TikZ の使い方 (圏論編)

alg-d
http://alg-d.com/math/kan_extension/

2019 年 2 月 16 日

これは TikZ のマニュアル [1] から，圏論で図式を描く場合に使えそうな部分を抜き出したものである．

目次

1 Ti*k*Z

Ti*k*Z は Ti*k*Z ist *kein* Zeichenprogramm の略である．(英語にすると Ti*k*Z is not a drawing program.)

2 tikzpicture 環境

Ti*k*Z では，基本的に `tikzpicture` 環境を使用し，この環境の中に `\path ～～～～;` という形式で記述することで図形を描いていく (セミコロンが必要なことに注意)．`\path` は複数書くことが可能である．よって実際に使用する際には次のような形式になる．

```
\begin{tikzpicture}
\path ～～～～;
\path ～～～～;
\path ～～～～;
\end{tikzpicture}
```

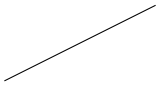
～～～～ の部分には Operation と呼ばれるコマンドを書く．詳細な説明は以降の節でするが，ここでは例として `\path (0,0) --(2,1);` を挙げておく．これは $(0,0)$ と $(2,1)$ を結ぶ線分を定義するコマンドである．この線分のように `\path` で定義されるものを path と呼ぶ．但し，path は定義しただけでは描画されない．従って

```
\begin{tikzpicture}
\path (0,0) --(2,1);
\end{tikzpicture}
```

と tex ファイル上に書いても何も表示されない．path を描画したい場合には `\path` の後ろに `[draw]` と書く．つまりこの場合は `\path[draw] (0,0) --(2,1);` となる．更に，`\path[draw]` の略記として `\draw` と書くこともできるため，この場合は `\draw (0,0) --(2,1);` とも書くことができる．以上を踏まえて

```
\begin{tikzpicture}
\draw (0,0) --(2,1);
\end{tikzpicture}
```

と tex ファイル上に書いてコンパイルすると，次のような図が表示されることになる．

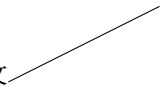


tikzpicture 環境は本文中にも数式中にも書くことができるため，

```
\[
\begin{tikzpicture}
\draw (0,0) --(2,1);
\end{tikzpicture}
\]
```

のように \[\] の中に書けば上のように表示されるし，

```
このように本文
\begin{tikzpicture}
\draw (0,0) --(2,1);
\end{tikzpicture}
中に表示される
```

のように本文中に書けば，このように本文  中に表示される．

2.1 略記法

tikzpicture 環境には簡単に使用するための略記法が用意されている．

- \tikz{コマンド}
  これは\begin{tikzpicture}コマンド;\end{tikzpicture} と同様である．(注: セミコロンは付与される．)
- \tikz テキスト;
  これは\begin{tikzpicture}テキスト;\end{tikzpicture} と同様である．(注: セミコロンが必要．)

故に先に使用した例

```
\begin{tikzpicture}
\draw (0,0) --(2,1);
\end{tikzpicture}
```

であれば \tikz{\draw (0,0) --(2,1)} や \tikz \draw (0,0) --(2,1); と書くことができる．

2.2 オプション引数について

TikZ では，各所でオプション引数を使って様々な設定を行う．通常の TEX と同様，これは [ ] の中にカンマ区切りで記述する．記述は **オプション名=設定値** の形式で行う．例えば \path に対して，オプション draw に red を設定し，fill に blue を設定する場合は \path[draw=red,fill=blue] となる．ここで注意として，**設定値** にカンマが含まれる場合は，それを明示するために **設定値** 全体を { } で囲う必要がある．例えばオプション shift に (1,1) を設定する場合 \path[shift={(1,1)}] となる．

もう一つ注意として，オプションには初期値と既定値が設定されている場合がある．

- 初期値とは，最初から設定されている値であり，例えば `\path` のオプション `line width` の初期値は `0.4pt` である．よってこの場合，単に `\path` と書くのは `\path[line width=0.4pt]` と書くのと同じである．
- 既定値が設定されているオプションの場合，**=設定値** の部分を省略することができ，省略した場合は既定値が使われる．例えば `\path` のオプション `rounded corners` の既定値は `4pt` である．よってこの場合 `\path[rounded corners]` と書くことができ，これは `\path[rounded corners=4pt]` と同じである．

3 座標系

Ti*k*Z では「点」を指定することで様々なものを描いていくことになる．「点」の指定形式には様々なものがあり，（**座標系名** `cs:`**設定値**）の書式で指定することができる．**座標系名** の部分には，以下で述べる `canvas` などのような名称を指定する．**設定値** の部分はオプション同様，設定する内容をカンマ区切りで設定する．**座標系名** には以下のようなものがある．

(1) `canvas` ([1]13.2.1)
所謂 xy 座標であり，（`canvas cs:x=1cm,y=2cm`）のように x 軸方向と y 軸方向の長さを指定する．この場合は原点から x 軸方向に 1 cm，y 軸方向に 2 cm の位置を表す．ただこれはよく使う座標系なので，カンマを使って（`1cm,2cm`）のように書くこともできる．また更に単位を省略して（`1,2`）と書くこともできる．この場合単位はデフォルトでは cm になる ([1]11.1).

(2) `canvas polar` ([1]13.2.1)
これは所謂極座標で，（`canvas polar cs:angle=30,radius=1cm`）のように書く．これは角度 30 度，距離 1 cm の点を表す．この座標系もよく使うので（`30:1cm`）のようにコロンを使用して書くこともできる．なお `angle` には -360〜720 を設定することができる．

(3) `xyz` ([1]13.2.1) 省略.

(4) `xyz polar` ([1]13.2.1) 省略.

(5) `xy polar` ([1]13.2.1) 省略.

(6) `barycentric` ([1]13.2.2) 省略.

(7) `tangent` ([1]13.2.4) 省略.

(8) `perpendicular` ([1]13.3.1) 省略.

(9) `intersections` ([1]13.3.2) 省略.

(10) `node` ([1]13.2.3)
これは「名前」を使用して別の点を参照する方法である．詳しくは第 5.3 節で後述する．

以上は絶対的な座標指定であるが，相対的な指定もできる．その場合は `++(1,2)` のように書く．これは直前に現れた点から x 軸方向に 1，y 軸方向に 2 移動した位置を表す．

4 Path

ここでは `\path` の中で使用する「Operation」について説明する．

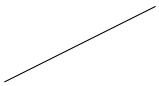
4.1 The Move-To Operation ([1]14.1)

`\path` の内部で単に座標を書いた場合，これは Move-To Operation といい，「現在位置」を指定した座標に変更する効果を持つ (「現在位置」の意味は次の節以降を読むことで分かるであろう)．

先程使用した `\tikz \draw (0,0) --(2,1);` の場合は `(0,0)` が Move-To Operation である．

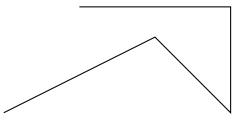
4.2 The Line-To Operation ([1]14.2)

