

Code tips from “Introduktion til L^AT_EX”^{*}

Or just the `dlfltxbcodetips` package

Lars Madsen[†]

August 5, 2008

Contents

Introduction	1
1 Extra symbols	2
1.1 A big version of <code>\times</code>	2
1.2 Negated up- and downarrows	2
2 Fun with theorems	2
2.1 Shaded or framed theorems with the <code>ntheorem</code> package	2
2.2 Theorems that start with a list	3
3 Various features regarding alignment	4
3.1 Arrow between lines	4
3.2 Switch dead space	4
3.3 Left alignment	5
3.4 Alignment with material encased in delimiters on different lines	6
4 Declaring sets	8
5 Misc	8
5.1 Additional layout for the <code>pfgpages</code> package	8
5.2 Overloading <code>_</code> in math-mode	9
Bibliography	9

Introduction

In my L^AT_EX book (Madsen, 2007) we present some macros that might be helpful to the readers. Some of these extra macros might be useful to others as well so these macros have been included in the `dlfltxbcodetips` package. The package is published on CTAN and the package is released under the normal lppl license.

^{*}Version: 0.2

[†]Web: <http://home.imf.au.dk/daleif> Email: daleif@imf.au.dk

The »dfltxb« part of the name simply stands for »daleif« and L^AT_EX book. The dfltxbcodetips package is the first package in the »dfltxb«-bundle which, over time, will contain most of the home made packages that I use to create my book (though not the book source itself).

Some of the macros might be better of included in the mathtools package by Morten Høgholm, but he is quite busy elsewhere at the moment.

Note: The macro \dbx will often be used to simulate some text or mathematical material.

1 Extra symbols

1.1 A big version of \times

\bigtimes A few extra symbols have been created. First of is \bigtimes which is a large operator version of \times, but without having to load special fonts.¹

$$\bigtimes_{n=1}^k A_n$$

$$\bigtimes_{n=1}^k A_n$$

1.2 Negated up- and downarrows

\nuparrow The package creates \nuparrow and \ndownarrow by rotating and reflecting \nrightarrow and \nleftarrow respectively.¹

$$A \nuparrow B \quad B \ndownarrow C$$

$$A \nuparrow B \quad B \ndownarrow C$$

Remark. The mathdesign package is incompatible with amssymb, but it does define the symbols we need from it to define \nuparrow and \ndownarrow. Use

`\usepackage[noamssymb]{dfltxbcodetips}`

to disable the autoloading of amssymb, and remember to load dfltxbcodetips *after* mathdesign.

2 Fun with theorems

2.1 Shaded or framed theorems with the ntheorem package

\NewShadedTheorem The ntheorem package can create shaded or framed theorems, but they take up to much space (in my opinion). So we make our own macro \NewShadedTheorem. It has exactly the same syntax as the ordinary \newtheorem. Requirements: the framed, ntheorem (loaded with the framed option), and color or xcolor. You will have to \theoremframecommand redefine \theoremframecommand to get a background color or a frame. This package

¹Updated version due to Enrico Gregorio.

initialises `\theoremframecommand` to do nothing.

Caveat. The theorem environment constructed will not have a starred companion as `ntheorems normal \newtheorem` does.

```
\def\theoremframecommand{\fboxsep=10pt\fbox}
\NewShadedTheorem{sthm}{Theorem}[chapter]
\def\theoremframecommand{%
\colorbox{red}}
\NewShadedTheorem{slemma}{sthm}{Lemma}
\newtheorem{prop}{sthm}{Proposition}
\begin{sthm}
normal test.
\end{sthm}
\begin{slemma}
a lemma.
\end{slemma}
\begin{prop}
a theorem with no background.
\end{prop}
```

Theorem. normal test.

Lemma. a lemma.

Proposition. a theorem with no background.

2.2 Theorems that start with a list

A theorem that starts with a list looks odd because the first item comes directly after the heading.²

```
\begin{thm}
\begin{enumerate}
\item \dbx[2cm]
\item \dbx[2cm]
\item \dbx[2cm]
\end{enumerate}
\end{thm}
```

Theorem 1. 1.

2.

3.

`\InsertTheoremBreak` The macro `\InsertTheoremBreak` helps.

```
\begin{thm}
\InsertTheoremBreak
\begin{enumerate}
\item \dbx[2cm]
\item \dbx[2cm]
\item \dbx[2cm]
\end{enumerate}
\end{thm}
\begin{thm}
\InsertTheoremBreak*
\begin{enumerate}
\item \dbx[2cm]
\item \dbx[2cm]
\item \dbx[2cm]
\end{enumerate}
\end{thm}
```

Theorem 2.

1.

2.

3.

Theorem 3.

1.

2.

3.

²Depends on the configuration.

The un-stared version remove the space above the list, the stared version does not.

Caveat. If the theorem comes too close to a page break it is quite likely that the page break will end up between the theorem header and the start of the list.

