

# {tikzcd}

## 基于 TikZ 的交换图包

版本 0.9f      11.19, 2018

TikZ 包本身是可以画出交换图和其他数学图形的, 并生成高质量的图片。而 tikz-cd 包则是为画交换图提供了一系列简单的命令。TikZ 包也是可行的, 但是没有必要, 因为本包这里的例子几乎覆盖了大部分的情形。虽然画交换图的还有很多包, 比如 amscd, XY-pic, 但是这里基于 TikZ 的 tikz-cd 包语法与 tikz 接近, 能够画出更为复杂更加漂亮的交换图。

## 目录

### 1 初始化

本包被 TexLive 自动收录, 要加载此包, 只需要在导言区输入 `\usepackage{tikz-cd}` 或者加载 TikZ 包以后再导入库 `\usetikzlibrary{cd}`。

#### 1.1 画图

基本的话交换图的语法是下面的环境

```
1 \begin{tikzcd}[<选项>]
2 <内容>
3 \end{tikzcd}
```

此环境生成一个矩阵, 类似于 `tabular` 环境, [`< 选项 >`] 用来修改图的外观, 在 TikZ 包中的任意选项都可在这里使用。

`tikzcd` 环境中的元素都是以数学模式排版, 但是您也可以把它放在 `\[...\]` 或者 `equation` 环境中, 使得交换图居中。

#### 1.2 插入箭头

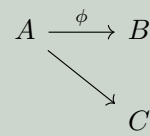
在 `tikzcd` 环境内, 下面的命令是一样的, 都生成箭头

```
1 \arrow[<选项>]
2 \ar[<选项>]
```

这里的 [`< 选项 >`] 是一系列逗号隔开的选项, 用来指定箭头指向, 箭头类型, 增加标签等。

箭头的指向是靠一串包含 `r, l, d, u` (分别表示右左下上) 的字符所确定, 标签可以放在箭头上, 其放置语法与 TikZ 的 `quotes` 库的引用语法相同, 注意下面的 “`phi`” 的使用

```
\begin{tikzcd}
A \arrow{rd} \arrow{r,"\phi"} & B \\
& C
\end{tikzcd}
```

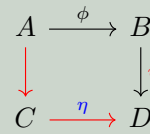


要想进一步修改箭头的外观，注意 [`< 选项 >`] 可以使用 TikZ 的 `\path` 命令的任意参数，类似的，标签也可以通过下面的语法接受额外的选项：

#### 1 “<标签内容>”<标签选项>

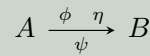
如果 `< 标签内容 >` 或 `< 标签选项 >` 含有逗号，那么它们需要用 `{}` 包起来。

```
\begin{tikzcd}
A \arrow{r,"\phi"} \arrow{d, red}
& B \arrow{d,"\psi" red} \\
C \arrow{r, red,"\eta" blue}
& D
\end{tikzcd}
```



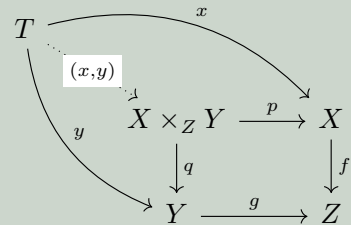
箭头可以有任意多个标签，反复使用 `quotes` 选项即可。下面的例子展示了如何控制标签的位置。尤其注意 `< 标签选项 >` 中的 `swap` 使得标签在箭头的另一侧，这里的 `swap` 等效为撇号'。

```
\begin{tikzcd}
A \arrow[->,>=stealth,r,"\phi" near start,"\psi"swap,
"\eta" near end] & B
\end{tikzcd}
```







下面给出两个实际的例子。

```
\begin{tikzcd}
T
\arrow{drr, bend left,"x"}
\arrow{ddr, bend right,"y"}
\arrow{dr, dotted,"{(x,y)}description"} & & \\
& X \times_Z Y \arrow{r,"p"} \arrow{d,"q"} & X \\
& Y \arrow{r,"g"} & Z
\end{tikzcd}
```











**Arrows with hook**

<code>{hook}</code>	
<code>{hook'}</code>	
<code>{hookrightarrow}</code>	
<code>{hookleftarrow}</code>	







**Arrows with tail**

<code>{tail}</code>	
<code>{rightarrowtail}</code>	
<code>{leftarrowtail}</code>	




**Two-headed arrows**

<code>{two heads}</code>	
<code>{twoheadrightarrow}</code>	
<code>{twoheadleftarrow}</code>	





