# XY-pic User's Guide

Kristoffer H. Rose ⟨krisrose@brics.dk⟩[×]

Version 3.3, December 19, 1996


## Abstract

XY-pic is a package for typesetting graphs and diagrams using Knuth's TeX typesetting system. XY-pic works with most of the many formats available, *e.g.*, plain TeX, LaTeX, and AMS-TeX. Several styles of input for various diagram types are supported; they all share a mnemonic notation based on the *logical composition of visual components*. This guide concentrates on how to typeset "matrix-like" diagrams, such as commutative diagrams, in the following style:

$$\begin{tikzcd}
U \arrow[ddr, bend right, "y"'] \arrow[drr, bend left, "x"] \arrow[dr, dotted, "{(x,y)}" description] & & \\
& X \times_Z Y \arrow[d, "q"] \arrow[r, "p"'] & X \arrow[d, "f"'] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}$$

was typeset by the XY-pic input lines

```
\xymatrix{
 U \ar@/_/[ddr]_y \ar@/^/[drr]^x
   \ar@{.>}[dr]|-{(x,y)}            \\
  & X \times_Z Y \ar[d]^q \ar[r]_p
                 & X \ar[d]_f       \\
  & Y \ar[r]^g   & Z                }
```

Such diagrams have the following characteristics:

- Specified as a matrix of entries that are automatically aligned in rows and columns.
- Any entry may be connected to any other entry using a variety of arrow styles all rotated and stretched as required.
- Arrows may be decorated with labels that are tied to a specified point along the arrow and extend in a particular direction; and arrows may be paired, cross, and visit/bend around other entries "on the way."

Several other styles of input are supported; a short survey of the possibilities is included last at the end along with information on how XY-pic can be obtained.


[×]BRICS, Computer Science, Aarhus University (bld. 540), Ny Munkegade, DK–8000 Aarhus C, Denmark.

## Contents

# Preface

This guide explains some features of XY-pic that are relevant to typesetting of "matrix-like diagrams" as used in, for example, category theory; please refer to the reference manual [8] for complete information on the described constructions. The guide assumes that you have some experience in using TEX for typesetting mathematics, *e.g.*, have studied [2, ch. 16–19], [3, sec. 3.3], or [9], and that XY-pic is installed on your TEX system as described in the `INSTALL` file accompanying the distribution.

The first section describes what you need to get started, in particular all that is needed to typeset the diagram in the abstract. Section 2 and 3 explain advanced use of arrows and entries, respectively. Finally, section 4 explains where and under what conditions XY-pic is available, gives the relation of version 3.3 to previous versions, and lists further sources of information.

Throughout we give exercises that you should be able to solve as you go along; all exercises are answered at the end just prior to the references and index.

# 1 Basics

This section explains the XY-diagram construction concepts needed to get started with typesetting matrix-like diagrams.

## 1.1 Loading

The XY-pic setup used in this guide is loaded by inserting the lines

```
\input xy
\xyoption{all}
```

in the definitions part of your document.[1] If you wish to load only the features you use, or you wish to use non-standard facilities like the `v2` backwards compatibility mode[2] or the `ps` POSTSCRIPT[3] backend then this is also possible as described in the reference manual [8].

## 1.2 Entries

A diagram is created by the command

```
\xymatrix{ ... }
```

where the "..." should be replaced by *entries* to be *aligned in rows and columns* where

- entries in a row are separated by `&` and
- entire rows are separated by `\\`.

For example,

$$\begin{tikzcd} A & \boxed{\sum_{i=n}^{m} i^2} & \\ & \bullet & D \arrow[ul] \end{tikzcd}$$

was typeset by

```
\xymatrix{
  A &*+[F]{\sum_{i=n}^m {i^2}} \\
    & {\bullet} & D \ar[ul]     }
```

Notice the following:

- entries are typeset as mathematics (using "text style"); entries should not start with a macro (as illustrated by the use of `{}` around `\bullet`.
- all entries are centered,
- the separation between rows and columns is usually quite large in a diagram,
- empty entries at the end of rows may be omitted,
- "XY-decorations" (here `\ar[ul]`) in entries allow drawing of arrows and such relative to the entries without changing the overall layout, and
- "XY-modifiers" (here `*+[F]`) first in entries allow changing the format and shape in many ways.

## 1.3 Arrows

An "arrow" in an XY-pic diagram is a generic term for the drawn decorations between the entries of the basic matrix structure. In XY-pic all arrows must be specified along with the entry they start in; this is called their *base entry*. Each particular arrow command then refers explicitly to its *target entry*. This is obtained using the `\ar` command which accepts many options of which we will describe a few here and some more in section 2. In its simplest form an arrow is entered as `\ar[`*hop*`]` where *hop* is a sequence of single letters: `u` for up, `d` for down, `l` for left, and `r` for right, *e.g.*, the arrow `\ar[ur]` reads "typeset an arrow from the current entry to that one up and one right."

**Exercise 1:** Which entry does `[]` refer to?

The relative coordinates specified in this way are purely logical, *e.g.*, if the diagram contains very wide entries then "diagonal" arrows will be nearly horizontal. The constructed arrows are aligned along the line between the centers of the base and target entries; they will not

[1] LATEX $2_\varepsilon$ [3] users can use `\usepackage[all]{xy}`.

[2] If you use the version 2 loading command `\inputxypic` then the `v2` option will be loaded automatically.

[3] POSTSCRIPT is a registered Trademark of Adobe, Inc. [1].

automatically disappear under entries that they cross (we discuss how this is achieved in section 2.7).

The arrow style kan be changed by writing the command as `\ar@`*style*`[`*hop*`]`. This will be described in more detail in section 2.3; here we just list the most common `@`*style*s (obvious variations also work):

`@{=>}` `@{.>}` `@{:>}` `@{~>}` `@{-->}` `@{-}` `@{}`

**Exercise 2:** Typeset

$$\begin{tikzcd}
\bullet \arrow[r, no head] \arrow[d, dashed, no head] \arrow[dr, Rightarrow, no head] & \bullet \arrow[d, dotted, no head] \\
\bullet & \bullet \arrow[l]
\end{tikzcd}$$

## 1.4 Labels

You can put labels on arrows. Labels are conceptualized as sub- and superscripts on arrows such that they are placed in the usual positions (as "limits"), *i.e.*, `^` reads "above" and `_` "below" on an arrow pointing right. Notice that the positions depend *only* on the direction of the arrow, the absolute notions of "up," "down," etc. are not important. For example,

```
$\xymatrix@1{
 X\ar[r]^a_b & Y & Z\ar[l]^A_B }$
```

will set $X \xrightarrow[b]{a} Y \xleftarrow[A]{B} Z$ (the `@1` is a special code that can be used for "one-line" diagrams to improve the placement on the line; more such spacing codes are described in section 3.3).

