HA-prosper package

Documentation

Version 4.21

Hendri Adriaens
http://stuwww.uvt.nl/~hendri/downloads/haprosper.html
Center for Economic Research
Tilburg University, the Netherlands

August 25, 2004

Contents

List of Tables

List of Figures

1 General information

This manual provides information about working with the HA-prosper package. This package is a patch for the prosper class. Be sure to also study the documentation of the prosper class and the hyperref package. You can find this documentation for instance at the CTAN site for prosper [16], at the prosper website [17] or in your local LaTeX distribution (for the MiKTeX distribution [12], the documentation can be found in the `doc` directory in the MiKTeX root). Information about prosper can also be found at the WikiProsper site [24].
If you want to start working with HA-prosper immediately, have a look at the examples (see section 5.1). If you are interested in how HA-prosper actually works, have a look at the first lines of the source code available in the `Source` directory of the package or the `source` tree of your LaTeX distribution which explains how to generate the source code documentation.
Note that from version 4.1, HA-prosper uses the xkeyval package, which is available from CTAN [26] and is included in the MiKTeX distribution. Do not forget to install this package. See also section 5.2.
Do not forget to check the HA-prosper website [8] from time to time to see if there are updates for this package or new styles available. If, *after reading this manual*, you have questions or comments about HA-prosper, see section 5.6.

1.1 Upcoming changes

This is the last release of the package HA-prosper. All future developments in the line of this package will be collected in a new class called TeXciting. The reason that this package will be converted into a class is that some ideas for improvements (like A4 paper support) can only be realized when stepping away from prosper. The conversion will take some time and any bugs in HA-prosper will be dealt with in the meantime. Changes necessary for presentations to step from HA-prosper to TeXciting will be kept to a minimum. Don't forget to look for the class!

1.2 Compatibility

This section discusses compatibility issues between HA-prosper and some other packages. The HA-prosper package has been created and tested in an environment of MiKTeX 2.3 [12] and GhostScript 8.00 [3] and newer versions of that software.

1.2.1 prosper

Since HA-prosper is a patch for prosper, it is important that you have the right version of prosper which is 1.5 (`prosper.cls` version 1.24)[1]. This is the CTAN version [16].

1.2.2 ppr-prv

A very nice package in combination with HA-prosper is the 'Prosper Preview' class, ppr-prv [15], created by Mathieu Goutelle [11]. The aim of this class is to produce a printable version of the slides written with prosper or HA-prosper with two slides (or more at will) per page and a table of contents on the first page in case you

[1] This package is also compatible to the CVS version 1.25 of prosper, but that version is still in beta stage and it is advisory not to use it. If you still want to use version 1.25 because of vTeX support, delete or comment line 88 of `prosper.cls` then, namely were the pstcol package is loaded, since that causes colors not to be defined properly.

are using HA-prosper. Table 1 shows which versions of ppr-prv and HA-prosper are compatible. Each row of the table lists the compatible versions.

Table 1: Compatible versions of HA-prosper and ppr-prv.

HA-prosper	ppr-prv	HA-prosper	ppr-prv
3.7x	0.09	3.8	0.10
3.9	0.11	4.0	0.12
4.11	0.13	4.2	0.13

1.2.3 semcolor/pstcol/pstricks/color/xcolor

Recently, the incompatibilities of these packages have been solved and through a modification of semcolor, the improved compatibility is also available to HA-prosper users. Check the documentation of these packages to see whether or not your LaTeX distribution already includes the compatible versions. If so, then you can use both pstricks color commands in your presentation or style as well as xcolor commands.

1.2.4 babel

HA-prosper redefines the `itemize` and the `enumerate` environments at the beginning of the document. This solves a bug that would otherwise occur when using the babel package with the `french` option. The prosper macro `\NoFrenchBabelItemize` is obsolete in HA-prosper and has no function anymore. The order of loading the packages is important since HA-prosper attempts to redefine some babel macros. In case you want to use babel, load babel first and afterwards HA-prosper.