3 Various features regarding alignment

3.1 Arrow between lines

Sometimes lines in an alignment are related in the sense of *from which it follows*, usually indicated by \Rightarrow . We would like to place this between the lines in a nice way. The following macro is due to Morten Høgholm.

```
\ArrowBetweenLines      \ArrowBetweenLines[\langle arrow \rangle]
\ArrowBetweenLines*     \ArrowBetweenLines*[\langle arrow \rangle]
```

Simply hold back one column of alignment for the arrow:

```
\begin{alignat}{2}
&& f(x) \& = \dbx[2cm] \\\
&\ArrowBetweenLines
&& g(x) \& = - \dbx[2cm] \\\
&\ArrowBetweenLines[\Downarrow]
&& h(x) \& = 0
\end{alignat}
```

$$\begin{array}{rcl}
 f(x) & = & \boxed{} \quad (1) \\
 \Updownarrow & & \\
 g(x) & = & -\boxed{} \quad (2) \\
 \Downarrow & & \\
 h(x) & = & 0 \quad (3)
 \end{array}$$

Note the height of the arrow line and that the line automatically does not contain equation numbers. The starred version can be used to place the arrows on the right. Though several alignment columns on each line might become a problem.

```
\begin{alignat}{2}
&& f(x) \& = \dbx[2cm] \\\
&\ArrowBetweenLines*
&& g(x) \& = - \dbx[2cm] \\\
&\ArrowBetweenLines*[\Downarrow]
&& h(x) \& = 0
\end{alignat}
```

$$\begin{array}{rcl}
 f(x) & = & \boxed{} \quad (4) \\
 & & \Updownarrow \\
 g(x) & = & -\boxed{} \quad (5) \\
 & & \Downarrow \\
 h(x) & = & 0 \quad (6)
 \end{array}$$

This macro have been added to the `mathtools` package, and will be available with the next `mathtools` release, afterwhich it will be removed from this package.

3.2 Switch dead space

The standard *equation* environment has the feature that if the text before it is short and the formula likewise, then the drop between the preceding text and the formula will be

`\abovedisplayshortskip` instead of the usual `\abovedisplayskip`. The macros from the `amsmath` package does not have this feature (except ***equation****).

```
text
\[ a=b \]
text
\begin{align*}
a=b
\end{align*}
```

`\SwapDeadSpace` The macro `\SwapDeadSpace` can simulate this feature

```
text
\[ a=b \]
text
\begin{align*}
\quad \text{\SwapDeadSpace}
\quad a=b
\end{align*}
```

3.3 Left alignment

In [Swanson \(1999\)](#), Ellen Swanson presents some recommendations regarding how to arrange displayed formulas. One thing she recommends is alignment to the left and with all subsequent lines indented by 2 em.

$$\begin{aligned} & \boxed{} = \boxed{} \\ & = \boxed{} \\ & = \boxed{} . \end{aligned}$$

or

$$\begin{aligned} &= \frac{1}{2} \left(\frac{1}{2} \right) \\ &= \frac{1}{4} \end{aligned}$$

or

$$\begin{aligned} & \boxed{} = \boxed{} \\ + & \boxed{} \\ \times & \boxed{} . \end{aligned}$$

and

- _____
- _____
- ⊗ _____.

But having to do all those indentations by hand quickly becomes quite tedious. Instead, align everything to the left, and pull back the first line. This is what the following macro does.

Please note that this macro have been added to the mathtools package and will be available with the next release of mathtools, afterwhich it will be removed from this package.

`\MoveEqLeft` `\MoveEqLeft[⟨number⟩]`

The optional argument is a number (without unit, em will be used) indicating the amount of indentation. 2 em is the default. Now simply use `\MoveEqLeft` instead of the `»&«` on the first line.

```
\begin{align*}
\MoveEqLeft f(x) + g(x) + \dbx[4cm] \\\
& = \dbx[5cm] \\\
& = \dbx[5cm]
\end{align*}
```

$$\begin{aligned} f(x) + g(x) + & \boxed{} \\ = & \boxed{} \\ = & \boxed{} \end{aligned}$$

The optional argument is usually used whenever the alignment is moved to the right of the equal signs.

```
\begin{align*}
\MoveEqLeft[3] f(x) + g(x) + \dbx[4cm] \\\
= {} & \dbx[5cm] \\\
& + \dbx[5cm]
\end{align*}
```

$$\begin{aligned} f(x) + g(x) + & \boxed{} \\ = & \boxed{} \\ & + \boxed{} \end{aligned}$$