It is possible to use labels that are not single letters, digits, or control sequences: if a simple math formula in the default style (script style) is desired then simply enclose in `{...}`. In practice anything can be used as a label as described in section 2.2.

Each label is placed perpendicular to the arrow at the point halfway between the centers of the base and target objects. This is usually the most æsthethic, however, in diagrams where the sizes of the entries vary much it is sometimes nicer to place the label at the center of the actual arrow. This behaviour is requested by inserting a `-` (minus) right after the `^` or `_`: $A \times B \times C \times D \xrightarrow{+} B$ was typeset by

```
$\xymatrix@1{
 A\times B\times C\times D \ar[r]^-{+} &B
}$
```

(it becomes $A \times B \times C \times \overset{+}{D} \longrightarrow B$ without the `-`). In fact `-` is in just one of the may possible placings of labels described in section 2.1.

**Exercise 3:** Typeset the second axiom of category theory as

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "f;g"'] & B \arrow[d, "g"] \arrow[dr, "g;h"] & \\
& C \arrow[r, "h"'] & D
\end{tikzcd}$$

## 1.5 Breaks

It is also possible to "break" an arrow with a label using the `|` character: `$\xymatrix@1{A\ar[r]|f&B}$` will set $A \relbar f \!\rightarrow B$ .

If you just want an empty break you should use the special `\hole` break: the arrow $A \relbar\ \ \rightarrow B$ was typeset by including `$\xymatrix@1{ A\ar[r]|\hole & B }$` in the text.

A different use of breaks is to place a label somewhere in a diagram outside the normal matrix mesh: this is accomplished by "breaking" an invisible arrow obtained using the `@{}` arrow style: the square

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "="] & B \arrow[d] \\
B \arrow[r] & C
\end{tikzcd}$$

was typeset by

```
\xymatrix{\ar @{} [dr] |{=}
 A \ar[d] \ar[r] & B \ar[d] \\
 B \ar[r]        & C        }
```

There is more on breaks in section 2.7.

**Exercise 4:** Typeset the first axiom of category theory as the display

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "f"'] & B \arrow[d, "g"] \arrow[dl, "i_B" description] \\
B \arrow[r, "g"'] & C
\end{tikzcd}$$

## 1.6 Curving

Arrows can be made to curve, for example to avoid going through another entry, using the special style `@/`*curving*`/`. The simplest styles of *curving* are the following, shown applied to an arrow from $A$ to $B$:

`@/^/` $\begin{tikzcd} A \arrow[r, bend left] & B \end{tikzcd}$

`@/_/` $\begin{tikzcd} A \arrow[r, bend right] & B \end{tikzcd}$

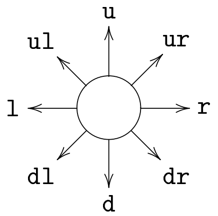
`@/_1pc/` $\begin{tikzcd} A \arrow[r, bend right=60] & B \end{tikzcd}$

As the last example shows a dimension can be inserted just after `^` or `_` if more or less curving is desired.

In case it is easier to specify the in- and out-going directions of the curving then that is also possible: use

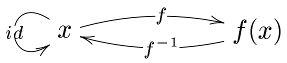
`@(`*in*`,`*out*`)`

where *in* and *out* are one of the following *directions*:




In this case the curving is computed such that the curve begins at the base entry in the *in* direction and ends at the target entry from the *out* direction (this means that `@(`$d_1$`,`$d_2$`)` and `@(`$d_2$`,`$d_1$`)` are mirror images. See section 2.6 for more directions).

**Exercise 5:** Typeset




### 1.7 Speeding up typesetting

One thing that you will notice is that XY-pic is sometimes slow in typesetting diagrams (this is to be expected considering the number of drawing operations performed as reflected by the number last in each `xymatrix` message). If you followed the rule of starting all entries with a (nonexpandable) character or `{` then you can insert the declaration

`\CompileMatrices`

in the preamble of your document: this will create temporary files[4] containing *compiled* versions of each matrix that can be loaded very quickly; they are automatically recreated when a matrix is changed.

If this causes some diagrams to not work, then such compilation can be explicitly switched off by using `\xymatrixnocompile` in place of `\xymatrix`. Compilation can be switched off completely with `\NoCompileMatrices` (which respects TEX grouping as does `\CompileMatrices`, by the way).

And if you are still not satisfied with the speed then you can add the following:

`\OnlyOutlines`

which will omit all compiled pictures; the additional command `\ShowOutlines` will add a dotted rectangle outlining the size of the picture.

[4] The temporary files are named the same as your document but `.tex` is replaced by `-`*n*`.xyc` where *n* is a sequence number.

## 2 More Arrows and Labels

In this section we explain a number of variations of the arrow commands that are useful in commutative diagrams.

### 2.1 Explicit label positioning

The label commands explained in section 1.4 place the label text near the point along the arrow halfway between the centers of the base and target entries. This, however, may be changed by inserting a *place* between the `^`, `_`, or `|`, and the actual label (in fact `-` is a *place*). In general you may insert the following:

- `<` will place the label at the point where the actual arrow begins, *i.e.*, "appears from under" the base, so `$\xymatrix@1{A\ar[r]^<{+}&B}$` will typeset $A \xrightarrow{+} B$ .

- Similarly, `>` will place the label at the point where the actual arrow ends, *i.e.*, "disappears below" the target, so `$\xymatrix@1{A\ar[r]^>{+}&B}$` will typeset $A \xrightarrow{+} B$ .

- `<<` and `>>` will place the following label at a point just a bit[5] further from the beginning and end of the arrow, so `$\xymatrix@1{A\ar[r]^>>{+}&B}$` will typeset $A \xrightarrow{+} B$ . Using more `<`s or `>`s will move the label further in.

- A factor in `()`s: `(`$a$`)` indicates that the label should be "tied" to the point $a$ of the way from the center of the base entry (called `(0)`) to the center of the target (called `(1)`) instead of in the middle, so `$\xymatrix@1{A\ar[r]^(.3){+}&B}$` will typeset $A \xrightarrow{+} B$ .