`--(座標)` を Line-To Operation といい，「現在位置」から指定した「座標」までの線分を描く (「座標」の指定方法は第 3 節で述べた通り)．再び `\tikz \draw (0,0) --(2,1);` を考えると，この場合は `--(2,1)` が Line-To Operation である．よってこの例では `\path` の中に 2 つの Operation が入っていることになる．このような場合，Operation は前から順に処理される．故にこの例では，まず Move-To Operation `(0,0)` で $(0,0)$ に移動した後，Line-To Operation `--(2,1)` で，$(0,0)$ から $(2,1)$ への線分を描くという動作になるため，結果として次のような図が描かれる．



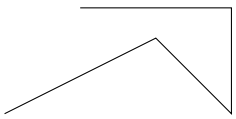
今の例は単なる線分であるが，Line-To Operation は「現在位置」も移動するため，例えば次のようにすることで折れ線を描くことができる．

```
\tikz \draw (0,0) --(2,1) --(3,0) --(3,1.4) --(1,1.4);
```



第 3 節で述べた `++` を使えば次のように書くこともできる．

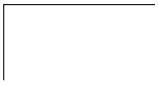
```
\tikz \draw (0,0) --++(2,1) --++(1,-1) --++(0,1.4) --++(-2,0);
```



`--` の他にも `-|` や `|-` という指定方法も存在する．`-|` は点を横線 → 縦線の順で使って繋げる．逆に `|-` は縦線 → 横線の順となる．

```
\tikz \draw (0,0) -|(2,1);
```

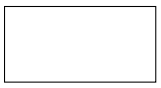
```
\tikz \draw (0,0) |-(2,1);
```



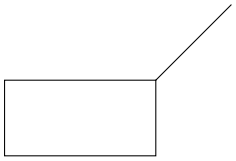
4.3 The Rectangle Operation ([1]14.4)

`rectangle(座標)` で「現在位置」と「座標」を頂点とする長方形を描く．例えば次のようになる．

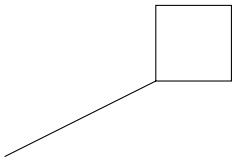
```
\tikz \draw (0,0) rectangle(2,1);
```



また `rectangle` を `--` と組み合わせて使うこともできる (組み合わせて使えるのは，以下で述べる他の Operation についても同様である)．例えば `\tikz \draw (0,0) rectangle(2,1) --(3,2);` とすると「$(0,0)$ へ移動し」「$(2,1)$ を頂点とする四角を描き」「$(3,2)$ へと線分を描く」というようになるため次のようになる．



一方，`\tikz \draw (0,0) --(2,1) rectangle(3,2);` とした場合は「$(0,0)$ へ移動し」「$(2,1)$ へと線分を描き」「$(3,2)$ を頂点とする四角を描く」となるので次のようになる．



Move-To Operation も組み合わせれば次のようなことも可能である (これは「$(0,0)$ へ移動し」「$(0,1)$ へと線分を描き」「$(0.5,0)$ へ移動し」「$(1,1)$ を頂点とする四角を描く」となる)．

```
\tikz \draw (0,0) --(0,1) (0.5,0) rectangle(1,1);
```

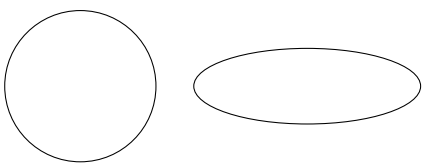
もちろんこれは，`\draw` を 2 つ使って次のように書いてもよい．

```
\begin{tikzpicture}
\draw (0,0) --(0,1);
\draw (0.5,0) rectangle(1,1);
\end{tikzpicture}
```

4.4 The Circle and Ellipse Operations ([1]14.6)

`circle` で「現在位置」を中心とする楕円を描く．大きさなどはオプションで指定する．例えば次のようになる．

```
\tikz \draw (0,0) circle[radius=1]
(3,0) circle[x radius=1.5,y radius=0.5];
```



なお `\draw (0,0) circle[x radius=1.5,y radius=0.5];` は `\draw (0,0) ellips(1.5 and 0.5);` と書くこともできるが後者は古い書き方である．

4.5 The Arc Operation ([1]14.7)

楕円弧を描く Operation である．詳細は省略．

4.6 The Parabola Operation ([1]14.9)

放物線を描く Operation である．詳細は省略．

4.7 The Sine Operation ([1]14.10)

サインカーブを描く Operation である．詳細は省略．

4.8 The Curve-To Operation ([1]14.3)

`..` `controls`（座標 1）`and`（座標 2）`..`（座標 3）で 3 次ベジェ曲線を描くことができる．例えば次のようになる．

```
\tikz \draw (0,0) .. controls (2,0) and (3,1) .. (5,1);
```

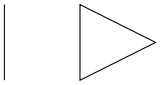
座標 1 と座標 2 が一致している場合は `and`（座標 2）を省略することができる．

```
\tikz \draw (0,0) .. controls (2,1) .. (5,0);
```

4.9 cycle ([1]14.2.1)

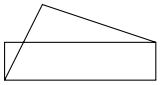
（座標）の代わりに `cycle` と書くことができる．これは直前に使用した Move-To Operation の座標を表す．

```
\tikz \draw (0,0) --(0,1) (1,0) --(1,1) --(2,0.5) --cycle;
```



```
\tikz \draw (0,0) --(1,1) -|cycle;
```

```
\tikz \draw (0,0) --(0.5,1) --(2,0.5) rectangle cycle;
```



4.10 Path のオプション

ここでは `\path` に付けられるオプションについて説明する．

- `[draw=色]` ([1]15.3)
  指定した色で path を描く．Ti*k*Z における色の指定では xcolor と同じものが使用できる．

```
\tikz \path[draw=red] (0,0) --(1,0);
```


- `[color=色]` ([1]15.2)
  指定した色を使用する．

```
\tikz \draw[color=red] (0,0) --(1,0);
```

color= は省略できる．

```
\tikz \draw[red] (0,0) --(1,0);
```


- [line width=長さ] ([1]15.3.1)
  線の太さを変更する．初期値は 0.4pt である．

```
\tikz \draw[line width=5pt] (0,0) --(1,0);
```


ちなみにこれを使えば，次の 2 つの書き方には違いがあることが分かる．

```
\begin{tikzpicture}
\draw[line width=5pt] (0,0) --(1,0) --(1,1);
\draw[line width=5pt] (1.5,0) --(2.5,0) (2.5,0) --(2.5,1);
\end{tikzpicture}
```

cycle についても次のような違いがある．

```
\begin{tikzpicture}
\draw[line width=5pt] (0,0) --(1,0) --(0,1) --(0,0);
\draw[line width=5pt] (1.5,0) --(2.5,0) --(1.5,1) --cycle;
\end{tikzpicture}
```

