cy : 0 / r : -0

例 xs で指定した横幅の半分を半径とする円になります。

```
circle
```

制限 LaTeX の制限から半径に関しては、制限があります。

4.3.4 oval

書き方 `oval rx ry cx cy`

機能 (cx,cy) を中心とし、横径を rx、縦径を ry とした楕円を書きます。

なお、cx, cy の所に数値を指定した場合は (cx,cy) を原点として扱います。また、rx, ry の所に数値を指定した場合は、正の場合は、そのまま、負の場合は省略値からの offset になります。

省略値 cx：0 / cy：0 / rx：-0 / ry：-0

例 xs, ys の長方形に内接する楕円を書きます。

```
oval
```

4.3.5 bezier

書き方 `bezier x0 y0 x1 y1 x2 y2`

機能 (x0,y) を始点、(x2,y2) を終点とし、(x1,y1) を制御点とするベジエ曲線を書きます。

省略値 x0：0 / y0：0 / x1：m / y1：-0 / x2：-0 / y2：0

例 上向きの二次曲線になります。

```
bezier
```

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
```

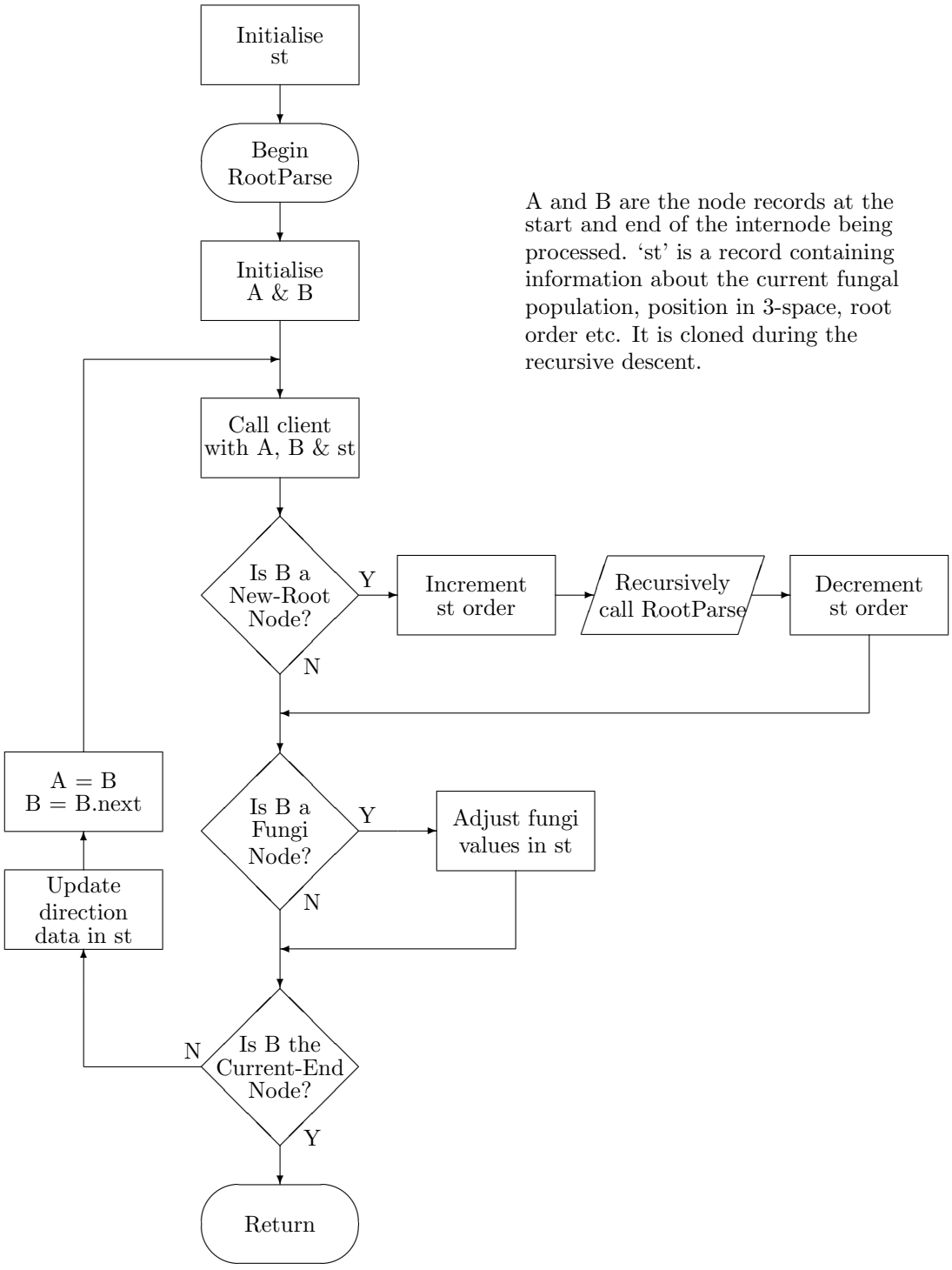
と入力します。リダイレクトは機能しないようです。

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　この資料について

この資料のオリジナルは 2005 年にテリー・ブラウン氏が作成した、「Flow — a syntax to generate flowcharts in the LaTeX picture environment[9]」にあります。

[9] http://mirror.ctan.org/support/flow/flowdoc.pdf

Flow の version up (+k-0.01〜+k-0.06) したものは栗野 (kurino@zaregoto.org) が、元 Nifty の PAF00305 氏の作成された日本語訳を少しいじって、拡張部分を追加したものになっています。

日本語 (翻訳) 版のオリジナルは「Flow - LATEX の picture 環境でフローチャートを描くプログラム[10]」で公開されていたのですが、どうゆうわけだか、最近[11]閉鎖されてしまったので、僕が慌てて、日本語版のコピーを作ったという次第です。

[10] http://homepage2.nifty.com/PAF00305/math/flowdoc_ja

[11] 2012/05/01 頃.. 偶然なのかもしれないのですが、この僕の作業を開始する直前..