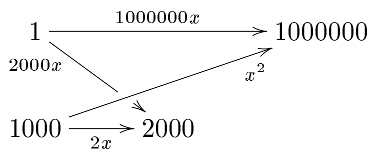
- A factor can be given *after* some `<` or `>`s, in which case the place is computed as if the base was specified by the `<`s and target specified by the `>`s. Hence `$\xymatrix@1{A\ar[r]^<(0){+}&B}$` will typeset $A \xrightarrow{+} B$ .

- Finally, there is a simple way to denote the place on an arrow where it intersects with a straight line between two points: the place `!{`$t_1$`;`$t_2$`}` places the label relative to the point on the arrow where the line from the target $t_1$ to the target $t_2$ crosses it. **Bug:** Only works for straight arrows at present.

As usual more possibilities can be found in the reference manual [8, fig. 2].

[5] "A bit" is in fact a TEX `\jot` which is usually 3pt.

**Exercise 6:** Typeset




## 2.2 Labeling with any object

XY-pic supports a general format for entering any TEX text as labels (as well as entries to be explained later). The character `*` is reserved for this: in its simplest form `*{`*math*`}` will typeset the *math* material as an object. This is like `{`*math*`}` except that the default style is ignored and there is no added blank margin.

However, in general the following form of `*`*object* is available:

| |
|---|
| `*`*modifiers*`{`*text*`}` |

where *modifiers* can be used to change the shape and size of the constructed object. The following are the most common, the full list of possibilities can be found in the reference manual [8, fig. 3]:

| | |
|---|---|
| `+` | grow |
| `+<`*dimen*`>` | grow by *dimen* |
| `+=` | grow to enclosing square |
| `-` | shrink |
| `-<`*dimen*`>` | shrink by *dimen* |
| `-=` | shrink to contained square |
| `!` | do not center |
| `[o]` | round |
| `[l] [r] [u] [d]` | adjust left, right,... |
| `[F] [F=]` | frame double |
| `[F.] [F--]` | dotted dashed |
| `[F-,] [F-:<3pt>]` | shaded rounded |

Since objects specified this way start with no margin, a single `+` is usually included to get the default spacing.

**Exercise 7:** Typeset $A \xrightarrow{x} B$ .

There can only be one *boxing*. This can be any box generation command. The following are the most useful *boxing*`{`*text*`}` combinations:

| | |
|---|---|
| `@`*variant*`{`*tip*`}` | tip (or shaft) object |
| `\txt{...}` | ordinary *text* |
| `\composite{...*...}` | combined objects |
| `\frm{}` | repeat last object |

(the possibilities for *variant* and *tip* are given in the following section). Finally, `\hbox{...}`[6] is a quick way to ensure text-mode interpretation of a single object. However, `\txt` allows the use of `\\` in *text* to create a line break, and the special form `\txt<6pc>{...}` will constrain the text to a centered 6pc wide column. By the way, `\txt` can be used outside of XY-pic constructions.

Finally, several objects can be combined using the last form; the `*`s serve to separate the composed object.

**Exercise 8:** Typeset $A \overset{\text{High label}}{/\ast\ast\ast\ast\ast\ast\ast/} B$ .

## 2.3 More arrow styles

The arrow styles described in section 1.3 are all examples of the general *arrow style* constructions

| |
|---|
| `@`*variant*`{`*tail shaft head*`}` |
| `@`*variant*`{`*head*`}` |

that in describes arrows with the indicated *tail*, *shaft*, and *head* (on the first form the tail and head can be omitted; the second style defaults to having no tail and a standard shaft).

The following possibilities exist for *head* and *tail* which we will denote *tips* (here shown as heads):

| | | |
|---|---|---|
| `<` | `>` | `x` |
| `<<` | `>>` | `+` |
| `\|<` | `>\|` | `\|` |
| `\|<<` | `>>\|` | `\|\|` |
| `(` | `)` | `o` |
| `/` | `//` | |

and the *shaft* should be one of the following:

| | |
|---|---|
| `-` | `--` |
| `~` | `~~` |
| `.` | |

The *variant* should be empty or one of the following:

| | |
|---|---|
| `^` | "above" variant |
| `_` | "below" variant |
| `2` | "double" variant |
| `3` | "triple" variant |

Here are some standard arrows in this notation, all from $A$ to $B$ as usual:

| | |
|---|---|
| `@{<->}` | $A \longleftrightarrow B$ |
| `@^{<->}` | $A \leftharpoonup\!\!\rightharpoonup B$ |
| `@_{<->}` | $A \leftharpoondown\!\!\rightharpoondown B$ |
| `@2{<->}` | $A \Longleftrightarrow B$ |
| `@3{<->}` | $A \Lleftarrow\!\Rrightarrow B$ |

[6] The plain TEX command `\hbox` corresponds to `\mbox` in LATEX and `\text` in the $\mathcal{AMS}$ variants.

As a special convenience `=` and `:` are provided as abbreviations for `-` and `.` with variant forced to 2.

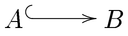
As it can be seen, the variant will affect the *entire* arrow. Sometimes this is not what is wanted. In that case a *local variant* can be used by entering any of the *tail*, *shaft*, and *head*, on the following form:

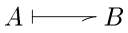
---

*variant*{*tip*}
*variant*{*shaft*}

---

Here are some arrows where this is required:

`@{^{(}->}` 

`@{|-_{>}}` 

Notice that there is no distinction between shafts and tips using this form, thus it is necessary to include all three of *tail*, *shaft*, and *head*, when using it. The advantage is that it is possible then to "fill with a tip." Furthermore, the following additional possibilities are available when using this notation:

| | | | | | |
|---|---|---|---|---|---|
| `{\|-}` | | `^{\|-}` | | `_{\|-}` | |
| | | `{\|=}` | | `3{\|-}` | |
| `{*}` | | `^{‘}` | | `_{‘}` | |
| | | `^{’}` | | `_{’}` | |

The even more general form `*`*object* can be used, where *object* refers to any of the constructions described in section 2.2.

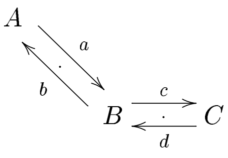
**Exercise 9:** Typeset  .

