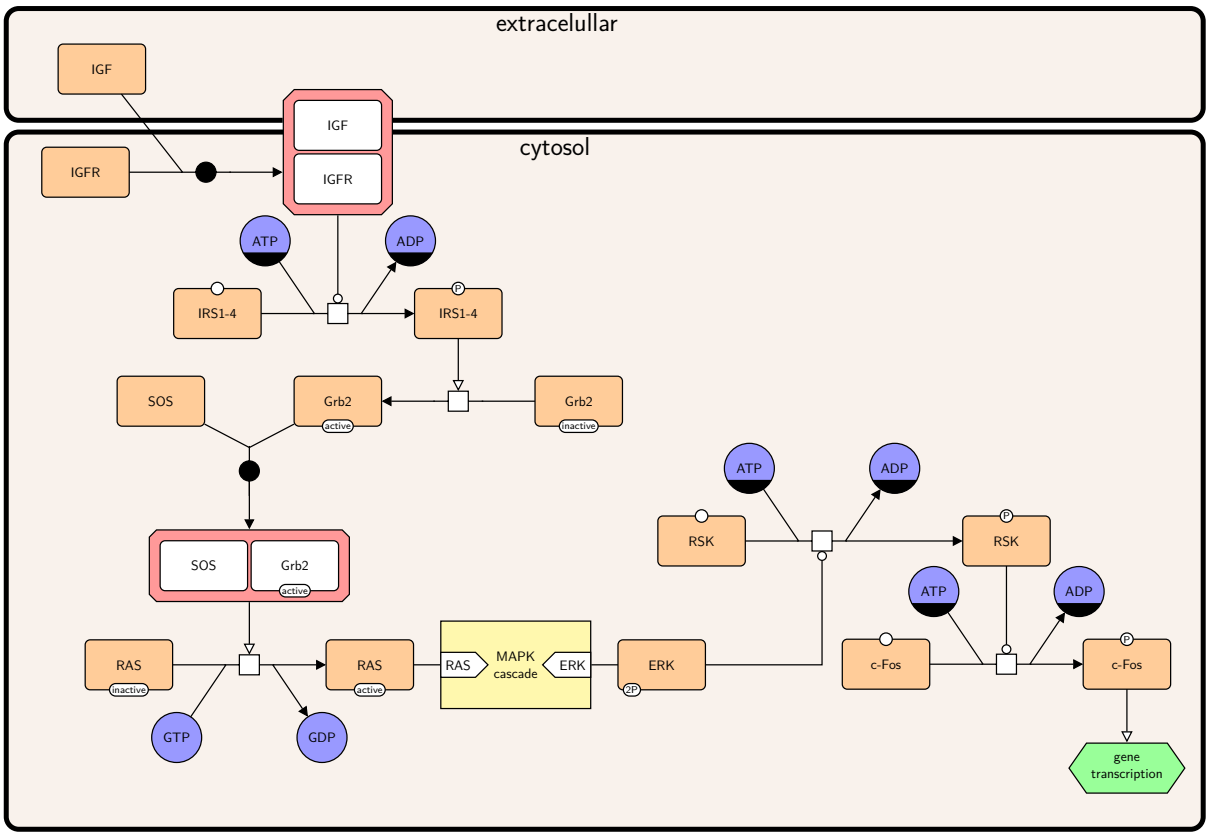
sbgntikz

manual for version 0.1

Adrien Rougny

May 10, 2018



Map adapted from SBGN PD language L1V1.3, Journal of Integrative Bioinformatics, 2015 Sep 4;12(2):263

Contents

1 Introduction

1.1 About

sbgntikz is a Ti*k*Z [1] library to draw SBGN PD, AF or ER maps [2] directly into LaTeX documents. It basically encodes SBGN glyphs into Ti*k*Z shapes and arrowheads named by keywords, making them straightforwardly drawable within a Ti*k*Z picture. Drawing a specific glyph is then as simple as specifying its corresponding keyword in some Ti*k*Z command.