**Harpoons**

<code>{harpoon}</code>	
<code>{harpoon'}</code>	
<code>{rightharpoonup}</code>	
<code>{rightharpoondown}</code>	
<code>{leftharpoonup}</code>	
<code>{leftharpoondown}</code>	





**Dashed arrows**

<code>{dashed}</code>	
<code>{dashrightarrow}</code>	
<code>{dashleftarrow}</code>	

**Squiggly arrows**

<code>{squiggly}</code>	
<code>{rightsquigarrow}</code>	
<code>{leftsquigarrow}</code>	
<code>{leftrightsquigarrow}</code>	

**Non-arrows**

<code>{no head}</code>	
<code>{no tail}</code>	
<code>{dash}</code>	
<code>{equal}</code>	

```
\begin{tikzcd}
  A \arrow[r, tail, two heads, dashed] & B \\
\end{tikzcd}
```

$$A \rightharpoonright \twoheadrightarrow B$$

TikZ 本身的箭头选项 `latex,stealth` 也可以使用,以及加载了 `arrows.meta` 库以后的 `Latex,Stealth`。

```
\begin{tikzcd}
  A \arrow[r,->,>=Stealth] & B \\
  C \arrow[r,->,>=latex] & D \\
\end{tikzcd}
```

$$A \longrightarrow B$$

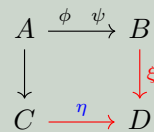
$$C \longrightarrow D$$
**1.4 箭头的其他语法**

下面的箭头命令形式是在标签的 `quotes` 语法出来之间的形式,现在看来似乎复杂了,但是为了向前的兼容性,其功能仍然是可用的。

`\arrow[<选项>]{<方向>}<标签>`

其等价的命令`\ar`也能用这种形式表达，这里是一个例子

```
\begin{tikzcd}
A \arrow{d} \arrow{r}[near start]{\phi}[near end]{\psi}
& B \arrow[red]{d}{\xi} \\
C \arrow[red]{r}[blue]{\eta}
& D
\end{tikzcd}
```



还有更加进一步简化的命令：

```
\rar[<选项>]<标签> \lar[<选项>]<标签> \dar[<选项>]<标签> \uar[<选项>]<标签>
\drar[<选项>]<标签> \urur[<选项>]<标签> \dlar[<选项>]<标签> \ular[<选项>]<标签>
```

其中第一个等价于`\arrow[<选项>]{r}<标签>`，剩下的同理。

## 2 控制交换图的外形

本节描述由此包定义的一系列专用化选项的关键词。要想全局设置的话，可以方便地使用如下命令：

```
\tikzcdset{<选项>}
```

除了此包中的关键词，在 `TikZ` 中的参数也会影响图的外观。

### 2.1 一般选项

```
/tikz/commutative diagrams/every diagram (style, no value)
```

这个样式应用于 `tikzcd` 环境。初始情形下，它包含如下选项：

```
/tikz/row sep=normal,
/tikz/column sep=normal,
/tikz/baseline=0pt
```

这里的`baseline=0pt`设置使得公式的编号正确放置（有一个特例，单行的图固定在矩阵的基底上，这正是您想要的）。

```
/tikz/commutative diagrams/diagrams=<选项> (no default)
```

这个关键词使得`<选项>`附属于样式`every diagram`。

```
/tikz/commutative diagrams/every matrix (style, no value)
```

此样式用于 `TikZ` 的矩阵，初始情形下，它的设置为：

```
/tikz/inner sep=0pt
```

```
/tikz/commutative diagrams/every cell (style, no value)
```

此样式也是用于矩阵，初始情形下

```
/tikz/shape=asymmetrical rectangle,
/tikz/inner xsep=1ex,
/tikz/inner ysep=0.85ex
```

`inner xsep, inner ysep` 选项决定了交换图的任意一个元素和指向它的箭头的距离。

`/tikz/commutative diagrams/cells=< 选项 >` (no default)

此关键词将<选项>附属给样式 `every cell`。

`/tikz/commutative diagrams/row sep=< 尺寸 >` (no default)

此关键词的行为类似于 TikZ 的前端 `/tikz/row sep` 选项。初始可取的尺寸及对应值如下：

tiny	small	scriptsize	normal	large	huge
0.45em	0.9em	1.35em	1.8em	2.7em	3.6em

注意，用 `\tikzcdset` 全局设置 `row sep=1cm` 是无效的，因为 `row sep` 选项在每个图开始的时候都会重置。要想使得每个图都是 `row sep=1cm`，可以修改 `normal` 的定义

```
\tikzcdset{row sep/normal=1cm}
```