## 2.4 Sliding arrows sideways

It is often desirable to have several parallel arrows between two objects. This can be done by sliding either or both arrows sideways a distance given as a TEX dimension enclosed in `@<>`s: it specifies how far "sideways" the arrow should be moved, *e.g.*,

```
\xymatrix{
 A \ar@<1ex>[dr]^a_{.} \\
   & B \ar@<1ex>[ul]^b \ar@<1ex>[r]^c
     & C \ar@<1ex>[l]^d_{.} }
```

will typeset




A positive distance will slide the arrow in the "`^`-direction," *e.g.*, the two arrows above are slid in the direction of the labels $a$, $b$, $c$, and $d$, respectively; a negative distance in the "`_`-direction." The distance `<@1ex>` is often appropriate since it corresponds roughly to the height of letters like "x," in the type size being used.

**Exercise 10:** Typeset  .

## 2.5 More targets

The target address can be given in a large number of formats called *positions*. The full range of possibilities is described in the reference manual [8, fig. 1]; here is a number of useful ones in addition to the *hop* format described in section 1.3:

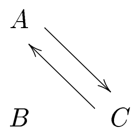
- `[`*r*`,`*c*`]`, where $r, c$ are integers, denotes the *relative entry* found $r$ rows below and $c$ columns to the right of the current entry (the current entry itself is thus `[0,0]`). This always corresponds to a `[`*hop*`]`, *e.g.*, `[1,2]` is the same as `[drr]` and `[-2,0]` is the same as `[uu]`.

- `"`*r*`,`*c*`"`, where $r, c$ are positive integers, denotes the *absolute entry* found in the $r$th row and $c$th column of the diagram. The top left entry is `"1,1"`.

- $t'$`;`$t$ , where $t'$ is any target, changes the base entry of the present arrow to $t'$ and then sets the target to $t$ relative to the original base entry. For example,

  ```
  \xymatrix{ A \\
             B & C \ar@<1ex>[ul]
                   \ar@<1ex>[ul];[] }
  ```

  typesets

  *i.e.*, the second `\ar[ul]` arrow starts at the `[ul]` entry and ends in the current entry.

See section 3.5 for how to use a label as a target.

## 2.6 Changing the target

It is possible to overwrite a target with another by appending something of the form `*`*object* to it. This has the effect of typesetting the *object* at the current position, thus effectively on top of the target, and then use what was typeset as the target.

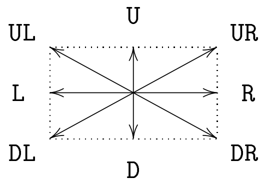
A target may also have its position changed by one of the following constructions:

- `+`*vector* or `-`*vector* which changes the target to be a zero-sized one at the position obtained by adding or subtracting the *vector* to its center, or

- `!`*vector* which moves the center of the target by the *vector*;

where a *vector* should have the form

- `<`$D_x$`,`$D_y$`>` , where $D_x, D_y$ are TEX dimensions, is the vector with those coordinates,
- the following "corner offsets" of a target are vectors as shown:




(they must be specified in upper case), and

- `/`$d$ *dimen*`/` is the *vector* going *dimen* in the particular direction $d$ which can be either the eight simple ones in section 1.6, empty to denote that the *current direction* (the last direction of an arrow) should be used, or one of the following:

| | |
|---|---|
| `va(`$\alpha$`)` | absolute angle |
| $d$`:a(`$\alpha$`)` | relative angle in degrees |
| $d$`:(`$x$,$y$`)` | relative vector |
| $d$`^` / $d$`_` | short for `:a(90)` / `:a(-90)` |

where the $d$ in the last four may be empty to denote the "current direction."

- `0` is the zero vector.

Many, many more possibilities are described in the reference manual [8].

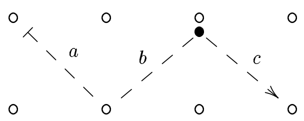
**Exercise 11:** What is the difference between a target $t$ and the target $t$`+0`?

## 2.7 Arrows passing under

Arrows can pass under (or via) any other entry: Just insert `'`$t$, *i.e.*, a quote (apostrophe) character followed by a target, for each entry that should be visited except the last, "ordinary & final" entry:

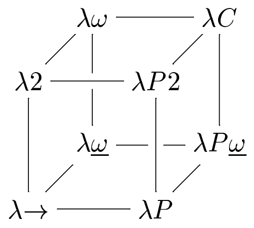
```
\xymatrix{
 {\circ}
  \ar@{|-->} '[dr] ^a
             '[rr]+D*{\bullet} ^b
              [drrr] ^c
         & {\circ} & {\circ} & {\circ} \\
 {\circ} & {\circ} & {\circ} & {\circ} }
```

typesets




As you see, labels are set separately on each segment.

**Exercise 12:** Typeset the "lambda cube"




*Hint*: "going under" an empty entry leaves a small gap at that location. The compactness is achieved using a trick described in section 3.3.

## 2.8 More bending arrows

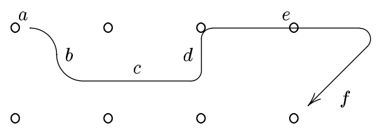
Finally, arrows can bend around entries: just insert `` ` ``$dt$, *i.e.*, a backquote and a direction $d$ followed by a target $t$, for each "turn" that starts out in the $d$-direction and ends in a quarter turn towards the target $t$. This is different from the curving described earlier in that all turns consist of a straight part ending in a turn which is a multiple of $^1/_8$ circle segments, and each segment allows separate labels.

The possible directions are those of section 1.6 and 2.6, and the possible targets include all those discussed above. Actually the direction is only required for the first in a series of turns since the final direction of one turn is the obvious choice for the following turn. Furthermore, turns can be changed from the default by adding either `^`$d$ for anticlockwise turn to $d$ or `_`$d$ for clockwise turn to $d$, where $d$ is the "exit direction" of the turn.

Finally, the turns will have radius 10pt by default, but this can be changed to any dimension $R$ from a particular turn and onwards by inserting `/`$R$ immediately after the `` ` `` of the turn. Here is an example involving all of these features:

```
\xymatrix{
 {\circ} \ar `r[d]    ^a
          `[rr]      ^b
          `/4pt[rr] ^c
          `[rrr]     ^d
          `_dl[drrr]^e
           [drrr]^f
        & {\circ} & {\circ} & {\circ} \\
 {\circ} & {\circ} & {\circ} & {\circ} }
```

typesets




The example illustrates the following points:

- If the segment can not be made as short as required then it will point ‘past” the target. This is useful for “going around” entries.
- There is *one* ‘ *per turn* thus each target appears as many times as there are turns towards it, except the last target that appears one more time namely both with ‘ for each turn towards it *and* once as an “ordinary” target to set the final stretch.
- The sizes of the intermediate targets are ignored.