The present manual is intended for an audience that knows SBGN but not particularly Ti*k*Z. The rest of the present section is dedicated to the first steps in using *sbgntikz*: installing the library and drawing a first map (while introducing some basic Ti*k*Z syntax). Section 2 references all glyphs and their associated keywords, whereas section 3 gives some Ti*k*Z options and syntaxes that I find most useful to draw SBGN maps. I believe users already familiar with Ti*k*Z will mostly be interested in reading section 2, and might have different (and maybe better) solutions to the issues presented in section 3.

1.2 Installation and usage

The directory `tikz-sbgn/` should be copied to a directory where it can be found by the TeX engine:

- in the directory of your LaTeX source file
- in your local `texmf` directory. This directory is usually `/home/<user>/texmf/` under Linux, `/Users/Library/texmf/` under MacOS, and `C:/Users/<user>/texmf` under Windows, but it can depend on your OS version and TeX distribution.

Your `texmf` directory can be found using the `kpsewhich -var-value=TEXMFHOME` command.

Usually, Ti*k*Z is installed within your TEX distribution, so Ti*k*Z and *sbgntikz* can be imported directly into your LATEX source file with no further installation adding the following two commands to your preamble:

```
\usepackage{tikz}
\usetikzlibrary{sbgn}
```

An SBGN map can then be drawn within a Ti*k*Z picture using the `sbgn` key:

```
\begin{tikzpicture}[sbgn]
% tikz code to draw an SBGN map
\end{tikzpicture}
```

1.3 A first map

SBGN is all about drawing nodes with specific shapes and arcs with specific arrow heads. Fortunately, drawing Ti*k*Z pictures is not different, making it pretty straightforward to draw SBGN maps using *sbgntikz*: the `\node` command is used to draw nodes, while the `\draw` command is used to draw arcs. The code to draw an SBGN node (or an attribute) will usually look like the following:

```
\node[<sbgn node>, ...] (name) at (point) {LABEL};
```

where

- `<sbgn node>` is a keyword corresponding to the type of node to be drawn (e.g. `simple chemical` for a simple chemical);
- `...` is a list of other options for the node (e.g. its relative positioning towards another node, color, line width …);
- `(name)` specifies the name of the node (optional);
- `at (point)` specifies the point on the canvas where to draw the node (optional, by default `(0,0)` if no relative positioning is specified in the nodes' options);
- `{LABEL}` specifies the label of the node that will be displayed (mandatory but can be empty).

As for arcs, they can be drawn using the following piece of code:

```
\draw[<sbgn arc>, ...] (a) -- (b);
```

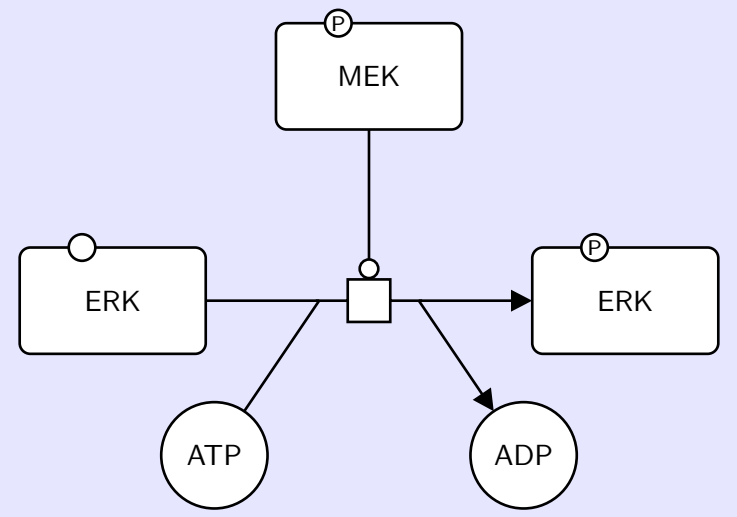
where

- `<sbgn arc>` is a keyword corresponding to the type of arc to be drawn (e.g. `necessary stimulation` for a necessary stimulation);
- `...` is a list of other options for the arc (e.g. its color, line width …);
- `(a)` is a point on the canvas or the name of a node from which the arc will depart;
- `(b)` is a point on the canvas or the name of a node on which the arc will arrive.

