tikz-flowchart—Ti*k*Z 流程图绘制宏包*

耿楠†
西北农林科技大学信息工程学院计算机科学系

2019/08/20

摘要

这是一个使用 Ti*k*Z 绘制传统程序流程图的简单宏包，通过定义 `proc`、`test`、`io`、`term` 等 Ti*k*Z 的 `node` 命令样式实现。该宏包核心代码摘录自Brent Longborough设计的流程图绘制样例，参考了 `tikz-imagelabels` 宏包的设计思路，提供了`\flowchartset` 命令以设置绘制参数。

目录

*该文档是 `tikz-flowchart` v1.0.01, dated 2019/08/20 的说明文档。
†https://github.com/registor/tikz-flowchart

Change History

v1.0.01

General: chang .sty file to .dtx file . 1

1 宏包简介

流程图是诸如手册、报告、论文等文档中经常用到的排版元素，`tikz-flowchart` 宏包的目的是为了更为方便地实现传统流程图的绘制。Figure 1是使用 `tikz-flowchart` 宏包绘制`for` 循环结构的一个简单示例。

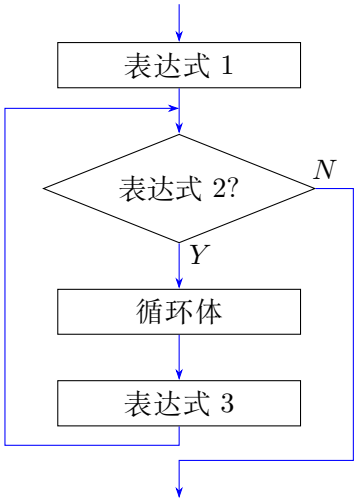

图 1: for 循环流程图

Figure 1由如下代码绘制:

```
\begin{tikzpicture}[scale=0.85]
  % 布置结点
  \node [proc] (p1) {表达式 1};
  \node [test, join, text width = 5em] (t1) {表达式 2?};
  \node [proc] (p2) {循环体};
  \node [proc, join] (p3) {表达式 3};
  % 布置用于连接的坐标结点，同时为其布置调试标记点。
  \node [coord, above = 0.5 of p1] (c1)  {}; \cmark{1}
  \node [coord] (c2) at ($(p1)!0.35!(t1)$)  {}; \cmark{2}
  \node [coord, below = 0.25 of p3] (c3)  {}; \cmark{3}
  \node [coord, below = 0.2 of c3] (c4) {}; \cmark{4}
  \node [coord, below = 0.5 of c4] (c5) {}; \cmark{5}
```

```
  \node [coord, right = 0.5 of t1] (c6) {}; \cmark{6}
  \node [coord, left = 0.5 of t1] (ct) {}; \cmark{t}
  \node [coord] (c7) at (c3 -| ct)  {}; \cmark{7}
  % 判断框连线，每次绘制时，先绘制一个带有一个固定
  % 位置标注的路径 (path)，然后再绘制箭头本身 (arrow)。
  \path (t1.south) -- node [near start, right] {$Y$} (p2.north);
  \draw [norm] (t1.south) -- (p2.north);
  \path (t1.east) -- node [near start, above] {$N$} (c6);
  \draw [norm] (t1.east) -- (c6) |- (c4) -- (c5);
  % 其它连线
  \draw [norm] (c1) -- (p1);
  \draw [norm] (p3.south) |- (c7) |- (c2);
\end{tikzpicture}
```

2 使用方法

2.1 载入宏包

在导言区使用：`\annotationimage[`⟨*debug*⟩`]{`⟨*tikz-flowchart*⟩`}` 命令载入宏包。

2.2 布置结点

3 代码实现

该宏包仅需要载入`tikz` 和`xifthen` 宏包，如果这些宏包没有载入，则自动载入这些宏包。

```
\RequirePackage{tikz}
\RequirePackage{xifthen}
%
```

3.1 宏包选项

使用 `tikz-flowchart`kvoptions 来处理传给该宏包的参数。

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
  family=flowchart,
  prefix=flowchart@,
  setkeys=\kvsetkeys
}
```

定义调试状态参数 (布尔类型)。

```
\DeclareBoolOption[false]{debug}
```

参数解析，获取定义的宏包参数。

```
\DeclareDefaultOption{}
\kvsetkeys{flowchart}{}
\ProcessKeyvalOptions*
```

载入需要的 Ti*k*Z 宏包运行库。

```
\usetikzlibrary{
  arrows.meta,      % 箭头形状
  shapes.geometric, % 几何形状
  chains,           % 链式布局
  calc,             % 坐标计算
}
```

3.2 配置命令

为\flowchartset 命令定义一个pgfkeys 族，所有配置参数 (例如：⟨*norm color*⟩等) 都会存储在/flowchartPGF 键中。这能够确保这些配置参数不会覆盖别的同类参数。