您还可以定义新的尺寸，但是注意 PGF 要求新的关键词要被直接初始化，例如定义一个尺寸 `my size` 为 `1ex`，您应该使用

```
row sep/my size/.initial=1ex
```

`/tikz/commutative diagrams/column sep=< 尺寸 >` (no default)

此关键词和上面的 `row sep` 类似。可用的初始尺寸如下

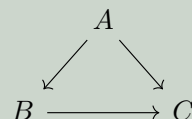
tiny	small	scriptsize	normal	large	huge
0.6em	1.2em	1.8em	2.4em	3.6em	4.8em

`/tikz/commutative diagrams/sep=< 尺寸 >` (no default)

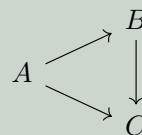
此关键词相当于同时设置 `row sep=<尺寸>`，`column sep=<尺寸>`。

在下面的例子中，如果 `column sep` 或者 `row sep` 设置不合理，会使得三角形看起来太宽或太高。

```
\begin{tikzcd}[column sep=small]
& A \arrow[dl] \arrow[dr] & \\
B \arrow{rr} & & C \\
\end{tikzcd}
```



```
\begin{tikzcd}[row sep=tiny]
& B \arrow{dd} \\
A \arrow{ur} \arrow{dr} & \\
& C \\
\end{tikzcd}
```



`/tikz/commutative diagrams/cramped=< 尺寸 >` (style, no value)

默认情形下，交换图元素周围会添加大量的空白，这对大的显示的图是合理的。此关键词除去了多余的空白，为小的图所定制。

下图稍微显示了 `cramped` 和非 `cramped` 样式的区别。

This `\begin{tikzcd} A \arrow[r] & B \end{tikzcd}` is a regular diagram.

This `\begin{tikzcd}[cramped, sep=small] A \arrow[r] & B \end{tikzcd}` is a cramped diagram.

This `$A \to B$` is just a formula.

This  $A \longrightarrow B$  is a regular diagram.

This  $A \rightarrow B$  is a cramped diagram.

This  $A \rightarrow B$  is just a formula.

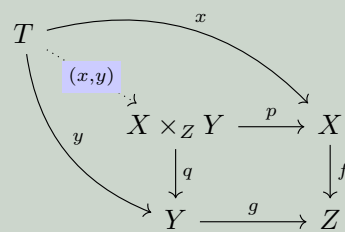
`/tikz/commutative diagrams/math mode=< 布尔变量 >` (default true)

此关键词决定图里的内容是否以数学模式输出。如果全局设置或者在图里面设置的话，它会影响图的元素和箭头的标签，如果把它用在`\arrow`选项里，它只影响标签。

`/tikz/commutative diagrams/background color=< 颜色 >` (no default 初始为白色)

此关键词存储颜色名，然后用来被填充背景的样式所读取，比如`description`和`crossing over`。注意此关键词并不会使得图的背景被填充。

```
\begin{tikzcd}
T
\arrow[ddr, bend left, "x"]
\arrow[ddr, bend right, "y"]
\arrow[dr, dotted, background color=blue!20,
"({x,y})"description] & & \\
& X \times_Z Y \arrow[r, "p"] \arrow[d, "q"] & X \\
& & \downarrow f \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}
```



## 2.2 箭头的全局选项

`/tikz/commutative diagrams/every arrow` (style, no value)

此样式应用于`\arrow`。初始情形下，它包含如下设置：

```
/tikz/draw,
/tikz/line width=rule_thickness,
rightarrow
```

`/tikz/commutative diagrams/arrows=< 选项 >` (no default)

此关键词将<选项>附属给样式`every arrow`。

`/tikz/commutative diagrams/arrow style=< 样式 >` (no default)

此关键词决定箭头类型是 §?? 节中所列的哪一种。初始设置对于使用 Computer Modern 字体的任意字号都是适用的。可用的<样式>选择有

**Latin Modern** 稍微修改初始设置，用于 Latin Modern 字体的任意字号。

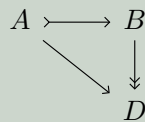
**math font** 这个设置用到了 Glyph meta arrow 箭头类型。

**tikz** 此设置用了 TikZ 的 `arrows.meta` 库。

关键词的设置一般在导言区，而且只设置一次。

如果您是用来 Computer Modern 和 Latin Modern 以外的字体，您最好选择`math style`样式。这种设置并不能保证和所有字体吻合，但是在很多场合得到的结果都是很好的。如果`math font`样式产生不满意的结果，您可以考虑`tikz`样式，并且设置`/tikz/>=`值使得最切合您的字体。

```
\tikzcdset{
  arrow style=tikz,
  diagrams=>={Straight Barb[scale=0.8]}}
\begin{tikzcd}
  A \arrow[r, tail] \arrow[rd] & B \arrow[d, two heads] \\
  & D
\end{tikzcd}
```