**Exercise 13:** Typeset $\begin{tikzcd} A \arrow[r, bend right=60, Rightarrow] & B \end{tikzcd}$ .

## 2.9 Defining new arrow types

Last in this treatment of arrows we will explain how new arrows can be defined. The crucial fact is that the characters used for *tips* and *shafts* are restricted to the following:

| | |
|---|---|
| `><|ox+/()[]␣` | *tip* characters |
| `-.~:=` | *shaft* characters |

When an arrow is interpreted by XY-pic it is first split into the three components and then each component is looked up in a library of so-called “directionals.” It is possible to add new such directionals using the command

`\newdir{` *directional* `}{` *composite* `}`

where *directional* should be a sequence either of tip characters or of shaft characters, and *composite* should be a list of objects separated with `*` just like the argument to `\composite` described in section 2.2. If arrows of a particular *variant* (always one of the letters `^_23`) needs an alternate definition then another declaration can be given with the variant inserted between `\newdir` and the first `{`.

There is one object modifier which is very useful in this context, in addition to those of section 2.2:

| | |
|---|---|
| `!`*vector* | shift object *vector* |

(where the possibilities for *vector* are described in section 2.6). Combined with the direction code this is very powerful, for example,

```
\newdir{|>}{%
 !/4.5pt/@{|}*:(1,-.2)@^{>}*:(1,+.2)@_{>}}
```

defines a new tip that makes

```
\xymatrix{ A \ar @{=|>} [r] & B }
```

typeset $A \Longrightarrow B$ . Notice how the “relative direction” is used here to rotate some of the composed components.

**Exercise 14:** Often tips used as “tails” have their ink on the wrong side of the point where they are placed. Fortunately space (␣) is also a tip character so we can define the directional `␣>` to generate a “tail-spaced” arrow. Do this such that

```
\xymatrix{ A \ar @{>->}  @< 2pt> [r]
             \ar @{ >->} @<-2pt> [r] & B }
```

typesets

$$\begin{tikzcd} A \arrow[r, tail, shift left] \arrow[r, tail, shift right] & B \end{tikzcd}$$

Finally, when XY-pic diagrams are used in conjunction with Knuth’s *computer modern fonts* then the declaration

`\SelectTips {cm}{}`

will change the tips to some that look similar, *e.g.*,

```
$\SelectTips{cm}{}
 \xymatrix@1{A\ar@{->>|}[r]&B}$
```

typesets $A \twoheadrightarrow\!| B$ . The second argument (here `{}`) can be used to specify a point size `{10}`, `{11}`, or `{12}`, if desired (the default is `{10}`[7]); `{cm}` can be replaced by `{eu}` and `{xy}` to get tips in “Euler” and the default technical style, respectively. The declaration respects TEX grouping.

**Exercise 15:** Typeset $A \twoheadrightarrow\!\!\twoheadrightarrow B$ . *Hint*: With the construction `\object`*object* one can typeset an XY-pic *object* (anything that could follow a `*` in section 2.2) in any context.

# 3 More Entries

This section explains what can go in an entry and how the general form of the entries is changed.

## 3.1 Manual entry formatting

All the entries we have seen thus far have been simple math objects. However, it is possible to change the format of an individual entry by using the form:

`*`*object arrows*

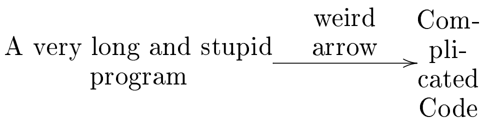
This allows complete control over what object is placed in the entry, overriding any spacing and other conventions for the entry. This was how the frame was obtained in the figure in section 1.2.

A simple use of this is to insert text in entries using `\txt` objects just like labels as described above in section 2.2:

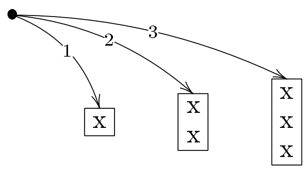
[7] Except when XY-pic is loaded as a LATEX $2_\varepsilon$ package where the default size is used.

```
\xymatrix{
 *\txt{A very long and stupid\\program}
  \ar[rr]^-{\txt{weird\\arrow}}
 &&*\txt<2pc>{Com\-pli\-cated\\Code}}
```

will typeset




**Exercise 16:** Typeset




## 3.2 Extra entries outside the matrix

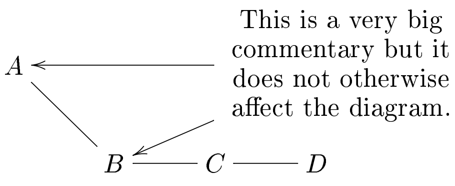
It is possible to put extra entries in your diagrams that are not part of any "entry" of the matrix created by `&` and `\\`. This is done with the *excursion* command

| |
|---|
| `\save` $t$ `\restore` |

where $t$ should be a target in one of the formats described in sections 2.5-2.6. $t$ can do any kind of typesetting desired, for example,

```
\xymatrix{
 A \ar@{-}[dr]
 &{}\save[]+<3cm,0cm>*\txt<8pc>{%
     This is a very big commentary
     but it does not otherwise affect
     the diagram.}
     \ar[l] \ar[d] \restore \\
 & B \ar@{-}[r] & C \ar@{-}[r] & D
}
```

will typeset




It illustrates how a "down" arrow does not necessarily have to point particularly straight down – in this case because it is based in the displaced pseudo entry.

## 3.3 Spacing and rotation

The `*`*object* form described above can be used to space individual objects differently, however, it is also possible to change the overall spacing of a matrix by inserting the following codes *between* `\xymatrix` and the following `{`:

| | |
|---|---|
| `@=`*dimen* | set spacing |
| `@R=`*dimen* | set row spacing |
| `@C=`*dimen* | set column spacing |
| `@M=`*dimen* | set entry default margin |
| `@W=`*dimen* | set entry default width |
| `@H=`*dimen* | set entry default height |
| `@L=`*dimen* | set label margin |

`=` can be replaced by any of `+`, `+=`, `-`, and `-=` of section 2.2 with the same meaning, *i.e.*, replace "set" with "increase," "increase to at most," "decrease," and "decrease to at least," respectively. For example, `$\xymatrix@1@=0pt@M=0pt{A&B\\C&D}$` in the text typesets$\begin{matrix}AB\\CD\end{matrix}$.