Knowing those two basic syntaxes, one can draw pretty much any desired SBGN map. Following is an example of code to draw a simple PD map. It relies on relative positioning provided by Ti*k*Z's `positioning` library, as positioning all nodes with absolute coordinates would be too cumbersome (see section 3 for few more details, or the PGF/Ti*k*Z manual for lot more details).

```
\documentclass{standalone}

\usepackage{tikz}
\usetikzlibrary{positioning, sbgn}

\begin{document}

\begin{tikzpicture}[sbgn]
\node[macromolecule] (erk) {ERK};   % this node has no absolute nor relative positioning, so it
↪  is placed at (0,0) by default
\node[sv] at (erk.120) {};  % the state variable is placed on the border of the node, at an angle
↪  of 120 deg
\node[generic process, connectors = horizontal, right = of erk] (p) {}; % we add connectors, and
↪  use relative positioning
\node[macromolecule, right = of p] (perk) {ERK};
\node[sv] at (perk.120) {P};
\node[simple chemical, below left = of p] (atp) {ATP};
\node[simple chemical, below right = of p] (adp) {ADP};
\node[macromolecule, above = 2cm of p] (pmek) {MEK};
\node[sv] at (pmek.120) {P};

\draw[consumption] (erk) -- (p.west);   % p being the name of the process node, p.west is the tip
↪  of its left connector
\draw[consumption] (atp) -- (p.west);
\draw[production] (p.east) -- (perk);
\draw[production] (p.east) -- (adp);
\draw[catalysis] (pmek) -- (p);
\end{tikzpicture}

\end{document}
```

Compiling the above code would produce the following figure:



2 Drawing glyphs

2.1 Nodes

Macromolecule (PD) /tikz/macromolecule


```
\node[macromolecule] {LABEL};
```

Macromolecule multimer (PD) /tikz/macromolecule multimer


```
\node[macromolecule multimer] {LABEL};
```

Nucleic acid feature (PD) /tikz/nucleic acid feature


```
\node[nucleic acid feature] {LABEL};
```

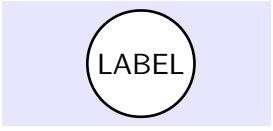
Nucleic acid feature multimer (PD) /tikz/nucleic acid feature multimer


```
\node[nucleic acid feature multimer] {LABEL};
```

Unspecified entity (PD) /tikz/unspecified entity


```
\node[unspecified entity] {LABEL};
```

Simple chemical (PD) /tikz/simple chemical




```
\node[simple chemical] {LABEL};
```

Simple chemical multimer (PD) /tikz/simple chemical multimer


```
\node[simple chemical multimer] {LABEL};
```

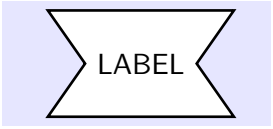
Complex (PD) /tikz/complex


```
\node[complex] {LABEL};
```

Complex multimer (PD) /tikz/complex multimer

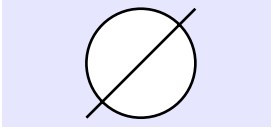

```
\node[complex multimer] {LABEL};
```

Perturbation (PD) /tikz/perturbation




```
\node[perturbation] {LABEL};
```

Emptyset (PD) /tikz/emptyset



```
\node[empty set] {};
```

Biological activity (AF) /tikz/biological activity


```
\node[biological activity] {LABEL};
```

Entity (ER) /tikz/entity


```
\node[entity] {LABEL};
```

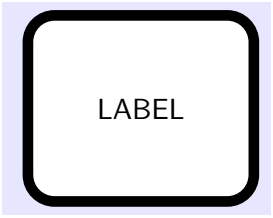
Outcome (ER) /tikz/outcome


```
\node[outcome] {};
```

Value (ER) /tikz/value


```
\node[value] {val};
```

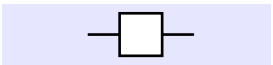
Compartment (PD, AF, ER) /tikz/compartment



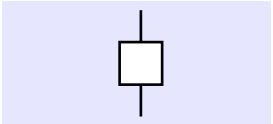

```
\node[compartment, shape=rectangle, minimum
↪  width=85pt, minimum height=70pt, rounded
↪  corners=10pt] {LABEL};
```

Connectors (PD) /tikz/connectors={vertical, horizontal}

Vertical or horizontal connectors can be added to process nodes and logical operators.