- [ultra thin] ([1]15.3.1)
  [line width=0.1pt] と同じ．
- [very thin] ([1]15.3.1)
  [line width=0.2pt] と同じ．
- [thin] ([1]15.3.1)
  [line width=0.4pt] と同じ．
- [semithick] ([1]15.3.1)
  [line width=0.6pt] と同じ．
- [thick] ([1]15.3.1)
  [line width=0.8pt] と同じ．
- [very thick] ([1]15.3.1)
  [line width=1.2pt] と同じ．
- [ultra thick] ([1]15.3.1)
  [line width=1.6pt] と同じ．
- [line cap=種類] ([1]15.3.1)
  線の先端の部分の形状を変更する．「種類」は round，rect，butt の 3 種類を設定できる．butt が初期値である．設定時に具体的にどうなるかは次の例を参照．

```
\begin{tikzpicture}
\draw[line width=10pt] (0,1.5) --(1,1.5);
\draw[line width=10pt,line cap=round] (0,1) --(1,1);
\draw[line width=10pt,line cap=rect] (0,0.5) --(1,0.5);
\draw[line width=10pt,line cap=butt] (0,0) --(1,0);
\end{tikzpicture}
```

- [line join=種類] ([1]15.3.1)
  折れ線部分の形状を変更する．「種類」は round，bevel，miter の 3 種類を設定できる．miter が初期値である．設定時に具体的にどうなるかは次の例を参照.

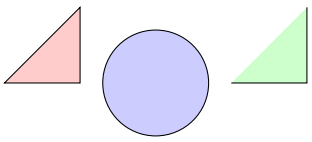
```
\begin{tikzpicture}
\draw[line width=10pt]                    (0,0) --++(0.5,1) --++(0.5,-1);
\draw[line width=10pt,line join=round] (2,0) --++(0.5,1) --++(0.5,-1);
\draw[line width=10pt,line join=bevel] (4,0) --++(0.5,1) --++(0.5,-1);
\draw[line width=10pt,line join=miter] (6,0) --++(0.5,1) --++(0.5,-1);
\end{tikzpicture}
```

- [dash pattern=パターン] ([1]15.3.2)
  点線などを描くためのオプションである．パターン には，例えば on 2pt off 4pt on 8pt off 6pt のような記述をする．これは「2pt 描き 4pt 空け、8pt 描き 6pt 空け」を繰り返す「パターン」である.

```
\tikz \draw[dash pattern=on 2pt off 4pt on 8pt off 6pt]
(0,0) --(30pt,0) --(30pt,6pt) --(0,6pt) --(0,12pt) --(30pt,12pt);
```


- [solid] ([1]15.3.2)
  [dash pattern=] と同じ.
- [dashed] ([1]15.3.2)
  [dash pattern=on 3pt off 3pt] と同じ.
- [densely dashed] ([1]15.3.2)
  [dash pattern=on 3pt off 2pt] と同じ.
- [loosely dashed] ([1]15.3.2)
  [dash pattern=on 3pt off 6pt] と同じ.
- [fill=色] ([1]15.5)
  指定した色で path の「内側」を塗りつぶす (「内側」の意味は以下の例を参照).

```
\begin{tikzpicture}
\draw[fill=red!20] (0,0) --(1,0) --(1,1) -- cycle;
\draw[fill=blue!20] (2,0) circle[radius=0.7];
\draw[fill=green!20] (3,0) --(4,0) --(4,1);
\end{tikzpicture}
```



```
\begin{tikzpicture}
\draw[fill=red!20] (0,0) --(0,1) --(1,1) --(1,0) --cycle
(0.3,0.3) --(0.3,0.7) --(0.7,0.7) --(0.7,0.3) --cycle;
\end{tikzpicture}
```

```
\begin{tikzpicture}
\draw[fill=red!20] (0,0) --(0,1) --(1,1) --(1,0) --cycle
(0.3,0.3) --(0.7,0.3) --(0.7,0.7) --(0.3,0.7) --cycle;
\end{tikzpicture}
```

なお \path[fill] の略記として \fill を，\path[fill,draw] の略記として \filldraw を使う事ができる．ちなみに \filldraw については，順番としてはまず fill で塗りつぶされた後に draw で

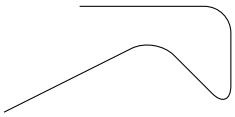
線が描かれることになる．次の例を参照．

```
\tikz \draw[fill=red!20,line width=5pt]
(0,0) --(1,0) --(1,1);
```

- [rounded corners=長さ]

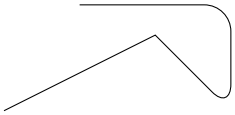
  角を丸くしたい場合に使用する．長さは「丸」の大きさである．4pt が既定値である．

```
\tikz \draw[rounded corners=10pt] (0,0) --(2,1)
--(3,0) --(3,1.4) --(1,1.4);
```



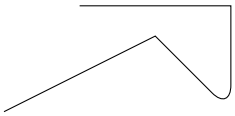
今の例は全ての角が丸くなっているが，次のようにオプションを途中に書くと，その場所以降の角のみ丸くなる．

```
\tikz \draw (0,0) --(2,1)
[rounded corners=10pt] --(3,0) --(3,1.4)
--(1,1.4);
```



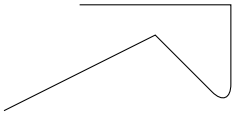
更に途中から元に戻したい場合は [sharp corners] を指定する．

```
\tikz \draw (0,0) --(2,1)
[rounded corners=10pt] -- (3,0)
[sharp corners] --(3,1.4) --(1,1.4);
```



もしくは、{} を使って「スコープ」を指定することもできる．

```
\tikz \draw (0,0) --(2,1)
{[rounded corners=10pt] --(3,0)} --(3,1.4)
--(1,1.4);
```



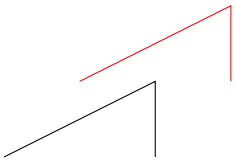
rectangle に対しても有効である．

```
\tikz \draw[rounded corners] (0,0) rectangle(2,1);
```

- [shift=(座標)] ([1]25.3)

  指定した座標の分だけ path 全体を平行移動する．例えば次のように shift={(1,1)} と書いた場合，x 軸方向に 1，y 軸方向に 1 移動する．

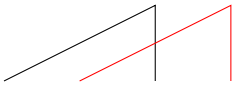
```
\begin{tikzpicture}
\draw                    (0,0) --(2,1) --(2,0);
\draw[red,shift={(1,1)}] (0,0) --(2,1) --(2,0);
\end{tikzpicture}
```



- [xshift=長さ] ([1]25.3)

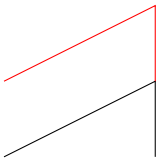
  x 軸方向に平行移動する．

```
\begin{tikzpicture}
\draw                  (0,0) --(2,1) --(2,0);
\draw[red,xshift=1cm] (0,0) --(2,1) --(2,0);
\end{tikzpicture}
```



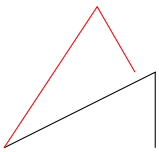
- [yshift=長さ] ([1]25.3)
  y 軸方向に平行移動する．

```
\begin{tikzpicture}
\draw                  (0,0) --(2,1) --(2,0);
\draw[red,yshift=1cm] (0,0) --(2,1) --(2,0);
\end{tikzpicture}
```