In case *uniform spacing* is desired, several alternative forms exist.

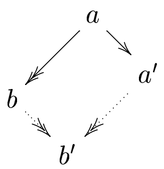
| | |
|---|---|
| `@!` | force all spaces equal |
| `@!0` | — and ignore entry sizes |
| `@!R` | force equal row spacing |
| `@!C` | force equal column spacing |

Finally a special notation allows *rotation* of an entire matrix:

| | |
|---|---|
| `@`$d$ | rotate towards $d$ |

Only the matrix grid will rotate, however, not the actual contents.

**Exercise 17:** Typeset the "strip lemma"

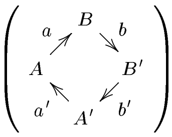



## 3.4 Entry style

As mentioned above, the entries of a diagram are set in math mode in text style. You may change this by redefinining the macro `\objectstyle`, and the label style by redefining `\labelstyle`. We can combine this with the above to get "small diagrams," *e.g.*, typing

```
$\left(
 \def\objectstyle{\scriptstyle}
```

```
 \def\labelstyle{\scriptstyle}
 \vcenter{\xymatrix @-1.2pc @ur {
  A \ar[r]^{a}  & B \ar[d]^{b}  \\
  A'\ar[u]^{a'} & B'\ar[l]^{b'} }}
\right)$
```

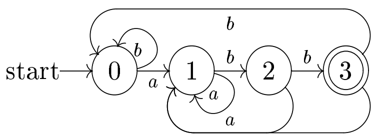
in a paragraph will typeset "."

You can even abandon the use of math mode entirely: the command `\def \objectstyle {\hbox}` will change the format of entries to plain text.

Similarly, all entries are rectangular by default, but it is possible to change this to *round* by declaring the default *modifiers* to be applied to all entries that do not override them (*cf.* section 2.2):

```
\entrymodifiers={++[o][F-]}
\SelectTips{cm}{}
\xymatrix @-1pc {
 *\txt{start} \ar[r]
 & 0 \ar@(r,u)[]^b \ar[r]_a
 & 1 \ar[r]^b \ar@(r,d)[]_a
 & 2 \ar[r]^b
   \ar `dr_l[l]  `_ur[l]  _a [l]
 &*++[o][F=]{3}
   \ar `ur^l[lll]`^dr[lll]^b [lll]
   \ar `dr_l[ll] `_ur[ll]    [ll] }
```

will typeset



Notice how we obtain the double ring using the `*`*object* form which then has to include all the desired modifiers (and how the use of computer modern tips is nice for diagrams as these).

## 3.5 Naming for later use as targets

If you build an entry with a long and complicated excursion then you might wish to be able to refer to it later. XY-pic provides a mechanism for this: there is a special target form which we haven't discussed yet:

| |
|---|
| $t$ `="`*name*`"` |

This will introduce the new target `"`*name*`"` which will refer to the target just before the `=`. This is particularly useful inside excursions, of course, and can also be used after labels.

**Exercise 18:** Typeset $A \xrightarrow{a} B \xrightarrow{b} C$.

## 3.6 Grouping objects

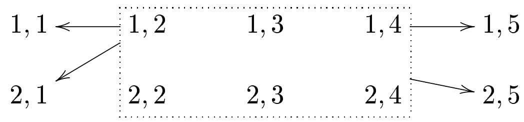
Sometimes you wish to frame or otherwise treat a rectangle of objects as a single object. This is possible with the last two target position forms that we will mention:

| | |
|---|---|
| $t$`.`$s$ | merge $t$ with simple $s$ |
| `{`$t$`}` | make $t$ simple |

The first will enlarge $t$ to also "cover" the "simple" $s$ (simple means that it cannot have changes etc. attached unless encapsulated in `{}`s). Here is an example where we merge and frame:

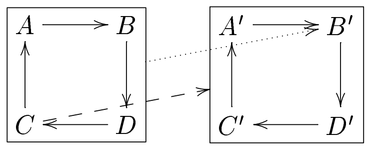
```
\xymatrix @R=1pc {
 1,1 & 1,2 & 1,3 & 1,4 & 1,5 \\
 2,1 & 2,2 & 2,3 & 2,4 & 2,5
 \save "1,2"."2,4"*[F.]\frm{}
  \ar"1,1" \ar"2,1" \ar"1,5" \ar"2,5"
 \restore }
```

will typeset



As you can see, the center of the merged object is the same as the one of the target *before* the "`.`"

Finally a more advanced example where we create two merged objects with center in their "real" center, name them and then connect to them. It also shows how macros can be used inside diagrams: they should always expand to "commands" like `\ar`..., etc.:



can be typeset by

```
\def\g#1{\save
 [].[dr]!C="g#1"*[F]\frm{}\restore}%
\xymatrix{
 \g1 A\ar[r]&B\ar[d]&\g2 A'\ar[r]&B'\ar[d]\\
     C\ar[u]&D\ar[l]&    C'\ar[u]&D'\ar[l]
 \ar @{.>}  "g1" ;"1,4"
 \ar @{-->} "2,1";"g2" }
```

Then we can make arrows from/to the two frames by using the two new targets `"g1"` and `"g2"` as shown.

**Exercise 19:** Change the lambda cube of exercise 12 such it is enclosed in a transparent (*i.e.*, dotted) cube.

# 4 Availability and Further Information

Below we describe how to and the conditions for obtaining XY-pic version 3.3, the compatibility with previous versions, and we conclude with a few appetisers to lure the reader into reading more about XY-pic.

## 4.1 Getting XY-pic

The easiest way to retrieve the latest version of XY-pic is through the XY-pic home page [5] on the World Wide Web; it is also available by Internet ftp from ⟨URL: *ftp://ftp.diku.dk/diku/users/kris/TeX/*⟩ as well as ⟨URL: *ftp://ftp.mpce.mq.edu.au/pub/maths/TeX/*⟩ in files starting with `xy`. Finally, XY-pic has been contributed to the CTAN archives where it is located in the directory `macros/generic/diagrams/xypic`.