1.2.5 Miscellaneous

Older versions of GhostScript might produce ugly display text and logo's. However, prints are fine in general. Furthermore, it has been shown that Acrobat Reader 4 [2] generally does not display pictures properly.
Since the `overlays` macro is not recognized by Scientific WorkPlace [19], this program is not supported. The advice is to use an editor for creating the slides, for example WinEdt [25] or TexNicCenter [23] (free).

1.3 Installation

This package is included in the MiKTeX distribution [12] and in TeXLive [22] and can be installed for you by these programs. However, if you downloaded this package from the HA-prosper website [8] or from the HA-prosper CTAN directory [7]), then here are some hints for installing the package into your LaTeX distribution.
The package includes pre-generated run and doc files, but you can reproduce them from the source if necessary. See the first lines of `HA-prosper.dtx` for information how to do this. The files in `Doc` should go into the `doc` tree of your distribution, the files in `Run` into the `tex` run tree and the files in `Source` into the `source` tree. See the documentation of your LaTeX distribution for information on installing HA-prosper into your LaTeX distribution or the TeX Frequently Asked Questions [20].
If you choose not to install the package into your LaTeX distribution, you can copy the necessary files (that is, the run files of HA-prosper in the `Run` directory and

the necessary files of the particular style, which can all be found in the relevant subdirectory of the `Run` directory) into your working directory.

1.4 Document set up

The structure of a presentation created with HA-prosper is comparable to that of a prosper presentation. See the prosper documentation for an overview.
Do not forget to include

```
\usepackage[style,options]{HA-prosper}
```

directly after the prosper document class command to be able to use the new definitions provided in `HA-prosper.sty`. Here `style` is the style you want to use, for instance `HA`, and `options` are any package options which will be described in section 2. A summary of these options can be found in section 1.5.

`\HAPsetup` You can set up certain features like footers and types of stepping environments globally (holding for the entire presentation, unless specified otherwise locally) using `\HAPsetup`, for instance

```
\HAPsetup{rf={My presentation},trans=Dissolve}
```

This command can set up the features that you will be discussed in section 2. You can find a summary in section 1.6.

1.5 Summary of package options

Table 2 contains a summary of all package options of HA-prosper. The numbers in between brackets are the sections where you can find more information about a specific option. See section 1.4 how to use these options.

Table 2: HA-prosper package options.

`sounds`	Turns on the `sound` key for slide environments (2.1).
`toc`	Creates a table of contents on slides (2.2).
`highlight`	Highlights the current slide in the table of contents (2.2.2).
`hlsections`	Highlights the current section on every slide within the section (2.2.2).
`portrait`	Creates portrait slides (2.7).
`notes`	Includes the notes in your presentation (2.8).
`notesonly`	Includes only the notes in your presentation (2.8).
`slidesonly`	Includes only the slides in your presentation (2.8).
`blackslide`	Includes a black slide at the start of the presentation (2.11).

1.6 Summary of global options

Table 3 lists all the features of HA-prosper that can be set using `\HAPsetup`. The numbers in between brackets are the sections where you can find more information about a specific feature. See section 1.4 how to use these global options.
The features listed in table 3 all have a default value, which are discussed in the relevant sections. These values are defined in the file `HA-prosper.cfg`. This file can be modified. Modifications will have an effect on all presentations created using HA-prosper[2].

[2]Note that the defaults saved in the `HA-prosper.cfg` file at the moment of installation are also saved in `HA-prosper.sty`. In case you accidentally delete a line in the `HA-prosper.cfg` file, the predefined defaults in `HA-prosper.sty` will be used.

Table 3: Global HA-prosper options for `\HAPsetup`.

`lf`	Left footer (2.12).	`sound`	Transition sound (2.1).
`rf`	Right footer (2.12).	`template`	Template for parts (2.3.2).
`sn`	Slide number layout (2.12).	`stype`	Stepping environment type (2.5.1).
`tsnav`	Title slide navigation (3.3).	`sstart`	Starting overlay (2.5.1).
`nsnav`	Normal slide navigation (3.3).	`iacolor`	Inactive color (2.5.1).
`trans`	Transition effect (2.1).	`counters`	Protect custom counters (2.6).