### 2.3 箭头的绝对放置

`\arrow`命令一般是在命令出现的地方生成一个箭头，然后指向相对它的一个位置，下面的关键词会覆盖这种行为，使得箭头的起止点都被选定。

`/tikz/commutative diagrams/from=< 参数 >` (no default)

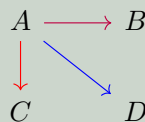
如果 `< 参数 >` 的形式是 `< 行数 >-< 列数 >`，或者是一串由 `r,l,d,u` 组成的字符，这些关键词设定箭头起点是交换图矩阵中相应的元素。否则，此参数就被假定为节点的名字并且设定为箭头的起点。

`/tikz/commutative diagrams/to=< 参数 >` (no default)

类似于`from`，但是针对箭头终点的。

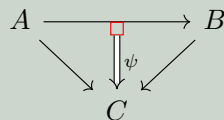
我们可以给 TikZ 矩阵中的每一个元素使用`[<options>]`语法给它命名，就像下面例子中的矩阵元  $C$  一样。如果您想用`from`或`to`来引用节点的话，千万不要用只含有`l,r,u,d`字符的名字来命名。下面来说明这些关键词的不同用处。

```
\begin{tikzcd}
  A \arrow[to=Z, red] \arrow[to=2-2, blue]
  & B \\
  |[alias=Z]| C
  & D
  \arrow[from=ul, to=1-2, purple]
\end{tikzcd}
```



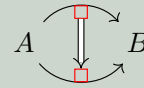
在下面的例子中，使用了空标签以便后面引用。`draw=red`选项用来指示这些空节点的位置，当然如果您使用这个技巧，您就想移除这些节点了，

```
\begin{tikzcd}[column sep=scriptsize]
  A \arrow[dr] \arrow[rr, "{name=U, below, draw=red}"] {}
  & & B \arrow[dl] \\
  & C \arrow[Rightarrow, from=U, "\psi"]
\end{tikzcd}
```





```
\begin{tikzcd}
A \arrow[r, bend left=50, ""{name=U, below, draw=red}]
\arrow[r, bend right=50, ""{name=D, draw=red}]
& B
\arrow[Rightarrow, from=U, to=D]
\end{tikzcd}
```



## 2.4 幻影箭头

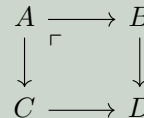
有时候需要在图的格点之外插入一个符号，实现这种最简单的方法就是把标签设成一种不可见的箭头。

`/tikz/commutative diagrams/to=< 参数 >` (no default)

生成不可见的箭头，指向此箭头的标签也是不可见的，标签会被固定在其连线的中点，以`\textstyle`的形式排版。要想得到更小的标签，可以使用`\scriptstyle`命令。

在下面的图片中，从  $A$  到  $D$  的箭头包含了`phantom`选项，而`\ulcorner`符号 ( $\ulcorner$ ) 插在靠近起点  $A$  的地方。

```
\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very
near start]
& B \arrow[d] \\
C \arrow[r]
& D
\end{tikzcd}
```



## 2.5 合理调整箭头的位置

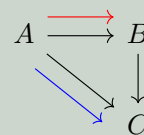
`/tikz/commutative diagrams/shift left=< 距离 >` (default 0.56ex)

通过<距离>参数使得箭头向左偏移。

`/tikz/commutative diagrams/shift right=< 距离 >` (default 1)

等价于`shift left=-<距离>`。

```
\begin{tikzcd}
A \arrow[r, red, shift left=1.5ex] \arrow[r]
\arrow[dr, blue, shift right=1.5ex] \arrow[dr]
& B \arrow[d, purple, shift left=1.5ex] \arrow[d] \\
& C
\end{tikzcd}
```



默认的`shift left`和`shift right`值适用于一系列平行的箭头，而无量纲的参数能帮助生成多重平行符号。

```
\begin{tikzcd}
A \arrow[r]
& B \arrow[r, shift left]
\arrow[r, shift right]
& C \arrow[r]
\arrow[r, shift left=2]
\arrow[r, shift right=2]
& \cdots
\end{tikzcd}
```

$$A \longrightarrow B \rightrightarrows C \rightrightarrows \cdots$$

`/tikz/commutative diagrams/shift=< 坐标 >`

(no default)

`/tikz/commutative diagrams/xshift=< 坐标 >`

(no default)