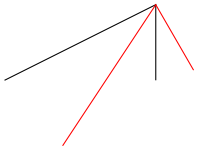
- [rotate=角度] ([1]25.3)
  原点を中心に，指定した 角度 だけ，path を回転させる．

```
\begin{tikzpicture}
\draw                 (0,0) --(2,1) --(2,0);
\draw[red,rotate=30] (0,0) --(2,1) --(2,0);
\end{tikzpicture}
```



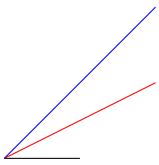
- [rotate around={角度:(座標)}] ([1]25.3)
  指定した 座標 を中心に，指定した 角度 だけ，path を回転させる．

```
\begin{tikzpicture}
\draw                                  (0,0) --(2,1) --(2,0);
\draw[red,rotate around={30:(2,1)}] (0,0) --(2,1) --(2,0);
\end{tikzpicture}
```



- [scale=倍率] ([1]25.3)
  原点を中心に，指定した 倍率 だけ，path を拡大させる．倍率 の絶対値は過度に大きくしたり小さくしたりすべきではないとのこと．
- [scale around={倍率:(座標)}] ([1]25.3) 指定した 座標 を中心に，指定した 倍率 だけ，path を拡大させる．
- [xscale=倍率] ([1]25.3)
- [yscale=倍率] ([1]25.3)
- [x=(座標)] ([1]25.2)
  座標系名 canvas などで使用する，x 軸方向の単位ベクトルを変更する．次の例を参照．

```
\begin{tikzpicture}
\draw                 (0,0) --(1,0);
\draw[x={(2,1)},red]  (0,0) --(1,0);
\draw[x={(2,1)},blue] (0,0) --(1,1);
\end{tikzpicture}
```



  このオプションは [x=長さ] という形式での指定も可能である．この場合 [x={(長さ,0pt)}] と指定するのと同じ意味である．またこのオプションの初期値は [x=1cm] である．
- [y=(座標)] ([1]25.2)
  [x=(座標)] と同様で，y 軸方向の単位ベクトルを変更する．このオプションも [y=長さ] という形式での指定も可能であり，[y={(0pt, 長さ)}] と同じ意味である．またこのオプションの初期値は [y=1cm] である．

- [z=(座標)] ([1]25.2)
  座標系名 xyz などで使用する，z 軸方向の単位ベクトルを変更する．このオプションも [z=長さ] という形式での指定も可能であり，[z={(長さ , 長さ)}] と同じ意味である．またこのオプションの初期値は [z=-3.85mm] である．
- [transform canvas={設定}] ([1]25.4)
- [preaction={設定}] ([1]15.10)
  \path を実行する前に， 設定 に書いた内容で \path を実行する．次の例では，$(0,1)$ から $(1,0)$ へ，白い太線を描いた後で青線を描くので，青線が赤線の上に立体的に交差しているようになる．

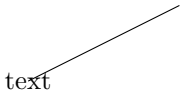
```
\begin{tikzpicture}
\draw[red] (0,0) --(1,1);
\draw[preaction={draw=white,line width=6pt}
     ,blue] (0,1) --(1,0);
\end{tikzpicture}
```

- [postaction={設定}] ([1]15.10)
  preaction と同様であり，\path 本体が実行された後に 設定 の内容が実行される．
- [arrows=開始部分-終了部分]
  path の形状を変更する．詳しくは第 8 節で説明する．

5 The Node Command ([1]17.2)

Ti*k*Z で図式を描く際に，圏論でいう対象にあたるのが node である．node コマンドを使う事で，上で述べてきた path に対して文字を付けたりすることができる．

node {文字} とすることで，現在位置に文字を描くことができる．例えば \draw (1,2) node {text}; とすると，(1,2) の位置に「text」という文字が描かれる．(なお，この文字は draw オプションが無くても描かれるため \path (1,2) node {text}; としてもよい．) もちろん他の，例えば Line-To Operation と組み合わせて使う事もできる．例えば \tikz \draw (0,0) node {text} --(2,1); とすると，$(0,0)$ と $(2,1)$ を繋ぐ線分を描き，更に $(0,0)$ に「text」という文字を描く．



ちなみに {text} と書く代わりに node contents オプションで文字を指定することもできる．例えば先の例では\tikz \draw (0,0) node[node contents=text] --(2,1); と書くことができる．

5.1 Node の形

node には「形」が存在する．node に draw オプションもしくは fill オプションを指定すると見ることができる．例えば \tikz \draw (0,0) node[draw] {text0} (2,0) node[fill=red!20] {text1}; とすると次のようになる．

このように node の形はデフォルトでは rectangle になっている．形を変えるには shape オプションを指定する．通常は次の 3 つが使用できる: shape=rectangle，shape=circle，shape=coordinate．例えば

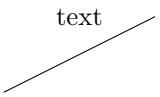
`\tikz \draw (0,0) node[draw,shape=circle] {text0};` とすると次のようになる．

なお `shape=` は省略してもよい．また `\usetikzlibrary{shapes.geometric}` とすることで他にも様々な「形」を使う事ができる ([1]70.3)．`shape=coordinate` については次の節で述べる．

5.2 at オプション

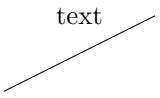
`node` のオプションとして `at` オプションがある．これは node の座標を指定するオプションであり，`at` オプションが指定されている場合はこの座標が優先して使われる．例えば次の例では，node の文字「text」は $(0,0)$ ではなく $(1,1)$ に描かれる．

```
\tikz \draw (0,0) node[at={(1,1)}] {text} --(2,1);
```



また `at` オプションを指定する代わりに，`node at (座標)` と書くこともできる．

```
\tikz \draw (0,0) node at (1,1) {text} --(2,1);
```



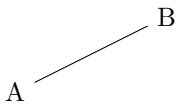
よって, 例えば単に座標 $(1,2)$ に node を置きたい場合は `\path node at (1,2) {text};` と書くことができる．(更に `at` を省略して `\path node (1,2) {text};` と書くこともできる．) また `\node` は `\path node` の略記である．よって `\node at (1,2) {text};` という書き方もできる．

5.3 Node に名前を付ける

node には「名前」を付けて，その位置を後から「名前」で参照することができる．「名前」は `node` の `name` オプションで指定する．例えば `\node[name=hoge] at (1,2) {text};` とすると，この node には `hoge` という「名前」が付く．もしくは `( )` を使って `node (名前)` という形式でも「名前」を付けることができる．例えば先の例では `\node (hoge) at (1,2) {text};` と書いても同じである．

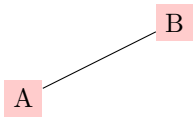
参照するときには，座標系名「`node`」を使用し `(node cs:name=名前)` と書く．もしくは省略して単に `(名前)` と書くこともできる．例えば次のようになる．

```
\begin{tikzpicture}
\node (hoge) at (0,0) {A};
\node (fuga) at (2,1) {B};
\draw (node cs:name=hoge) --(node cs:name=fuga);
\end{tikzpicture}
```