1.7 Compilation

If you use the `pdf` option in the `\documentclass`, compile with LaTeX2DVI - DVI2PS - PS2PDF to create slides with animations (incrementally displayed items created by the `itemstep` environment) that can be projected using for instance Acrobat Reader [2]. Notice that HA-prosper is build to work with papertype `letter` in the DVI2PS step (which is the default in the MiKTeX distribution and which can be achieved by adding `-t letter` to the command line of `dvips`), but the `Fyma` style supports `A4` paper as well. See section 3.4.
If you want to print the slides, use the `ps2pdf` option and compile them with LaTeX2DVI - DVI2PS - PS2PDF and you can print the slides without animations. You can also use the Acrobat Distiller (included in Acrobat [1]) to create the PDF from the PS file. This creates very efficient PDF files. Also see section 5.3. Do not forget to specify the prosper class option 'distiller' in this case to avoid errors when distilling the PS document. For instance

```
\documentclass[pdf,distiller]{prosper}
```

If you prefer to use GhostView [5] for printing purposes, you can of course suffice with LaTeX2DVI - DVI2PS and print the slides form GhostView[3].
A nice option for the 'documentclass' is 'draft'. This is provided by prosper and speeds up the process of compiling and viewing the presentation by leaving out images. Deleting the option produces a presentation that includes all images again. See the prosper class documentation for more details.

1.8 Upgrade information

This section contains information how to change your presentation in case you worked with an earlier version of HA-prosper before and want to upgrade it. The list below contains the hints for upgrading with a single version step at a time. If you for instance want to upgrade from version 3.5 to the latest version, read all the entries in the list starting from the entry right the entry of version 3.5. If a particular version is not mentioned, this means that no changes need to be made to adapt the presentation to that version when starting at the last version before.

3.0 From HA-prosper version 3.0, the package is based on the CTAN version of prosper. This means that a separate version of prosper is not distributed with this package anymore. If you used HA-prosper before, replace the old style-file with the new one to upgrade to the latest version of HA-prosper and do not forget to rename or delete prosper.cls in the working directory.

3.5 From version 3.5 of HA-prosper the style definitions (template) will be loaded by HA-prosper, not by prosper anymore. This is done to avoid conflict when

[3] You might have to shrink pages to fit the paper when printing the slides.

trying to use HA-prosper templates with prosper. Delete the style from the `\documentclass` command and put it in the `\usepackage` command of HA-prosper when updating HA-prosper. See also section 1.3.

3.6x
- When upgrading to HA-prosper version 3.6, you might need to delete the `.toc` file of an existing presentation first since table of contents commands have changed and the new HA-prosper is not able to understand the old commands.
- The `\email` and `\institution` commands have changed. In earlier version, defining them in the preamble of your presentation would make them appear on the title slide. Now, you have to put them inside the `\author` command. See section 2.13 for an example.

3.7x
- Download the full package to use the latest versions of the styles that are compatible to this version of HA-prosper.
- Rename `\section` and `\sectionandpart` to `\tsection` and `\tsectionandpart` respectively.

3.8
- Download the full package to use the latest versions of the styles that are compatible to this version of HA-prosper.
- Rename `\figureitem` to `\hiddenitem`. See section 2.5.3.
- Change slides that use the `dualslide` environment to for instance a normal `slide` and use the `\dualslide` macro to create a dualslide on that slide. See section 2.4.

3.9
- `\figureitem` and `\putfigureitem` are no longer supported; `\putfigure` should be replaced by the more general `\topbl`, see section 2.5.3.
- The dualslide dimensions have changed a bit, see section 2.4.
- More optional arguments for the `itemstep` environment are available now, see section 2.5.1.

4.0
- Commands `\TitleSlideNav`, `\NormalSlideNav`, `\LeftFoot` and `\RightFoot` are obsolete but still supported. You can use `\HAPsetup` instead, see section 1.6.
- Some style commands have changed but backward compatibility is provided, see section 4.
- `\onSlide` and `\OnSlide` are provided but the prosper macros `\fromSlide`, `\FromSlide`, `\untilSlide`, `\UntilSlide`, `\onlySlide` and `\OnlySlide` still work, see section 2.5.4.
- Toc behavior has changed a little bit. Empty slide titles will now produce empty toc entries and empty bookmarks. Deleting these happens by using respectively `toc=` and `bm=` (see section 2.2).