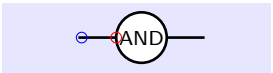


```
\node[generic process, connectors=horizontal]
↪  {};
```



```
\node[generic process, connectors=vertical] {};
```

Cardinal anchors are set to the tip of the connectors. Other border anchors are not changed.




```
\node[and, connectors=horizontal] (o) {};
\path[draw=blue] (o.west) circle[radius=2pt];
\path[draw=red] (o.180) circle[radius=2pt];
```

Clone (PD) /tikz/clone


```
\node[unspecified entity, clone] {LABEL};
```

Generic process (PD) /tikz/generic process

```
\node[generic process] {};
```

Omitted process (PD) /tikz/omitted process

```
\node[omitted process] {};
```

Unknown process (PD) /tikz/unknown process


```
\node[unknown process] {};
```

Association (PD) /tikz/association

```
\node[association] {};
```

Dissociation (PD) /tikz/dissociation

```
\node[dissociation] {};
```

Phenotype (PD, AF, ER) /tikz/phenotype


```
\node[phenotype] {LABEL};
```

And (PD, AF, ER) /tikz/and


```
\node[and] {};
```

Or (PD, AF, ER) /tikz/or


```
\node[or] {};
```

Not (PD, AF, ER) /tikz/not


```
\node[not] {};
```

Delay (PD, AF, ER) /tikz/delay


```
\node[delay] {};
```

Submap (PD) /tikz/submap


```
\node[submap, minimum width=85pt, minimum
↪  height=70pt] {LABEL};
```

Tag (PD) /tikz/tag


```
\node[tag] {LABEL};
```

2.2 Arcs

Logic arc (PD, AF, ER) /tikz/logic arc


```
\draw[logic arc] (0,0) -- (2cm,0);
```

Equivalence arc (PD) /tikz/equivalence arc


```
\draw[equivalence arc] (0,0) -- (2cm,0);
```

Consumption (PD) /tikz/consumption


```
\draw[consumption] (0,0) -- (2cm,0);
```

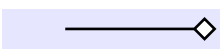
Production (PD) /tikz/production


```
\draw[production] (0,0) -- (2cm,0);
```

Reversible (PD) /tikz/reversible

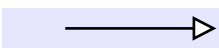

```
\draw[reversible] (0,0) -- (2cm,0);
```

Modulation (PD, AF, ER) /tikz/modulation



```
\draw[modulation] (0,0) -- (2cm,0);
```

Stimulation (PD, AF, ER) /tikz/stimulation

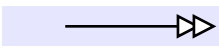


```
\draw[stimulation] (0,0) -- (2cm,0);
```

Necessary stimulation (PD, AF, ER) /tikz/necessary stimulation

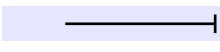
```
\draw[necessary stimulation] (0,0) -- (2cm,0);
```

Absolute stimulation (ER) /tikz/absolute stimulation



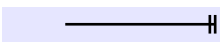
```
\draw[absolute stimulation] (0,0) -- (2cm,0);
```

Inhibition (PD, AF, ER) /tikz/inhibition



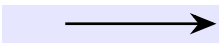
```
\draw[inhibition] (0,0) -- (2cm,0);
```

Absolute inhibition (ER) /tikz/absolute inhibition



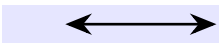
```
\draw[absolute inhibition] (0,0) -- (2cm,0);
```

Assignment (ER) /tikz/assignment



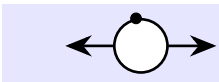
```
\draw[assignment] (0,0) -- (2cm,0);
```

Interaction (ER) /tikz/interaction



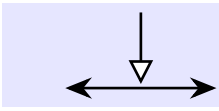
```
\draw[interaction] (0,0) -- (2cm,0);
```

N-ary interactions can be drawn using the nary node:



```
\draw[interaction] (0,0) -- node[nary, pos=0.5]
↪  (a) {} (2cm,0);
\node[outcome] at (a.100) {};
```

Anchor point (ER) /tikz/anchor point



```
\draw[interaction] (0,0) -- coordinate[anchor
↪  point, pos = 0.5] (a) (2,0);
\draw[stimulation] (1,1) -- (a);
```