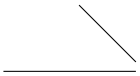
見てなんとなくわかる通り，この場合 node の「形」に合わせて線が引かれる．次のように `node` に `fill` を指定してみると分かる．(以降では座標系名 `node` の省略形を使用する．)

```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (hoge) --(fuga);
\end{tikzpicture}
```



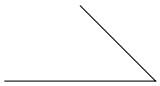
これはこれで便利であるが，node に名前だけ付けて文字は書きたくない場合，困る場合がある．例えば次の例を考える．

```
\begin{tikzpicture}
\node (a0) at (0,0) {};
\node (a1) at (2,0) {};
\node (a2) at (1,1) {};
\draw (a0) --(a1) --(a2);
\end{tikzpicture}
```



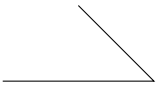
3 点に名前を付け，折れ線を描いたつもりが，node に「大きさ」があるため線が途切れてしまう．このような場合に使えるのが shape=coordinate である．次のように shape=coordinate を指定すると次のようになる (shape= は省略可能であったことに注意).

```
\begin{tikzpicture}
\node[coordinate] (a0) at (0,0) {};
\node[coordinate] (a1) at (2,0) {};
\node[coordinate] (a2) at (1,1) {};
\draw (a0) --(a1) --(a2);
\end{tikzpicture}
```



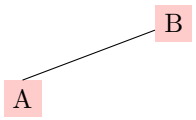
さて，実は \node[coordinate] (名前) at (座標) {}; の略記として coordinate (名前) at (座標) を使う事ができる．更に \path coordinate の略記として \coordinate を使う事ができる．よって上記の例は次のように書くことができる．

```
\begin{tikzpicture}
\coordinate (a0) at (0,0);
\coordinate (a1) at (2,0);
\coordinate (a2) at (1,1);
\draw (a0) --(a1) --(a2);
\end{tikzpicture}
```

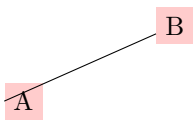


node には大きさがあり，「名前」により参照するときも「場所」を指定することができる．その方法は 2 つあり，一つは anchor である．

```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (node cs:name=hoge,anchor=north)
    --(node cs:name=fuga);
\end{tikzpicture}
```

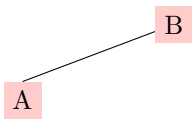



```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (node cs:name=hoge,anchor=west)
    --(node cs:name=fuga);
\end{tikzpicture}
```

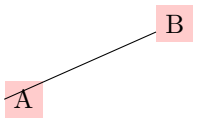


もう一つは angle である．

```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (node cs:name=hoge,angle=90)
    --(node cs:name=fuga);
\end{tikzpicture}
```

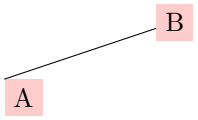


```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (node cs:name=hoge,angle=180)
    --(node cs:name=fuga);
\end{tikzpicture}
```

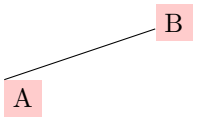


またこれも略記法があり，(node cs:name=hoge,anchor=north) や (node cs:name=hoge,angle=90) はそれぞれは (hoge.north)，(hoge.90) と書くことができる．

```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (hoge.north west) --(fuga);
\end{tikzpicture}
```



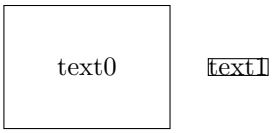

```
\begin{tikzpicture}
\node[fill=red!20] (hoge) at (0,0) {A};
\node[fill=red!20] (fuga) at (2,1) {B};
\draw (hoge.135) --(fuga);
\end{tikzpicture}
```



5.4 Node のオプション

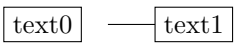
- [inner sep=長さ] ([1]17.2.3)
  node の内側の余白の大きさを設定する (CSS でいう padding である)．初期値は 0.3333em になっている．

```
\begin{tikzpicture}
\node[draw,inner sep=20pt] at (0,0) {text0};
\node[draw,inner sep=0pt]  at (2,0) {text1};
\end{tikzpicture}
```



- [inner xsep=長さ] ([1]17.2.3)
  node の内側の余白の大きさのうち，x 軸方向のみを設定する．
- [inner ysep=長さ] ([1]17.2.3)
  node の内側の余白の大きさのうち，y 軸方向のみを設定する．
- [outer sep=長さ] ([1]17.2.3)
  node の外側の余白の大きさを設定する (CSS でいう margin である)．

```
\begin{tikzpicture}
\node[draw,outer sep=10pt] (a) at (0,0) {text0};
\node[draw,outer sep=0pt]  (b) at (2,0) {text1};
\draw (a) --(b);
\end{tikzpicture}
```



- [outer xsep=長さ] ([1]17.2.3)
  node の外側の余白の大きさのうち，x 軸方向のみを設定する．
- [outer ysep=長さ] ([1]17.2.3)
  node の外側の余白の大きさのうち，y 軸方向のみを設定する．

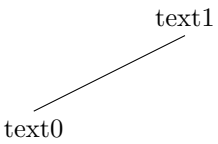
- [minimum height=長さ] ([1]17.2.3)
  (inner sep を含めた) node の高さの最小値を設定する．初期値は 0pt になっている．
- [minimum width=長さ] ([1]17.2.3)
  (inner sep を含めた) node の幅の最小値を設定する．初期値は 0pt になっている．
- [minimum size=長さ] ([1]17.2.3)
  minimum height と minimum width に同じ値を設定する場合に使用できる．
- [behind path] ([1]17.2.1)
  文字を path の下に描くようにする．

```
\tikz \draw[draw=red!20,line width=10pt] (0,0)
node[behind path] {text0} --(1,0) node {text1};
```

tex text1

- [in front of path] ([1]17.2.1)
  文字を path の上に描くようにする．
- [text=色] ([1]17.4.1)
  文字の色を指定する．
- [text width=長さ] ([1]17.4.3)
- [align=値] ([1]17.4.3)
- [anchor=位置] ([1]17.5.1)
  文字の位置を指定する．例えば anchor=north とすると，node の上の部分の点が，node の座標の位置になる．位置 には north, south, east, west, north east, north west, south east, south west などが使用できる．

```
\tikz \draw (0,0) node[anchor=north] {text0}
--(2,1) node[anchor=south] {text1};
```



center, base などの指定もできる．

```
\tikz \draw (0,0) node[anchor=center] {x}
--(1,0) node[anchor=center] {y};
```

x y

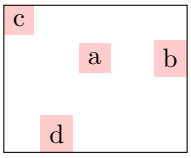
```
\tikz \draw (0,0) node[anchor=base] {x}
--(1,0) node[anchor=base] {y};
```

x y

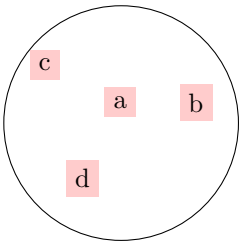
- [above=長さ] ([1]17.5.2)
  既定値は 0pt であり，この場合 anchor=south と同じである．長さ を指定した場合は，更にその分だけ上に移動する．
- [below=長さ] ([1]17.5.2)
  above と同様.
- [left=長さ] ([1]17.5.2)
  above と同様.
- [right=長さ] ([1]17.5.2)
  above と同様.
- [above left] ([1]17.5.2)