**License:** XY-pic is free software in the sense that it is available under the following license conditions:



In practice this means that you are free to use XY-pic for your documents but if you distribute any part of XY-pic (including modified versions) to someone then you are obliged to ensure that the full source text of XY-pic is available to them (the full text of the license in the file `COPYING` explains this in somewhat more detail ☺ ).

## 4.2 Backwards compatibility

The first widely distributed version of XY-pic was version 2 (from release 1.40 to release 2.6). A special *compatibility* mode is used automatically if the old style of loading is used (using files named `xypic.tex` and `xypic.sty`). You can also mix old and new diagrams in a document if you load as described in section 1.1 and add the declaration `\xyoption{v2}`. This provides almost full backwards compatibility: the following are the only known exceptions:

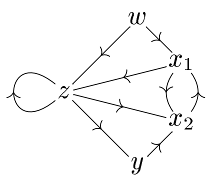
- Automatic "shortening" of arrow tails by `|<<` break was a bug and has been "fixed" so it does not work any more. *Fix*: Put a `|<\hole` break before it as described in section 2.3.
- The release 2.6 `*` position operator is not available. *Fix*: Use the `:` and `::` operators (described in detail in the reference manual [8]).
- Using $t_1$`;`$t_2$`:(`$x$`,`$y$`)` as the target of an arrow command does not work. *Fix*: Enclose it in braces, *i.e.*, write `{`$t_1$`;`$t_2$`:(`$x$`,`$y$`)}`.
- The old `\pit`, `\apit`, and `\bpit` commands are not defined. *Fix*: Use `*@{>}` (or `\tip`) with variants and rotation.
- The even older notation where an argument in braces to `\rto` and the others was automatically taken to be a "tail" is not supported. *Fix*: Use the supported `|<`... notation.

Finally note that sometimes the spacing with version 3.3 is "improved" relative to earlier versions ☺ . Please report all other things that do not work the same in version 2.6 and 3.3 to the author.

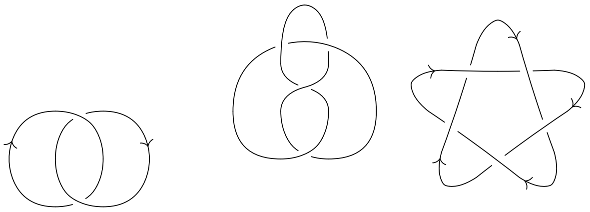
## 4.3 Further reading

The reference manual [8] describes several more input modes that are useful when the diagram is not organised as a matrix. We'll give some examples of such diagrams but refer to the reference manual for the details.

- The "graph" feature allows input of data structured as *directed graphs* to make it easy to produce such pictures as

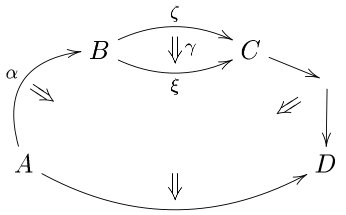



- The "knot" feature allows drawing of mathematical *knots and links* like

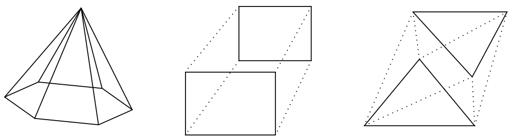


simple link figure 8-knot cinquefoil

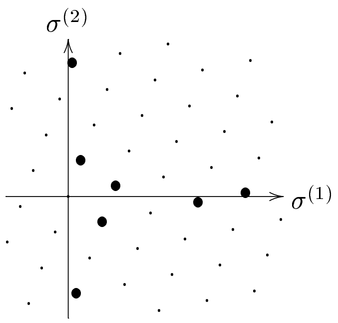
- The "2cell" feature provides special support for *categorical twocells* like



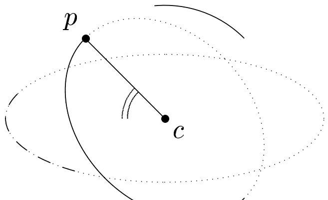

- The "poly" feature allows *polygon-like* structures:



- The "web" feature allows composition in lattices such as




- The "arc" feature is suitable for effects like




In each case the figure is produced with relatively few lines using techniques documented in the respective sections of the reference manual together with the details of the XY-pic kernel and extensions, "arrow," and "matrix" features as used in this guide; further extensions provide support for use of colour, PostScript effects, and much more.

Finally, the following documents might prove useful: *How to Typeset Pretty Diagram Arrows with TEX* [6] presents some of the design decisions behind XY-pic, *XY-pic and Notation for Categorical Diagrams* [7] explains how the modularity of XY-pic can be used to obtain complex effects with commutative diagrams, and *Typesetting Neural Nets using XY-pic* [4] explains how *neural networks* can be typeset using the package. Finally, a collection of complex examples is maintained on the XY-pic home page and the ftp servers discussed above.

### 4.4 Credits

XY-pic version 2 was a small package created by the author. The present version 3.3 is the result of more than three years of continued collaboration with Ross Moore,[8] and would never have been developed without his support and TEX-nical expertise.[9] This includes many of the modes mentioned above which were created for use within specialised branches of mathematics, however, the notations are designed to fit quite naturally within the framework of XY-pic and are useful more generally.

## A Answers to all exercises

**Answer to exercise 1 (p.2):** The target `[]` is the current entry itself.

**Answer to exercise 2 (p.3):** The author did

```
\xymatrix{
 {\bullet} \ar@{--}[d]\ar@{=}[dr]\ar@{-}[r]
           & {\bullet} \ar@{.}[d] \\
 {\bullet} & {\bullet} \ar[l] }
```

Notice how $\bullet$ has been enclosed in `{}` since it is an "expandable" entity, *i.e.*, a defined macro.

**Answer to exercise 3 (p.3):** The author used

```
\xymatrix{
 A \ar[r]^f \ar[dr]_{f;g}
  & B \ar[d]^g \ar[dr]^{g;h} \\
```

[8] Mathematics Department, Macquarie University, Sydney, Australia 2109; e-mail: ross@mpce.mq.edu.au.