```
\pgfkeys{
  /flowchart/.is family,
  /flowchart/.search also={/tikz},
}

\def\flowchartset{\pgfqkeys{/flowchart}}
```

然后，定义存储这些参数值的宏命令。

```
\flowchartset{
  free color/.store in = \freecolor, % 自由连线颜色
  norm color/.store in = \normcolor, % 常规连线颜色
  cong color/.store in = \congcolor, % 关联连线颜色
  proc fill color/.store in = \procfillcolor, % 顺序处理框填充颜色
  test fill color/.store in = \testfillcolor, % 判断框填充颜色
  io fill color/.store in = \iofillcolor, % 输入/输出框填充颜色
  term fill color/.store in = \termfillcolor, % 开始/结束框填充颜色
  proc text width/.store in = \proctxtwd, % 顺序处理框宽度
  test text width/.store in = \testtxtwd, % 判断框宽度
  io text width/.store in = \iotxtwd, % 输入/输出框宽度
  term text width/.store in = \termtxtwd, % 开始/结束宽度
  chain direction/.store in = \chaindir, % 结点自动布置方向
  minimum node distance/.store in = \minnodedis, % 最小结点间距
```

```
  maximum node distance/.store in = \maxnodedis, % 最大结点间距
  border line width/.store in = \bdlinewd, % 各类流程框边框宽度
  flow line width/.store in = \flowlinewd, % 各类流程线线条宽度
  stealth length/.store in = \stealthlen, % 箭头长度
  stealth width/.store in = \stealthwd, % 箭头宽度
}
```

3.3 默认参数值

为各个参数设置默认值以确保预设的各个宏的值有效，这些值可以由用户单独进行修改，修改后的值会覆盖参数默认值。

```
\flowchartset{
  free color = green, % 自由连线颜色（默认取 green）
  norm color = blue, % 常规连线颜色（默认取 blue）
  cong color = red, % 关联连线颜色（默认取 red）
  proc fill color = white, % 顺序处理框填充颜色（默认取白色）
  test fill color = white, % 判断框填充颜色（默认取白色）
  io fill color = white, % 输入/输出框填充颜色（默认取白色）
  term fill color = white, % 开始/结束框填充颜色（默认取白色）
  proc text width = 8em, % 顺序处理框宽度（默认取 8em）
  test text width = 5em, % 判断框宽度（默认取 5em）
  io text width = 6em, % 输入/输出框宽度（默认取 6em）
  term text width = 3em, % 开始/结束宽度（默认取 3em）
  chain direction = below, % 结点自动布置方向（默认取 below）
  minimum node distance = 6mm, % 最小结点间距（默认取 6mm）
  maximum node distance = 60mm, % 最大结点间距（默认取 60mm）
  border line width = \pgflinewidth, % 各类流程框边框宽度（默认取当前线条宽度）
  flow line width = \pgflinewidth, % 各类流程线线条宽度（默认取当前线条宽度）
  stealth length = 1.5mm, % 箭头长度（默认取 1.5mm）
  stealth width = 1.0mm, % 箭头宽度（默认取 1.0mm）
}
```

3.4 样式定义

以下是所有绘制流程图中需要的样式定义。

```
\tikzset{
```

首先，定义结点布局方式：

```
  start chain = going \chaindir,    % 结点自动布置方向（默认取 below）
  node distance = \minnodedis and \maxnodedis, % 结点间距
```

```
every join/.style = {norm},   % 默认自动连接线的连线样式
```

其次，定义基础绘图样式：

```
% 流程框样式的基础样式
base/.style = {line width = \bdlinewd, % 边框线宽
               draw,                   % 绘制边框
               on chain,               % 沿布局方向绘制
               on grid,                % 沿网格布局
               align=center,           % 内容居中对齐
               minimum height=2ex,      % 流程框最小高度
              },
```

接下来，定义proc、test、io、term 四个\textbackslash node 命令的绘图样式：