- [above right] ([1]17.5.2)
- [below left] ([1]17.5.2)
- [below right] ([1]17.5.2)
- [centered] ([1]17.5.2)
- [fit=複数座標] ([1]17.6, 54) (\usetikzlibrary{fit} が必要)
  node の形が，指定した座標を全て含むように作られる．

```
\begin{tikzpicture}
\node[fill=red!20] (a) at (0,0) {a};
\node[fill=red!20] (b) at (1,0) {b};
\node[fill=red!20] (c) at (-1,0.5) {c};
\node[fill=red!20] (d) at (-0.5,-1) {d};
\node[draw,inner sep=0pt,fit=(a)(b)(c)(d)] {};
\end{tikzpicture}
```



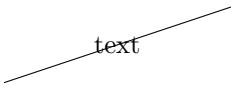

```
\begin{tikzpicture}
\node[fill=red!20] (a) at (0,0) {a};
\node[fill=red!20] (b) at (1,0) {b};
\node[fill=red!20] (c) at (-1,0.5) {c};
\node[fill=red!20] (d) at (-0.5,-1) {d};
\node[draw,shape=circle,inner sep=0pt,
      fit=(a)(b)(c)(d)] {};
\end{tikzpicture}
```



- [pos=割合] ([1]17.8)
  まず，普通に \tikz \draw (0,0) --(3,1) node {text}; とすると (3,1) に text と描かれる．

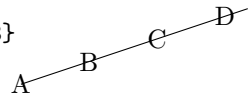
ここで node にオプション pos=0.5 と指定すると，(0,0) と (3,1) の中間に text と描くことができる．

```
\tikz \draw (0,0) --(3,1) node[pos=0.5] {text};
```



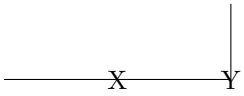
pos オプションには 0〜1 を指定することができ，0 が始点の位置，1 が終点の位置である．

```
\tikz \draw (0,0) --(3,1) node[pos=0] {A} node[pos=0.3] {B}
node[pos=0.6] {C} node[pos=0.9] {D};
```



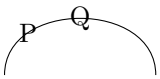
-| や |- の場合でも pos オプションは使用できる．この場合，角の部分が 0.5 の位置になる．

```
\tikz \draw (0,0) -|(3,1)
node[pos=0.25] {X} node[pos=0.5] {Y};
```



Curve-To Operation などでも使用できる．

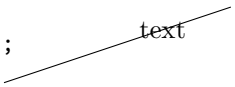
```
\tikz \draw (0,0) .. controls (0,2) and (3,2) .. (3,0)
node[pos=0.25] {P} node[pos=0.5] {Q};
```



`\draw (0,0) --(3,1) node[pos=0.5] {text};` という書き方はかなり不自然な気がするが，実は同じことを `\draw (0,0) -- node {text} (3,1);` と書くことができる．

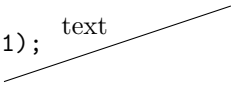
またこの場合でも pos オプションは使用できる．

```
\tikz \draw (0,0) -- node[pos=0.7] {text} (3,1);
```



- `[at start]` ([1]17.8)
  `[pos=0]` と同じ．
- `[very near start]` ([1]17.8)
  `[pos=0.125]` と同じ．
- `[near start]` ([1]17.8)
  `[pos=0.25]` と同じ．
- `[midway]` ([1]17.8)
  `[pos=0.5]` と同じ．
- `[near end]` ([1]17.8)
  `[pos=0.75]` と同じ．
- `[very near end]` ([1]17.8)
  `[pos=0.825]` と同じ．
- `[at end]` ([1]17.8)
  `[pos=1]` と同じ．
- `[auto=値]` ([1]17.8)
  このオプションは node が直線や曲線上にある場合のみ有効である．値 には `left`, `right`, `false` のいずれかを設定する．`[auto=left]` とした場合，node が線の進行方向の左側に配置される．`[auto=right]` の場合は右になる．`[auto=false]` の場合はこの制御が無効になる．

```
\tikz \draw (0,0) -- node[auto=left] {text} (3,1);
```



- `[swap]` ([1]17.8)
  `auto` の設定を `left` と `right` で入れ替える．

5.5 Node に複数行の文字を書く ([1]17.4.3)

node に複数行の文字を書きたい場合は以下のようにする．

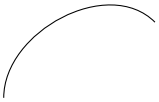
(1) node contents に通常複数行表示に使用する環境 (例えば tabular 環境) を置く．
(2) `\\`を使い改行する．なお `align` オプションを指定することができる．
(3) `text width` オプションを指定する．改行は幅に合わせて自動で行われる．

6 The To Operation ([1]14.13)

今まで (0,0) --(2,1) のような書き方をしてきたが，これは (0,0) to (2,1) と書くこともできる．この二つの違いは，to にはオプションを付けられることである．オプションには以下のようなものがある．

- [out=角度] ([1]73.3)
  path が始点から出るときの方向を角度で指定することができる (path は勝手に適当に曲がる)．例えば次のように [out=90] とすると，path が始点から真上に出る．

```
\tikz \draw (0,0) to[out=90] (2,1);
```



- [in=角度] ([1]73.3)
  out と同様で，終点に入るときの角度を指定する．

```
\tikz \draw (0,0) to[out=90,in=225] (2,1);
```

- [relative=真偽値] ([1]73.3)
  out や in における角度の指定を「絶対的」にするか「相対的」にするかを設定する．[relative=true] の場合が「相対的」で，[relative=false] の場合が「絶対的」である．既定値は true である．

```
\tikz \draw (0,0) to[out=90,in=225,relative] (2,1);
```

- [vend left=角度] ([1]73.3)
  [out=角度,in=(180-角度),relative] の略記である．

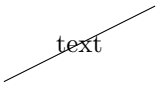
```
\tikz \draw (0,0) to[bend left=70] (0,2);
```

- [vend right=角度] ([1]73.3)
  vend left と同様．
- [looseness=数値] ([1]73.3)
  「looseness」を設定する．初期値は 1 である．次の例を参照．

```
\begin{tikzpicture}
\draw (0,0) to[out=0,in=-90] ++(1,1);
\draw (2,0) to[out=0,in=-90,looseness=0.5] ++(1,1);
\draw (4,0) to[out=0,in=-90,looseness=2] ++(1,1);
\end{tikzpicture}
```

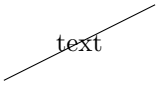
- [edge node={ノード}] ([1]14.13)
  to についても Line-To Operation と同様，to の後に node を書くことができる．

```
\tikz \draw (0,0) to node {text} (2,1);
```



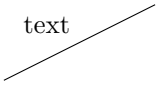
この代わりに，edge node オプションを使って，to のオプションとして node を指定することができる．

```
\tikz \draw (0,0) to[edge node={node {text}}] (2,1);
```