2.3 Nodes' and arcs' attributes

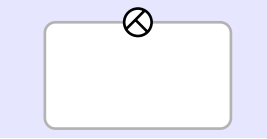
State variable (PD, ER) /tikz/sv


```
\node[entity, draw=gray!60] (m) {};
\node[sv] at (m.north) {val@var};
```

Existence state variable (ER) /tikz/sv existence

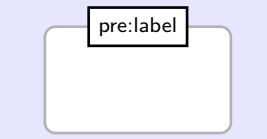
```
\node[entity, draw=gray!60] (m) {};
\node[sv existence] at (m.north) {};
```

Location state variable (ER) /tikz/sv location



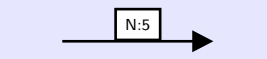
```
\node[entity, draw=gray!60] (m) {};
\node[sv location] at (m.north) {};
```

Unit of information (PD, ER) /tikz/ui



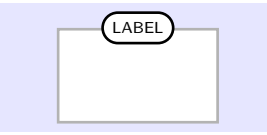

```
\node[entity, draw=gray!60] (m) {};
\node[ui] at (m.north) {pre:label};
```

In PD, stoichiometry can be drawn using a unit of information along an arc:



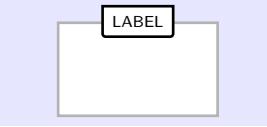

```
\draw[production] (0,0) -- node[ui, above,
↪  pos=0.5] {\tiny N:5} (2cm,0);
```

Unit of information simple chemical (AF) /tikz/ui simple chemical



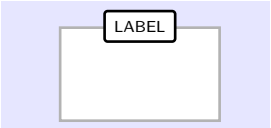

```
\node[biological activity, draw=gray!60] (m) {};
\node[ui simple chemical] at (m.north) {LABEL};
```

Unit of information nucleic acid feature (AF) /tikz/ui nucleic acid feature




```
\node[biological activity, draw=gray!60] (m) {};
\node[ui nucleic acid feature] at (m.north)
↪  {LABEL};
```

Unit of information macromolecule (AF) /tikz/ui macromolecule

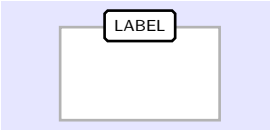



```
\node[biological activity, draw=gray!60] (m) {};
\node[ui macromolecule] at (m.north) {LABEL};
```

Unit of information perturbation (AF) /tikz/ui perturbation


```
\node[biological activity, draw=gray!60] (m) {};
\node[ui perturbation] at (m.north) {LABEL};
```

Unit of information complex (AF) /tikz/ui complex




```
\node[biological activity, draw=gray!60] (m) {};
\node[ui complex] at (m.north) {LABEL};
```

Unspecidied entity subunit (PD) /tikz/unspecified entity subunit


```
\node[unspecified entity subunit] {LABEL};
```

Macromolecule subunit (PD) /tikz/macromolecule subunit


```
\node[macromolecule subunit] {LABEL};
```

Macromolecule multimer subunit (PD) /tikz/macromolecule multimer subunit


```
\node[macromolecule multimer subunit] {LABEL};
```

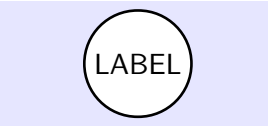
Nucleic acid feature subunit (PD) /tikz/nucleic acid feature subunit


```
\node[nucleic acid feature subunit] {LABEL};
```

Nucleic acid feature multimer subunit (PD) /tikz/nucleic acid feature multimer subunit


```
\node[nucleic acid feature multimer subunit]
↪  {LABEL};
```

Simple chemical subunit (PD) /tikz/simple chemical subunit




```
\node[simple chemical subunit] {LABEL};
```

Simple chemical multimer (PD) /tikz/simple chemical multimer subunit


```
\node[simple chemical multimer subunit] {LABEL};
```

Complex subunit (PD) /tikz/complex subunit

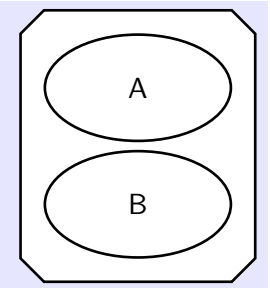

```
\node[complex subunit] {LABEL};
```