```
% 顺序处理框样式
proc/.style={base,           % 基础样式
             rectangle,      % 矩形边框
             text width=\proctxtwd, % 最大文本宽度（超过会自动换行）
             fill=\procfillcolor, % 填充色
            },
% 判断框样式
test/.style={base,           % 基础样式
             diamond,        % 菱形边框
             aspect=2.5,       % 长高比例
             text width=\testtxtwd, % 最大文本宽度（超过会自动换行）
             fill=\testfillcolor, % 填充色
            },
% 输入/输出框样式
io/.style={base,                      % 基础样式
           trapezium,                 % 平行四边形
           trapezium left angle=70,   % 平行四边形左倾角
           trapezium right angle=110, % 平行四边形右倾角
           text width=\iotxtwd,            % 最大文本宽度（超过会自动换行）
           fill=\iofillcolor, % 填充色
          },
% 开始/结束框样式
term/.style={proc,            % 基于 proc 样式
             rounded corners=2.0mm, % 为矩形添加圆角属性
             text width=\termtxtwd, % 最大文本宽度（超过会自动换行）
             fill=\termfillcolor, % 填充色
            },
```

再下来，定义流程线交点绘制样式：

```
% 流程连接点样式
connector/.style = {draw,              % 绘制边框
                    circle,            % 圆形
                    node distance=3cm, % 节点间距
                   },
% 绕接连线点样式（不相交的两个交汇路径）
connect/.style args={(#1) to (#2) over (#3) by #4}{
  insert path={
    let \p1=($(#1)-(#3)$), \n1={veclen(\x1,\y1)},
      \n2={atan2(\y1,\x1)}, \n3={abs(#4)}, \n4={#4>0 ?-180:180}  in
      (#1) -- ($(#1)!\n1-\n3!(#3)$) arc (\n2:\n2+\n4:\n3) -- (#2)
  }
},
```

还需要定义流程线转角点node 命令样式：

```
% coord 结点样式（用于布置流程线连接点）
coord/.style={coordinate,                % 笛卡尔坐标系
              %on chain,                  % 沿布局方向绘制
              %on grid,                   % 沿网格布局
              node distance=6mm and 25mm, % 节点间距
             },
```

为cmark 调试标记命令绘制样式：

```
% nmark 结点样式（用于布置调试坐标标记点）
nmark/.style={draw,                          % 绘制边框
              cyan,                          % 青色
              circle,                        % 圆形
              font={\sffamily\bfseries}, % 字体
             },
```

另外，需要定义各类流程线绘制样式：

```
% ------------------------------------------------
% 无箭头连线样式
lnorm/.style={line width = \flowlinewd,  % 线宽
              draw,                      % 绘制
              \normcolor,                % 颜色
             },
lfree/.style={line width = \flowlinewd,
              draw,
              \freecolor,
             },
lcong/.style={line width = \flowlinewd,
              draw,
              \congcolor,
```

```
            },
% 流程线实心交点样式
dotnorm/.style={draw,              % 绘制
                fill = \normcolor, % 填充颜色
                \normcolor,        % 颜色
               },
dotfree/.style={draw,
                fill = \freecolor,
                \freecolor,
               },
dotcong/.style={draw,
                fill = \congcolor,
                \congcolor,
               },
% 流程线空心交点样式
cdotnorm/.style={draw,       % 绘制
                 \normcolor, % 颜色
                },
cdotfree/.style={draw,
                 \freecolor,
                },
cdotcong/.style={draw,
                 \congcolor,
                },
% 带箭头连线样式
norm/.style={line width = \flowlinewd,     % 线宽
             -{Stealth[length=\stealthlen, % 箭头长度
                       width=\stealthwd,   % 箭头宽度
                      ]
              },
             draw,                         % 绘制
             \normcolor,                   % 颜色
            },
free/.style={line width = \flowlinewd,
             -{Stealth[length=\stealthlen,
                       width=\stealthwd,
                      ]
              },
             draw,
             \freecolor,
            },
cong/.style={line width = \flowlinewd,
```

```
                -{Stealth[length=\stealthlen,
                          width=\stealthwd,
                         ]
                 },
                draw,
                \congcolor,
               },
```

最后，再定义一个流程线标注文本样式：

```
  % 斜体字样式
  it/.style={font={\small\itshape}},
}
```

3.5 调试命令定义

为便于绘制过程中，能够直观连接各个流程线转角点，定义cmark 命令，以绘制转角点标记。

```
%% 判断是否为宏包传入了 debug 参数以打开调试功能，若没有传入 debug 参数，则关闭调试功能。
\ifflowchart@debug
```

传入了 debug 参数，创建用于调试的图层。

```
  % 设置一个用于调试的标记符号图层，注意确保这一图层位于顶层
  \pgfdeclarelayer{marx}
  \pgfsetlayers{main,marx}
```

定义\textbackslash cmark 命令。

```
  \newcommand{\cmark}[2][]{%
  \begin{pgfonlayer}{marx}
    \node [nmark] at (c#2#1) {#2};
  \end{pgfonlayer}{marx}
  }
```

未传入了 debug 参数，定义一个空的cmark 命令。

```
\else
  \newcommand{\cmark}[2][]{\relax}
\fi
```