- [edge label=文字] ([1]14.13)
  [edge node={node[auto]{文字}}] の略記である．

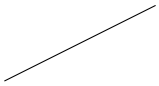
```
\tikz \draw (0,0) to[edge label=text] (2,1);
```



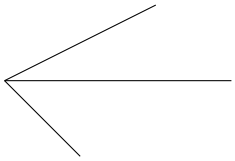
- [edge label'=文字] ([1]14.13)
  [edge node={node[auto,swap]{文字}}] の略記である．

7 The Edge Operation ([1]17.12)

\draw (0,0) edge (2,1); と書くことでも線を引くことができる．



to との違いは「現在位置」が移動しないことで，\draw (0,0) edge (2,1) edge (3,0) edge (1,-1); のような書き方ができる．



8 Arrow Tips

arrows オプションを指定することで path を「矢印」にすることができる．arrows オプションは [arrows=開始部分-終了部分] の形式で指定する (開始部分/終了部分は空文字列にすることも可能)．例えば \draw[arrows=|->] (0,0) -- (2,0); とすると次のようになる．

但し，arrows オプションについては arrows= を省略することが可能である (- を含むオプションは arrows オプションと解釈される)．例えば先の例では \draw[|->] (0,0) -- (2,0); としてもよい．

開始部分/終了部分には次のようなものが使用できる．

 \draw[stealth-stealth] (0,0) --(1,0);

 \draw[stealth reversed-stealth reversed] (0,0) --(1,0);

 \draw[to-to] (0,0) --(1,0);

 \draw[to reversed-to reversed] (0,0) --(1,0);

 \draw[latex-latex] (0,0) --(1,0);

 \draw[latex reversed-latex reversed] (0,0) --(1,0);

⟼ \draw[|-|] (0,0) --(1,0);

⟷ \draw[<->] (0,0) --(1,0);

⤚ \draw[>-<] (0,0) --(1,0);

更に \usetikzlibrary{arrows.meta} とする[*1]ことで，以下のようなものも使う事ができる (下記以外にも色々ある [1]16.5 を参照).

⥊ \draw[Arc Barb-Arc Barb] (0,0) --(1,0);

⟼ \draw[Bracket-Bracket] (0,0) --(1,0);

↬ \draw[Hooks-Hooks] (0,0) --(1,0);

↪ \draw[{Hooks[left]}-{Hooks[left]}] (0,0) --(1,0);

↩ \draw[{Hooks[right]}-{Hooks[right]}] (0,0) --(1,0);

⟷ \draw[Straight Barb-Straight Barb] (0,0) --(1,0);

⟼ \draw[Tee Barb-Tee Barb] (0,0) --(1,0);

⟷ \draw[Classical TikZ Rightarrow-Classical TikZ Rightarrow] (0,0) --(1,0);

⟷ \draw[Computer Modern Rightarrow-Computer Modern Rightarrow] (0,0) --(1,0);

● \draw[Circle-Circle] (0,0) --(1,0);

◆ \draw[Diamond-Diamond] (0,0) --(1,0);

開始部分/終了部分には複数指定することもできる.

↠ \draw[->>] (0,0) --(1,0);

⇛ \draw[->>>] (0,0) --(1,0);

⟹ \draw[->>>>] (0,0) --(1,0);

この場合，. を使う事で，path を途中で止めることができる.

↠ \draw[->.>>] (0,0) --(1,0);

⇛ \draw[->>.>] (0,0) --(1,0);

注: 場合によっては開始部分/終了部分を中括弧で囲わないと正常に動作しない場合がある (例: \draw[-{Stealth[red]}] (0,0) --(1,0);). 難しい場合はオプションを arrows={開始部分-終了部分} の形式で書けば，常に正常に動作する. ([1]16.2)

9 tikzpicture 環境のオプション

- [baseline=長さ] ([1]12.2.1)
  tikzpicture 環境の「baseline」を指定した高さにする. なお [2] も参照.
- [baseline=(座標)] ([1]12.2.1)
  tikzpicture 環境の「baseline」を指定した座標の高さにする. なお [2] も参照.

[*1] arrows や arrows.spaced というのもあるが，これらは古く非推奨となっている ([1]42).

10　オプション指定時に使える便利な方法

例えば赤く太い矢印を複数描く場合，次のように同じオプション指定を何度も書くことになってしまう．

```
\begin{tikzpicture}
\draw[->,color=red,thick] (0,0) --(15:1);
\draw[->,color=red,thick] (0,0) --(75:1);
\draw[->,color=red,thick] (0,0) --(135:1);
\end{tikzpicture}
```

このような場合に使える便利な方法がいくつかある．

10.1　スタイル　([1]12.4.1, 86.4.4)

「スタイル」とは，いくつかのオプション設定にまとめて名前をつけるようなものである．**スタイル名/.style={設定内容}** と書くことで，**設定内容** に **スタイル名** という名前を付けることができ，以降はオプションに **スタイル名** と書くだけで **設定内容** が反映される．例えば次の例では，`->,color=red,thick` に `red arrow` という名前を付けているので，`\draw` のオプションに単に `[red arrow]` と書くだけで赤く太い矢印にすることができる．

```
\begin{tikzpicture}[red arrow/.style={->,color=red,thick}]
\draw[red arrow] (0,0) --(15:1);
\draw[red arrow] (0,0) --(75:1);
\draw[red arrow] (0,0) --(135:1);
\end{tikzpicture}
```

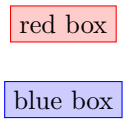
但しこの書き方では，`red arrow` を使えるのはこの `tikzpicture` 環境だけである．他の `tikzpicture` 環境でも使用できるようにしたい場合は `\tikzset` 命令を使う．

```
\tikzset{red arrow/.style={->,color=red,thick}}

\begin{tikzpicture}
\draw[red arrow] (0,0) --(15:1);
\draw[red arrow] (0,0) --(75:1);
\draw[red arrow] (0,0) --(135:1);
\end{tikzpicture}
```

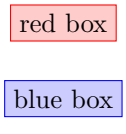
スタイルでは引数を 1 つ使うことができる．その場合は通常の TEX のように `#1` を使う．引数に値を入れるには，オプションに指定する際に **スタイル名=値** のように書く．

```
\begin{tikzpicture}[outline/.style={draw=#1,fill=#1!20}]
\node[outline=red]  at (0,1) {red box};
\node[outline=blue] at (0,0) {blue box};
\end{tikzpicture}
```



更にこの場合，既定値を設定することもできる．その場合は **スタイル名/.default=既定値** と書く．

```
\begin{tikzpicture}[outline/.style={draw=#1,fill=#1!20},
outline/.default=red]
\node[outline]      at (0,1) {red box};
\node[outline=blue] at (0,0) {blue box};
\end{tikzpicture}
```



10.2 スコープ ([1]12.3)