4.1x
- HA-prosper now uses xkeyval [26]. Install this package to work with HA-prosper.
- `\topbl` has been deleted. The macro can be found on the TEX FAQ [21].
- Replace `\item` by `\xitem` (see section 2.5.2).
- Replace `\hiddenitem` by `\xitemwait` (see section 2.5.3).
- When using `Lakar` style, replace `emptyslide` environments by `partslide` environments (see section 3.2).

4.2x No changes to presentations necessary.

2 New or changed environments and commands

This section describes the new commands provided by HA-prosper and the originally prosper commands that have been extended. The section assumes knowledge of the prosper documentation.

2.1 The slide environment

`slide` A slide is created using the `slide` environment. For instance

```
\begin{slide}[trans=Replace]{Sample slide}
...
\end{slide}
```

The new `slide` environment has four options. They are listed in table 4. Some

Table 4: Slide environment options.

`toc`	When specified, it will be used for the table of contents. When not specified, the slide title will be used to create a toc entry (see section 2.2).
`bm`	When specified, it will be used to create a bookmark. If not, the slide title will be used as a bookmark.
`trans`	Slide transition effect (see the prosper documentation) Default: `Replace`.
`sound`	Specify a sound file to be played when the slide is displayed.

remarks need to be made. First of all, when you supply an empty string as bookmark (so `bm=`), the bookmark will not be created. A similar thing holds for empty toc strings (`toc=`). See section 2.2. The bookmark option is supplied to provide low level text for creating a bookmark in case you want the slide title to contain LaTeX commands which might not be convertable to bookmarks[4]. For instance:

```
\begin{slide}[bm=Important!]{\red Important!}
```

`sounds` Second, adding transition sounds to a presentation is possible, but it is important to know that the sounds will not be embedded in the presentation and hence need to be supplied separately. Further, the presentation becomes system dependent by using sounds. It depends on the target machine if the specific sound file can be played or not. To switch on the use of sounds in your presentation, use the package option `sounds`, see section 1.5.
The options `trans` and `sound` can be set globally by `\HAPsetup` (see section 1.6)[5]. The `sound` slide option adds the transition sound to all slides within `\overlays`. For more flexibility, the sound option also has a user interface to set sounds on
`\SlideSound` pages directly. The sound file `sound.wav` is added using `\SlideSound` as follows.

```
\SlideSound{sound.wav}
```

This code adds a transition sound to the first overlay only:

```
\onSlide*{1}{\SlideSound{sound.wav}}
```

(see also section 2.5.4 for the use of `\onSlide`).

[4] Note that the bookmarking procedure uses `\pdfstringdef` of hyperref which is able to process things like `\"i`.

[5] Notice that backward compatibility for `\DefaultTransition` is provided.

2.2 Structuring the presentation

2.2.1 Building a table of contents

`toc` You can include a table of contents on each slide. This can be used for navigational purposes during the presentation. When printing the slides, you might decide to leave out the table of contents. You can request a table of contents by specifying the option `toc` in the `\usepackage` command for the style file, like

```
\usepackage[toc,HA]{HA-prosper}
```

(see also the examples in this distribution). HA-prosper will use slide titles by default to produce the table of contents, but this behavior can be changed (see section 2.1).
Some remarks. First of all, the table of contents will of course only be put on the slides when it has been implemented in the style that you are using. See section 3.4. Second, HA-prosper offers possibilities for style developers to split the table of contents into two parts the first of which will contain the sections and the second will contain the content of the current section.
Third, if you specify an empty string as toc entry (so `toc=`), the toc entry will be deleted from the table of contents. This allows for building an overview of your presentation. Hence the table of contents can both be used for navigational purposes (like the bookmarks) and to provide an overview of the presentation.
Finally, as with a usual table of contents in LaTeX, you need to run your presentation at least twice after altering slides or toc entries. On the first pass, the table of contents will be updated and on the second run, the new table of contents will be included in the presentation.