`/tikz/commutative diagrams/yshift=< 坐标 >`

(no default)

```
\begin{tikzcd}
A \arrow[r, yshift=0.7ex] \arrow[r, yshift=-0.7ex]
& B \arrow[d, xshift=0.7ex] \arrow[d, xshift=-0.7ex] \\
& C
\end{tikzcd}
```

$$\begin{array}{ccc} A & \rightrightarrows & B \\ & & \Downarrow \\ & & C \end{array}$$

`/tikz/commutative diagrams/start anchor=[坐标变换]< 锚位置 >`

(no default)

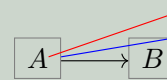
此关键词确定了箭头起点的位置，可选项是额外的坐标变换，空的<anchor>选项将不指定初始位置，也就是一般的情形了。

`/tikz/commutative diagrams/end anchor=< 距离 >`

(no default)

此关键词的设置也是类推，但是针对箭头的中点。

```
\begin{tikzcd}[cells={nodes={draw=gray}}]
A \arrow[r, black]
\arrow[r, blue, end anchor=north east]
\arrow[r,
red,
start anchor={[xshift=-1ex]},
end anchor={[yshift=2ex]north east}]
& B
\end{tikzcd}
```



`/tikz/commutative diagrams/shorten=< 距离 >`

(no default)

此关键词缩短此箭头两端的长度。

```
\begin{tikzcd}
A \arrow[r, shift left]
\ar[r, shorten=2mm, shift right]
& B
\end{tikzcd}
```

$$A \rightrightarrows B$$

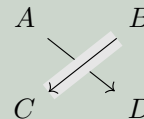
值得注意的是箭头两端长度的缩减可以用 TikZ 中的选项 `shorten` 和 `shorten >`。

## 2.6 三维交换图

`/tikz/commutative diagrams/crossing over` (style, no value)

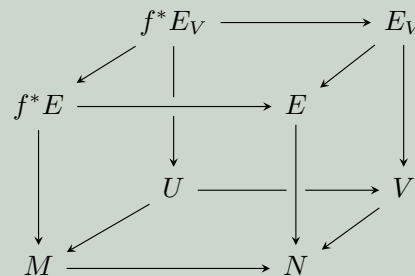
此选项在当前箭头下方画一条厚线，颜色为 `background color`，模拟两个交叉箭头的效果。

```
\begin{tikzcd}[background color=gray!20]
  A \arrow{dr} & B \arrow{dl, crossing over} \\
  C & D
\end{tikzcd}
```



由于箭头是按照其被读取的顺序来画的，因此有必要推迟画某些箭头来得到所要的效果，这可以通过 `from` 选项，如下图所示。

```
\tikzcdset{
  arrow style=tikz,
  diagrams={>=stealth}
}
\begin{tikzcd}[row sep=scriptsize, column
  sep=scriptsize, background color=green!20!black!20]
  & f^*E_V & \longrightarrow & E_V \\
  f^*E & \searrow & \downarrow & \swarrow \\
  & E & \longrightarrow & V \\
  & \downarrow & \downarrow & \downarrow \\
  & U & \longrightarrow & V \\
  f^*E & \searrow & \downarrow & \swarrow \\
  & M & \longrightarrow & N \\
  & \downarrow & \downarrow & \downarrow \\
  & M & \longrightarrow & N
\end{tikzcd}
```



`/tikz/commutative diagrams/crossing over clearance=< 距离 >` (no default, 初始值 1.5ex)

此关键词设置由 `crossing over` 所画 `backgrounded-color` 线条的宽度。

## 2.7 标签选项

`/tikz/commutative diagrams/every label=< 距离 >` (style, no value)

此样式应用于由 `\arrow` 命令生成的每个标签。初始值为

```
/tikz/auto,
/tikz/font=<something>,
/tikz/inner sep=0.5ex
```

这里的 `<something>` 是使得数学模式中应用 `\scriptstyle` 模式的东西。

`/tikz/auto` 选项使得标签在箭头前进方向的左边。选项 `/tikz/inner sep` 控制标签和相应箭头的距离。

`/tikz/commutative diagrams/labels=< 选项 >` (no default)

此关键词使得 `<选项>` 附属于 `every label`。

`/tikz/commutative diagrams/marking` (style, no value)

`/tikz/commutative diagrams/description`

(style, no value)

此样式使得标签放在箭头上，颜色为 background color。标签周围的间隙由 `/tikz/inner sep` 决定。

```
\begin{tikzcd}[background color=green!20!black!20]
  A \arrow[r, "\phi" description] & B
\end{tikzcd}
```

$$A \xrightarrow{\phi} B$$