TikZ では「外側」に指定されたオプションは「内側」のそれぞれの要素に適用されるようになっている．そこで次のように，path のオプションを tikzpicture 環境のオプションとして指定するということができる．

```
\begin{tikzpicture}[->,color=red,thick]
\draw (0,0) --(15:1);
\draw (0,0) --(75:1);
\draw (0,0) --(135:1);
\end{tikzpicture}
```

但し，この方法では全体に同じオプションが設定されてしまう．そこで使えるのが scope 環境である．

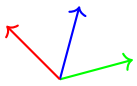
```
\begin{tikzpicture}[->,thick]
\begin{scope}[color=red]
\draw (0,0) --(15:1);
\draw (0,0) --(75:1);
\end{scope}
\draw (0,0) --(135:1);
\end{tikzpicture}
```

更に \usetikzlibrary{scopes} を使うと\begin{scope}, \end{scope} を単に {, } と書けるようになる．

```
\begin{tikzpicture}[->,thick]
{[color=red]
\draw (0,0) --(15:1);
\draw (0,0) --(75:1);
}
\draw (0,0) --(135:1);
\end{tikzpicture}
```

なおこの設定方法では，より「内側」の設定の方が優先される．

```
\begin{tikzpicture}[->,thick,color=red]
\begin{scope}[color=blue]
\draw[color=green] (0,0) --(15:1);
\draw (0,0) --(75:1);
\end{scope}
\draw (0,0) --(135:1);
\end{tikzpicture}
```

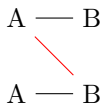


また scope 環境に指定するオプションもある．

- [name prefix=名前] ([1]17.2.1)

次の例を参照.

```
\begin{tikzpicture}
\begin{scope}[name prefix=top-]
\node (A) at (0,1) {A};
\node (B) at (1,1) {B};
\draw (A) --(B);
\end{scope}
\begin{scope}[name prefix=bottom-]
\node (A) at (0,0) {A};
\node (B) at (1,0) {B};
\draw (A) --(B);
\end{scope}
\draw[red] (top-A) --(bottom-B);
\end{tikzpicture}
```



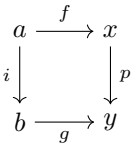
10.3 every

- [every picture/.style={設定}] ([1]12.2.1)
  全ての tikzpicture 環境に 設定 を適用する. これは \tikzset で使用する. 例えば全ての path に [line width=1pt] を適用したい場合は\tikzset{every picture/.style={line width=1pt}} と書く.
- [every scope/.style={設定}] ([1]12.3.1)
  全ての scope 環境に 設定 を適用する.
- [every path/.style={設定}] ([1]14)
  全ての \path に 設定 を適用する.
- [every circle/.style={設定}] ([1]14.6)
  全ての circle に 設定 を適用する.
- [every to/.style={設定}] ([1]14.13)
  全ての to に 設定 を適用する.
- [every node/.style={設定}] ([1]17.2.1)
  全ての node に 設定 を適用する.
- [every rectangle node/.style={設定}] ([1]17.2.1)
  shape=rectangle となっている全ての node に 設定 を適用する.
- [every circle node/.style={設定}] ([1]17.2.1)
  shape=circle となっている全ての node に 設定 を適用する.
- [every edge/.style={設定}] ([1]17.12.1)
  全ての edge に 設定 を適用する.

11　圏論で図式を描く例

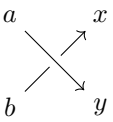
次は普通の四角い可換図式の例である.

```
\begin{tikzpicture}[auto,->]
\node (a) at (0,1.2) {$a$}; \node (x) at (1.2,1.2) {$x$};
\node (b) at (0,0) {$b$}; \node (y) at (1.2,0) {$y$};
\draw (a) -- node {$\scriptstyle f$} (x);
\draw (x) -- node {$\scriptstyle p$} (y);
\draw (a) -- node[swap] {$\scriptstyle i$} (b);
\draw (b) -- node[swap] {$\scriptstyle g$} (y);
\end{tikzpicture}
```



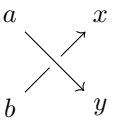
線を立体的に交差させたい場合は，次のように白い太線を使えばよい.

```
\begin{tikzpicture}
\node (a) at (0,1.2) {$a$}; \node (x) at (1.2,1.2) {$x$};
\node (b) at (0,0) {$b$}; \node (y) at (1.2,0) {$y$};
\draw[->] (b) --(x);
\draw[white,line width=6pt] (a) --(y);
\draw[->] (a) --(y);
\end{tikzpicture}
```



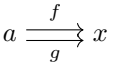
これは preaction を使うと次のように書ける.

```
\tikzset{cross/.style={preaction={draw=white,line width=6pt}}}
\begin{tikzpicture}
\node (a) at (0,1.2) {$a$}; \node (x) at (1.2,1.2) {$x$};
\node (b) at (0,0) {$b$}; \node (y) at (1.2,0) {$y$};
\draw[->] (b) --(x);
\draw[cross,->] (a) --(y);
\end{tikzpicture}
```



equalizer の時に使うような，平行射を描きたい場合は transform canvas を使うと上手くいく.

```
\begin{tikzpicture}[auto,->]
\node (a) at (0,1.2) {$a$}; \node (x) at (1.2,1.2) {$x$};
\draw[transform canvas={yshift=2pt}]
    (a) -- node {$\scriptstyle f$} (x);
\draw[transform canvas={yshift=-2pt}]
    (a) -- node[swap] {$\scriptstyle g$} (x);
\end{tikzpicture}
```



12　圏論では絶対に使わないと思うけど凄い機能

12.1　remember picture オプション

※ この機能は環境に依存しており，例えば upLaTeX+dvipdfmx の場合は pxpgfmark パッケージが必要. [3] を参照のこと.

まず tikzpicture 環境に remember picture オプションを指定して

と書くとこのように
```
\begin{tikzpicture}[remember picture,baseline=(node1.base)]
\node[draw,fill=red!20] (node1) {A};
\end{tikzpicture}
```
本文中の赤い四角の中に A と描かれる．

と書くとこのように A 本文中の赤い四角の中に A と描かれる．次に，同じように

と書くと
```
\begin{tikzpicture}[remember picture,baseline=(node2.base)]
\node[draw,fill=blue!20] (node2) {X};
\end{tikzpicture}
```
同様に青い四角が描かれる．

と書くと X 同様に青い四角が描かれる．

この remember picture オプション有りで作った node の名前は別の tikzpicture 環境で参照することができる (このとき overlay オプションが必要)．例えば次のように

```
\begin{tikzpicture}[remember picture,overlay]
\draw[->,thick,green!60!black] (node1) --(node2);
\end{tikzpicture}
```

と書くと，上の四角について path が描かれる．

参考文献

[1] PGF manual, Manual for Version 3.1, https://ctan.org/pkg/pgf

[2] @zr_tex8r, TikZ で“インラインな”図を描く, https://qiita.com/zr_tex8r/items/61d14cec3f54972578ea

[3] @zr_tex8r, TikZ は dvipdfmx をどこまでサポートするか？ (2), https://zrbabbler.hatenablog.com/entry/20130302/1362228901