3.4 Alignment with material encased in delimiters on different lines

Swanson also recommends that if one has material encased with delimiters and the delimiters are on different lines, then if space permits, the material should be indented such that the relationship is evident. Usually we would use a `\phantom` to do this, such as in the next example.

```
\begin{align*}
\dbx[5mm]&= \dbx[5mm]\bigl[{} \dbx[3cm] \\\
& \phantom{=}\dbx[5mm]\bigl[{} \\\
& \times \dbx[3cm]{} \\\
& \phantom{=}\dbx[5mm]\bigl[{} \\\
& - \dbx[3cm]{} \bigr] \\\
& = \dbx[3cm]
\end{align*}
```

$$\begin{aligned} \boxed{} &= \boxed{} \bigl[\boxed{} \\ & \times \boxed{} \\ & - \boxed{} \bigr] \\ &= \boxed{} \end{aligned}$$

The problem with this is that it gets tedious and prone to human error. How about instead maintaining a stack of material determining the indentation together with tools

to reset, add to and pop the stack. For this you can use the following macros

<code>\MathIndent</code>	<code>\MathIndent</code>
<code>\SetMathIndent</code>	<code>\SetMathIndent{<i><math code></i>}</code>
<code>\AddtoMathIndent</code>	<code>\AddtoMathIndent{<i><math code></i>}</code>
<code>\PopMathIndent</code>	<code>\PopMathIndent</code>
<code>\PopMathIndent*</code>	<code>\PopMathIndent*</code>

`\MathIndent` is used to set a space corresponding to the current indentation saved on the stack. `\SetMthIndent` takes its argument and saves it on the stack, calculates the current math indent length and ends by typesetting the given argument, i.e. no need to copy anything. Similarly the `\AddtoMathIndent` adds its argument to the stack and adds the length of it to the saved math indent. So instead of copying code, now we simply encase it with either `\SetMathIndent` (for the initialisation) or `\AddtoMathIndent`. `\PopMathIndent` is similar to `\MathIndent`, in that it sets a blank space corresponding to the contents of the stack after we have popped off the top item. `\PopMathIndent*` pops the stack but does *not* set any space.

Now, an illustrative example might be in order:

[illegible]

$$\begin{aligned} \square &= \square \left[\square \right. \\ &\quad + \square \\ &\quad + \square \left\{ \square + \left(\square \right. \right. \\ &\quad \quad \quad \left. \left. + \square \right) \right\} \\ &\quad + \square \left. \right] \\ &\quad + \square \end{aligned}$$

Notice the dual use of `\AddtoMathIndent` such that we can return to the indentation set by the `»{«`.

Of course, non-balanced `\left–\right` constructions may not be used.

4 Declaring sets

It is a good idea to avoid the one (or two) letter shortcuts for sets etc., e.g. `\R` for `\mathbb{R}`, it can cause problems when co-writing articles with people with other naming habits.

To help with this we provide

```
\DeclareMathSet \DeclareMathSet[⟨options⟩]{⟨identifier⟩}
```

By default it can be used as

```
\DeclareMathSet{R}% => \setR = \mathbb{R}
$ \setR $
```

\mathbb{R}

But we have several options to change things. Options (*⟨key⟩=⟨value⟩* style, note that *⟨identifier⟩* is what is given to the forming macro).

format the macro used to format the output, default: `\mathbb`

name this defaults to *⟨identifier⟩*, but can be used to change a part of the macro name, e.g. `\DeclareMathSet[name=Cat,format=\matcal]{C}` results in `\setCat=\mathcal{C}`.

prefix this defaults to `set`, but one might want to use

```
\DeclareMathSet[prefix=group,format=\mathrm]{U}
```

to get `\groupU`, the unitary group.

overwrite boolean, if true, then you can overwrite an existing macro, otherwise it will throw an error.

Note that for sets it is not a good idea to declare them as math operators, as some mathematical operations take sets as their argument, so the spacing before an operator would be wrong in that case.

Note that this will also be added to the `mathtools` package, and then removed from this package

5 Misc

5.1 Additional layout for the `pgfpages` package

```
\ProvidePGFPagesFour- The macro \ProvidePGFPagesFourOnOneWithSpaceForNotes will activate a
OnOneWithSpaceForNotes »4_on_1_with_space_for_notes« layout to be used with the pgfpages package. It
is basically the same as the »8_on_1« layout but leaving the second column empty for
reader to fill in their own notes.
```


5.2 Overloading `_` in math-mode

`\Overload-`
`UnderscoreInMath`

Placing `\OverloadUnderscoreInMath` in the preamble will overload the `_` character in math-mode such that

```
\[
  X_{ab} = X_{\max}
\]
```

$$X_{ab} = X_{\max}$$

that is `_ | . . . |` is that same as `_{\textup{. . .}}`.

Bibliography

Lars Madsen. Introduktion til \LaTeX . <http://www.imf.au.dk/system/latex/bog/>, 2007. The current version of the book is 3rd edition beta.

Ellen Swanson. *Mathematics into Type*. American Mathematical Society, 1999. An set of old notes, updated in 1999 by Arlene O'Sean og Antoinette Schleyer.