[9] Partial funding for this collaboration has been provided by a Macquarie University Research Grant (MURG), by the Australian Research Council (ARC), and through a research agreement with the Digital Equipment Corporation (DEC).

```
 & C \ar[r]_h & D             }
```

**Answer to exercise 4 (p.3):** The author entered

```
\xymatrix{
 A \ar[d]_f \ar[r]^f
            & B \ar[dl]|{i_B} \ar[d]^g \\
 B \ar[r]_g & C }
```

**Answer to exercise 5 (p.4):** The author did

```
\xymatrix{
 x \ar@(ul,dl)[]|{id} \ar@/^/[rr]|f
 && f(x) \ar@/^/[ll]|{f^{-1}} }
```

Note that both arrows are curved "above" relative to their direction.

**Answer to exercise 6 (p.4):** The author used the display

```
\xymatrix{
 1 \ar[rr] ^-{1000000x}
   \ar[dr] _(.2){2000x} |!{[d];[rr]}\hole
                        && 1000000 \\
 1000 \ar[r] _{2x}
      \ar[urr] _>>>>{x^2}
                        & 2000 }
```

In particular notice how the break was specified to happen exactly where the two arrows cross. For an easier but not so general method see exercise 12 last in section 2.7.

**Answer to exercise 7 (p.5):** The author typed

```
\xymatrix@1{ A \ar[r]^*+[o][F-]{x} & B }
```

**Answer to exercise 8 (p.5):** The author did

```
\xymatrix@1{
 A \ar @{/*\composite{{+}*{\times}}/} [rr]
       ^*+\txt{High\\label}
 && B}
```

**Answer to exercise 9 (p.6):** The author entered

```
\xymatrix{
 A \ar @/^/ @{^{<}-_{>}} [r]
   \ar @/_/ @{{*}{x}{*}} [r] & B }
```

**Answer to exercise 10 (p.6):** The author typed

```
\xymatrix@1{
 A \ar@/^/[r] \ar@/^/@<-1ex>[r] & B }
```

**Answer to exercise 11 (p.7):** The size: $t$+0 always has zero size.

**Answer to exercise 12 (p.7):** The author constructed

```
\xymatrix@!0{
 & \lambda\omega \ar@{-}[rr]\ar@{-}'[d][dd]
   & & \lambda C \ar@{-}[dd]
\\
 \lambda2 \ar@{-}[ur]\ar@{-}[rr]\ar@{-}[dd]
 & & \lambda P2 \ar@{-}[ur]\ar@{-}[dd]
\\
 & \lambda\underline\omega \ar@{-}'[r][rr]
   & & \lambda P\underline\omega
\\
 \lambda{\to} \ar@{-}[rr]\ar@{-}[ur]
 & & \lambda P \ar@{-}[ur]
}
```

A special thing is added: `@!` forces rows and columns to be equally spaced as discussed in section 3.3; `@!0` furthermore makes the spacing ignore the entry sizes, giving a completely fixed grid. The gaps could also be made with the `!`... crossing notation of exercise 6 last in section 2.1 but the above is shorter.

**Answer to exercise 13 (p.8):** The author typed

```
\xymatrix@1{
 A \ar@<-2pt> `d[r] `[r] [r]
   \ar@<+2pt> `d[r] `[r] [r] & B }
```

**Answer to exercise 14 (p.8):** The author used

```
\newdir{ >}{{}*!/-5pt/@{>}}
```

**Answer to exercise 15 (p.8):** The author typed

```
\xymatrix@1{
 A \ar[r] |-{\SelectTips{cm}{}\object@{>>}}
          |>{\SelectTips{eu}{}\object@{>}}
 & B }
```

**Answer to exercise 16 (p.9):** The author typed

```
\xymatrix{
 *=0{\bullet}
 \ar@/^/[dr]!U|1
 \ar@/^/[drr]!U|2
 \ar@/^/[drrr]!U|3
\\
 &*+[F]\txt{x}
 &*+[F]\txt{x\\x}
 &*+[F]\txt{x\\x\\x}
}
```

**Answer to exercise 17 (p.9):** The author entered the display

```
\xymatrix@dr@C=1pc{
```

```
 a \ar[r]\ar@{->>}[d] & a' \ar@{.>>}[d] \\
 b \ar@{.>>}[r]       & b' }
```

**Answer to exercise 18 (p.10):** The author typed

```
\xymatrix{
 A \ar[r] ^a="a" & B \ar[r] ^b="b" & C
  \ar @/^/ "a";"b" }
```

Notice the use of both explicit base and target in the arrow between the labels.

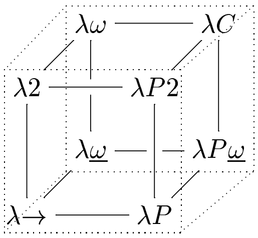
**Answer to exercise 19 (p.10):** The author added the Xy-code

```
\save [].[dd].[ddrr].[rr]*[F.]\frm{}="back"
 \restore
```

in the entry with $\lambda\omega$, and

```
 \save [].[dd].[ddrr].[rr]*[F.]\frm{}
  \ar@{.} +UL;"back"+UL \ar@{.} +UR;"back"+UR
  \ar@{.} +DL;"back"+DL \ar@{.} +DR;"back"+DR
  \restore
```

in the entry with $\lambda 2$ to produce




# References

[1] Adobe Systems Incorporated. *PostScript Language Reference Manual*, second edition, 1990.

[2] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[3] Leslie Lamport. *LATEX—A Document Preparation System*. Addison-Wesley, 2nd edition, 1994.

[4] Ross R. Moore. *Typesetting Neural Nets using Xy-pic*, 1994. Available through [5].

[5] Kristoffer H. Rose. The Xy-pic home page. ⟨URL: *http://www.diku.dk/users/kris/Xy-pic.html*⟩.

[6] Kristoffer H. Rose. How to typeset pretty diagram arrows with TEX—design decisions used in Xy-pic. In Jiří Zlatuška, editor, *EuroTEX '92—Proceedings of the 7th European TEX Conference*, pages 183–190, Prague, Czechoslovakia, September 1992. Czechoslovak TEX Users Group.

[7] Kristoffer H. Rose. Xy-pic and notation for categorical diagrams. Invited talk at ECCT-94, July 1994. Available through [5].

[8] Kristoffer H. Rose and Ross R. Moore. *Xy-pic Reference Manual*. DIKU, University of Copenhagen, Universitetsparken 1, DK–2100 København Ø, 3.0 edition, June 1995.

[9] Michael D. Spivak. *The Joy of TEX—A Gourmet Guide to Typesetting with the AMS-TEX Macro Package*. American Mathematical Society, second edition, 1990.

# Index