Introduction

In my \LaTeX\ 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 dlftxbo
codetips package. The package is published on CTAN and the package is released under the normal lppl license.
The »dlfltxb« part of the name simply stands for »daleif« and \texttt{\LaTeX} book. The \texttt{dlfltxbcodetips} package is the first package in the »dlfltxb«-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 \texttt{mathtools} package by Morten Høgholm, but he is quite busy elsewhere at the moment.

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

# Extra \texttt{\dbx} will often be used to simulate some text or mathematical material.

## A big symbols

\texttt{\bigtimes}  

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

\[
\bigtimes_{n=1}^k A_n
\]

## Negated up- and downarrows

\texttt{\nuparrow} and \texttt{\ndownarrow} by rotating and reflecting \texttt{\rightarrow} and \texttt{\leftarrow} respectively.\textsuperscript{1}

\[
A \nuparrow B \quad B \ndownarrow C
\]

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

\texttt{\usepackage[noamssymb]{dlfltxbcodetips}}

to disable the autoloading of \texttt{amssymb}, and remember to load \texttt{dlfltxbcodetips} after \texttt{mathdesign}.

# Fun with theorems

## Shaded or framed theorems with the \texttt{ntheorem} package

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

\textsuperscript{1}Updated version due to Enrico Gregorio.
initialises \theoremframecommand to do nothing.

Caveat. The theorem environment constructed will not have a stared companion as ntheorem 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}
{\begin{enumerate}
{\item \dbx[2cm]}
{\item \dbx[2cm]}
{\item \dbx[2cm]}
{\end{enumerate}
{\end{sthm}
{\begin{slemma}
{\begin{enumerate}
{\item a lemma.
{\item a lemma.
{\end{enumerate}
{\end{slemma}
{\begin{prop}
{\begin{enumerate}
{\item a theorem with no background.
{\end{enumerate}
{\end{prop}

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.\footnote{Depends on the configuration.}

\begin{thm}
{\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}

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

The macro \InsertTheoremBreak helps.

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

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

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

Simply hold back one column of alignment for the arrow:

```
\begin{alignat}{2}
& f(x) = \dbx[2cm] \quad (1) \\
\ArrowBetweenLines & g(x) = - \dbx[2cm] \quad (2) \\
\ArrowBetweenLines\Downarrow & h(x) = 0 \quad (3)
\end{alignat}
```

Note the height of the arrow line and that the line automatically does not contain equation numbers. The stared 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] \quad (4) \\
\ArrowBetweenLines* & g(x) = - \dbx[2cm] \quad (5) \\
\ArrowBetweenLines*\Downarrow & h(x) = 0 \quad (6)
\end{alignat}
```

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*).

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

The macro \SwapDeadSpace can simulate this feature.

\[ a=b \]
\begin{align*}
\SwapDeadSpace 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{equation}
\begin{array}{c}
\text{=}
\hline
\text{=}
\end{array}
\end{equation}

or

\begin{equation}
\begin{array}{c}
\text{=}
\hline
\text{=}
\end{array}
\end{equation}

or

\begin{equation}
\begin{array}{c}
\text{=}
\hline
\text{=}
\end{array}
\end{equation}

and

\begin{equation}
\begin{array}{c}
\text{=}
\hline
\text{=}
\end{array}
\end{equation}
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, after which 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] \\
& \MoveEqLeft = \dbx[5cm] \\
& \MoveEqLeft = \dbx[5cm]
\end{align*}

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] \\
& \phantom{=} \dbx[5cm] \\
& \phantom{=} \dbx[5cm] \\
& \phantom{=} \dbx[5cm] \\
& \phantom{=} \dbx[3cm]
\end{align*}

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

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

\MathIndent \MathIndent
\SetMathIndent \SetMathIndent{(math code)}
\AddtoMathIndent \AddtoMathIndent{(math code)}
\PopMathIndent \PopMathIndent
\PopMathIndent*

\MathIndent is used to set a space corresponding to the current indentation saved on
the stack. \SetMathIndent 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 it 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:

\begin{align*}
\dbx = {} & \SetMathIndent{\dbx[1cm] \Bigl[} \dbx[6cm] \\\n& \MathIndent + \dbx[7cm] \\\n& \MathIndent \AddtoMathIndent{{} + \dbx \Bigl\{} \dbx[2cm] + \Bigl(} \dbx[4cm] \\\n& \MathIndent + \dbx[4cm] \Bigr) \\\n& \PopMathIndent + \dbx[6cm] \Bigr\} \\\n& \PopMathIndent + \dbx[6cm] \Bigr]
\end{align*}

\begin{align*}
\dbx = \begin{bmatrix}
\mathbf{0} & \mathbf{1} & \mathbf{0} \\
\mathbf{1} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{1}
\end{bmatrix}
\end{align*}

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\[\langle options\rangle\{\langle identifier\rangle\}

By default it can be used as
\DeclareMathSet\{R\}% => \numbersR = \mathbb{R}

But we have several options to change things. Options \langle key\rangle=\langle value\rangle style, note that \langle identifier\rangle is what is given to the formatting macro).

- **format** the macro used to format the output, default: \mathbb
- **name** this defaults to \langle identifier\rangle, 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 pfgpages package

The macro \ProvidePGFPagesFourOnOneWithSpaceForNotes will activate a \fourononewithspacefor_notes layout to be used with the pfgpages package. It is basically the same as the \eightononone layout but leaving the second column empty for reader to fill in their own notes.
5.2 Overloading \_ in math-mode

Placing \OverloadUnderscoreInMath in the preamble will overload the \_ character in math-mode such that
\[
X_{\_ab} = X_{\_\max}
\]
that is \_...\_ is the same as _\{\textup{...}\}.

Bibliography