2.2.2 Highlighting table of contents entries

`highlight` If you want to highlight the current slide or the slide created by a `\part` or `\tsectionandpart` command (see section 2.3) in the table of contents, specify the `highlight` option, like

```
\usepackage[toc,highlight,HA]{HA-prosper}.
```

Of course, the `highlight` option only takes effect when also the `toc` option has been specified (and of course the style you are using should support highlighting and the table of contents). This `highlight` option can be nice for both yourself and the audience since everyone can clearly see on which slide or in which section of the presentation you are.

`hlsections` An extra option is the `hlsections` option. This will make the current section highlighted on every slide within the current section. This can be especially useful when sections and section content are split in the style that you use, which would otherwise result in unclarities about the current section[6]. When the table of contents is not split up, this option might be left out since in this case, the hierarchical structure of the table of contents clearly identifies the current section.

2.3 Parts and sections

2.3.1 tsection

`\tsection` By using the `\tsection` command, it is possible to divide the table of contents in several sections. The content of a section will be hidden until you access a slide inside this section. The section heading in the table of contents itself refers to

[6] Splitting of the table of contents is often implemented on portrait templates.

the first slide after the section command. The section command should be used between slides. The following shows an example.

```
\begin{slide}{First slide}
Some text
\end{slide}

\tsection[Bookmark for section 1]{Section 1}

\begin{slide}{Second slide}
Some text
\end{slide}
```

The optional argument of the `\tsection` macro can be used to specify the string to be used for the bookmark. Notice that all slides that are met before issuing the first `\tsection` command will always be displayed in the table of contents. Exceptions hold for two cases. The first is when issuing an empty string as toc entry in the `slide` environment (see section 2.1). The second is when the style that you are using implements a split table of contents. In this case, the content of every new section will replace the content of the old section when going from the old section to the new section. The entries before the first section will then be replaced by the entries of the first section when advancing to that section.

`\tsection*` Furthermore, `\tsection*` is available. It has the exact same possibilities as `\tsection`, but this command hides the section created by `\tsection*` completely until you enter it (for instance with a hyperlink to a slide, by using bookmarks or by browsing the slides). This can be useful if you want to include slides in your presentation, that you don't want to show in the table of contents, but which you could include into the presentation if you see that you have some time left. But there are of course other applications thinkable.

2.3.2 part

`\part` The `\part` command lets you identify separate parts in your presentation. The command `\part{Part I}` produces a normal slide with horizontally centered the words 'Part I'. The argument will also be used to create a bookmark and a toc entry (if a table of contents is requested). The `\part` takes the same options as the `slide` environments (see section 2.1), but it should be noted that in case you specify an empty string for the `bm` option, the part title will be used to create the bookmark.
An additional option for the `\part` command is the `template` option. This allows to select a different template (supplied by the style that you use) for the slide created by the `\part` command. For instance, the HA-style supplies the template 'wideslide'. You can use that template for the part to delete the table of contents from that particular slide. Example:

```
\part[template=wideslide,toc=Part number one,bm=The first part]{Part I}
```

Note that the option `template` can be set globally for the entire presentation (see section 1.6). The default value of `template` is `slide`.

2.3.3 tsectionandpart

`\tsectionandpart` The `\tsectionandpart` command is a combination of the `\tsection` command (see section 2.3.1) and the `\part` command (see section 2.3.2). This command is supplied since issuing the two previous commands after each other cannot replicate the output of `\tsectionandpart`. The command has the same optional arguments

as the \part command and besides creating a slide, it also creates a section in the table of contents. The section heading in the table of contents will now refer to the slide created by the \part command.

\tsectionandpart* Notice that \tsectionandpart* produces similar output as \tsection*. Namely, it hides the section in the table of contents completely until it is accessed by going to any slide within this section. See also section 2.3.1. As \tsectionandpart, it also creates a \part slide.

2.4 Dualslide

\dualslide The \dualslide macro allows to split content into two columns. Usage:

```
\dualslide[opt1][opt2][opt3]{options}{left}{right}
```

The parameters are explained in table 5.