Complex multimer subunit (PD) /tikz/complex multimer subunit


```
\node[complex multimer subunit] {LABEL};
```

Subunits (PD) /tikz/subunits

The subunits key allows drawing complexes around subunits:



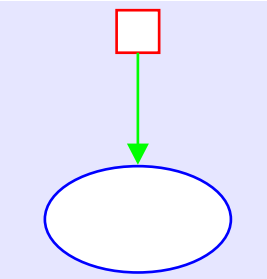

```
\node[unspecified entity subunit] (m1) {A};
\node[unspecified entity subunit, below=0.1cm of
↪  m1] (m2) {B};
\node[complex, subunits=(m1)(m2)] {};
```

3 Customizing and drawing maps effectively

3.1 Useful options for nodes and arcs

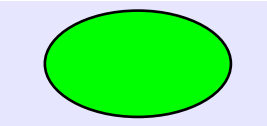
The style of nodes and arcs (and their attributes) can be customized at will using the numerous options offered by Ti*k*Z. Following are a few options that might be useful for customizing SBGN maps.

Foreground color (Nodes, Arcs) `/tikz/draw`



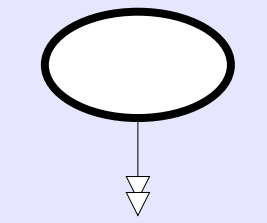
```
\node[generic process, draw = red] (p) {};
\node[unspecified entity, draw = blue, below =
↪ of p] (m) {};
\draw[production, draw = green] (p) -- (m);
```

Background color (Nodes) `/tikz/fill`



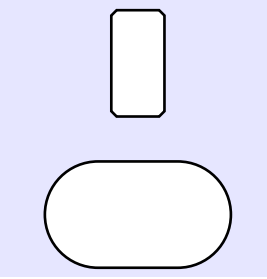
```
\node[unspecified entity, fill = green!120] (m)
↪ {};
```

Line width (Nodes, Arcs) `/tikz/line width`



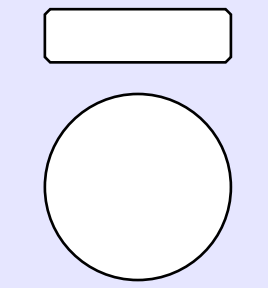
```
\node[unspecified entity, line width = 3pt] (m)
↪ {};
\draw[absolute stimulation, line width = 0.2pt]
↪ (m) -- (0cm,-2cm);
```

Minimum width (Nodes) `/tikz/minimum width`



```
\node[complex, minimum width = 20pt] (m) {};
\node[simple chemical, minimum width = 70pt] at
↪ (0cm,-2cm) (m) {};
```

Minimum height (Nodes) `/tikz/minimum height`



```
\node[complex, minimum height = 20pt] (m) {};
\node[simple chemical, minimum height = 70pt] at
↪  (0cm,-2cm) (m) {};
```

3.2 Positioning of nodes, arcs, and their attributes

Ti*k*Z offers two ways of positioning nodes: with absolute coordinates and relatively to other nodes. We discourage using absolute coordinates as those cannot be changed easily through the drawing process. Drawing maps is far easier using relative positioning. For example, state variables can be placed using the border anchor of the entity pool node they decorate:


```
\node[macromolecule] at (0,0) (m) {};
\node[sv] at (m.140) {val@var};
```

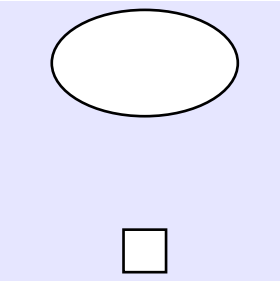
Here, we decided to place the state variable at an angle of 140° on the border of the entity pool node. Now, if we want to change the position of the entity pool node, we don't have to change the position of the state variable:


```
\node[macromolecule] at (0,2) (m) {};
\node[sv] at (m.140) {val@var};
```

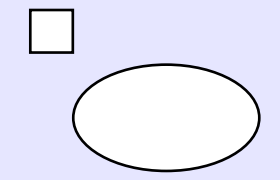
Nodes can also be easily be placed relatively to other nodes using the *positioning* Ti*k*Z library, with the following syntax:

```
\node[<sbgn_node>, <direction> = <distance> of <node_name>] ...;
```

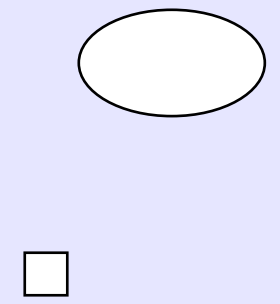
Here, `<direction>` defines in what direction the node should be place relatively to a previously defined node named `<node_name>`. This direction can be unitary (`left`, `right`, `above`, `below`) or composed of two values (vertical direction first, e.g. `above left`). The node will be placed in that direction at the distance `<distance>` of the border of node `<node_name>`. In case of a composed direction, one can define a distance for each sub-direction: `<distance1> and <distance2>`. If no distance is provided, the node will be placed at a distance of 1.5cm (default in *sbgntikz*). Following are some usage examples:



```
\node[unspecified entity] (m) {};
\node[generic process, below = of m] {};
```

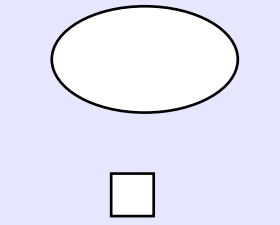


```
\node[unspecified entity] (m) {};
\node[generic process, above left = 0.5cm of m]
↪  {};
```



```
\node[unspecified entity] (m) {};
\node[generic process, below left = 2cm and 0.5cm
↪  of m] {};
```

To make the node closer to the other one, a negative distance can be defined:



```
\node[unspecified entity] (m) {};
\node[generic process, below left = 1cm and -1cm
↪  of m] {};
```

3.3 Bended arcs, multi-part arcs

Ti*k*Z offers a simple way to bend arcs with the following syntax:

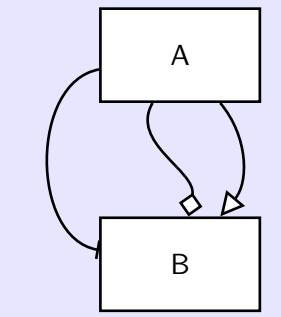
```
\draw (a) to [in=<in_angle>, out=<out_angle>] (b);
```

where `<in_angle>` specifies the angle at which the arc leaves the source point or node and `<out_angle>` the angle at which the arc arrives on the target point or node. Both angles are defined relatively to the picture's coordinate.

One can also use the following shortcut:

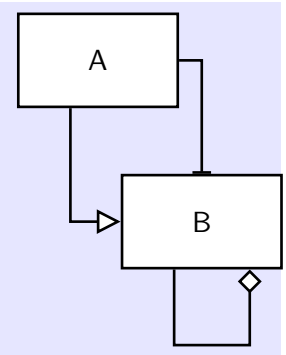
```
\draw (a) to [bend <direction>=<angle>] (b);
```

where <direction>={left, right} specifies the direction where to bend the arc and <angle> the angle at which the arc leaves the source point or node. The angle is this time defined relatively to the line passing through both points/nodes.




```
\node[biological activity] (a) {A};
\node[biological activity, below = of a] (b) {B};
\draw[modulation] (a) to [out=-120, in=80] (b);
\draw[stimulation] (a) to [bend left=40] (b);
\draw[inhibition] (a) to [bend right=80] (b);
```

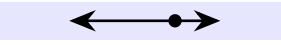
It is often necessary to break arcs into muliple parts for improved readability. Ti*k*Z offers very simple operations to break arcs into horizontal and vertical sub-parts, that replace the default `--` operation. The `|-` operation will produce an horizontal sub-part followed by a vertical one, and the `-|` a vertical sub-part followed by a horizontal one. It can also be convenient to use the `--+` and `--++` to draw arcs with more than two sub-parts.




```
\node[biological activity] (a) {A};
\node[biological activity, below right = 1.5cm and
↪  0.3 cm of a.center] (b) {B};
\draw[stimulation] (a.240) |- (b);
\draw[inhibition] (a) -| (b);
\draw[modulation] (b.240) --++ (0,-1) --++ (1,0)
↪  --++ (0,1);
```

3.4 Nodes along paths for ER maps

In ER maps, outcomes should be drawn along interaction or assignment arcs. Ti*k*Z provides a handy syntax to do so:

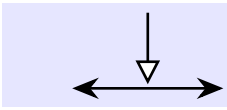


```
\draw[interaction] (0,0) -- (2,0)
↪  node[outcome, pos = 0.7] (o) {};
```

Here, the outcome is added along the path using the `node` command (with no \), as if it was a normal node. The `pos` option allows defining the distance of the node from the source of the arc relatively to the length of the arc. Here, we placed the outcome at a distance of 70%. More than one outcome can be placed along an arc, by repeating the `node` syntax:

```
\draw[interaction] (0,0) -- (2,0)
↪  node[outcome, pos = 0.3] (o1) {}
↪  node[outcome, pos = 0.5] (o2) {}
↪  node[outcome, pos = 0.7] (o3) {};
```

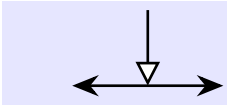
One other particularity of drawing ER maps is that one might have to draw arcs targeting other arcs. This is not straightforwardly possible in Ti*k*Z, as arcs must target points or nodes. One solution provided by *sbgntikz* is to draw `anchor point` coordinates along arcs, as in the following example:



```
\draw[interaction] (0,0) -- (2,0)
↪  coordinate[anchor point, pos = 0.5] (a);
\draw[stimulation] (1,1) -- (a);
```

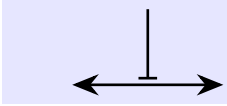
Here, the `anchor point` coordinate (which is a circle with no drawing, no label and a radius of 2pt) is placed halfway along the interaction arc.

Often in ER maps, one wants to draw horizontal or vertical modulation arcs departing from outcomes. These particular cases, on the other hand, can be easily achieved using Ti*k*Z's `|-` syntax:



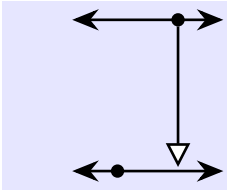
```
\draw[interaction] (0,0) -- (2,0);
\draw[stimulation] (1,1) -- (1,1 |- 2,0);
```

Additional space between the tip of the modulation arc and the target arc can be added (here 2pt, that is the anchor point's default radius):



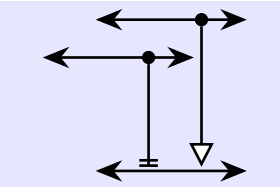
```
\draw[interaction] (0,0) -- (2,0);
\draw[inhibition] (1,1) -- ($(1,1 |-
↪  2,0)+(0,2pt)$);
```

Drawing vertical or horizontal modulation arcs can also be achieved using anchor points and Ti*k*Z's `let` syntax, which allows to place the anchor point at the X-coordinate (for a vertical arc) or Y-coordinate (for an horizontal arc) of the outcome. Following is an example:



```
\draw[interaction] (0,0) -- (2,0) node[outcome,
↪  pos = 0.7] (o1) {};
\draw[interaction] let \p1=(o1) in (0,-2) --
↪  (2,-2) coordinate[anchor point] (a) at
↪  (\x1,-2) node[outcome, pos = 0.3] (o2) {};
\draw[stimulation] (o1) -- (a);
```

In the second interaction, we define the point `\p1` as being the outcome of the second interaction, and we use its X-coordinate defined as `\x1` to place the anchor point. More coordinates can be defined and accessed using the let command, always using the syntax `\p`i and `\x`i,`\y`i:



```
\draw[interaction] (0,0) -- (2,0) node[outcome,
↪  pos = 0.7] (o1) {};
\draw[interaction] (-0.7,-0.5) -- (1.3,-0.5)
↪  node[outcome, pos = 0.7] (o2) {};
\draw[interaction] let \p1=(o1), \p2=(o2) in
↪  (0,-2) -- (2,-2) coordinate[anchor point] (a1)
↪  at (\x1,-2) coordinate[anchor point] (a2) at
↪  (\x2,-2);
\draw[stimulation] (o1) -- (a1);
\draw[absolute inhibition] (o2) -- (a2);
```

References

[1] Till Tantau. *The TikZ and PGF Packages.*

[2] Nicolas Le Novere, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, et al. The systems biology graphical notation. *Nature biotechnology*, 27(8):735, 2009.

Index