Table 5: Dualslide parameters.

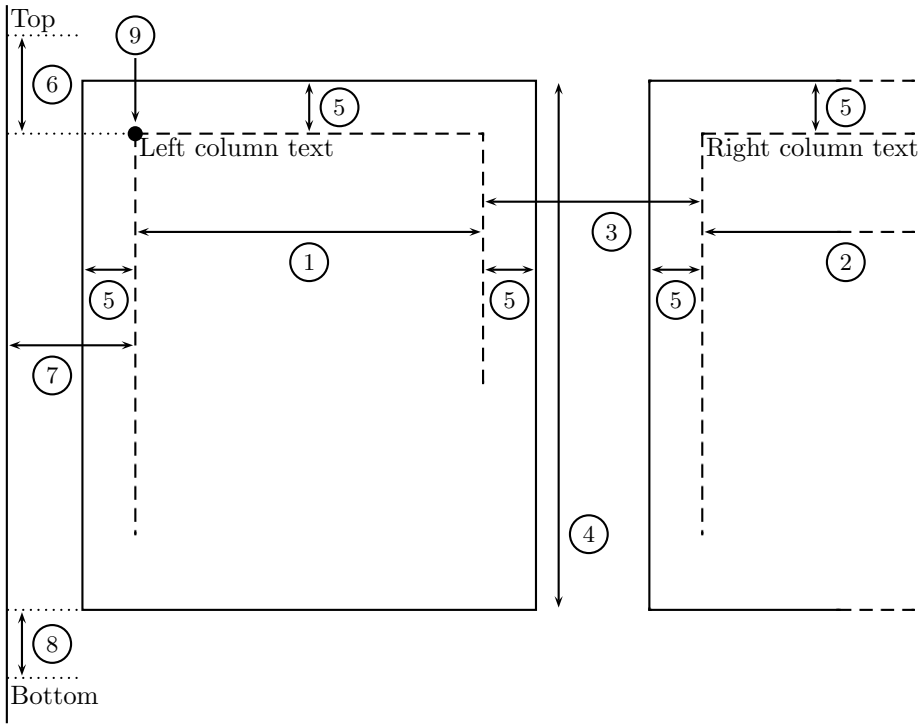
opt1	Any PSTricks options to change the style of the (optional) left frame. Example: linestyle=dotted.
opt2	Idem for the line in between the two columns.
opt3	Idem for the right column.
options	Options for the dualslide, see table 6.
left	The content for the left column.
right	The content for the right column.

Table 6: Dualslide options.

lineheight	If lineheight is specified, a line of the specified height will be created using \psline in between the two columns. Example: lineheight=6cm.
lfrheight	Creates a frame of the specified height around the left column.
rfrheight	Creates a frame of the specified height around the right column.
frsep	Space between text and the frames. Default: 1.5mm.
colsep	Space between the two columns. Default: 0.06\linewidth.
lcolwidth	Width of the left column. Default: 0.47\linewidth.
rcolwidth	Width of the right column. Default: 0.47\linewidth.
topsep	The extra space (additional to \baselineskip) between text above the columns and the text within the columns. Default: 0cm.
bottomsep	Idem for the bottom of the columns. Default: 0cm.
indent	Horizontal indent left to the left column. Default: 0cm.

The dimensions described in table 6 are represented graphically in figure 1. Important to notice is that the \dualslide macro uses the current cursor position as the reference point to position the first line of text of the left column (see also figure 1). This means that optional frames can extend to the text on the previous line. Use for instance topsep=0.3cm in that case to add extra space between the two lines of text. The default value of topsep is based on the situation that there is no text on top of the two columns. In that case, it is best to locate the first line of text of the left column at the same spot as text that is not created by \dualslide on other slides. The setting topsep=0cm does exactly this. However, with a combination of topsep and indent you can change this behavior and position the first line of text of the left column anywhere you want.

Figure 1: Dualslide dimensions.



Meaning of the numbers

1	`lcolwidth`	5	`frsep`
2	`rcolwidth`	6	`topsep`
3	`colsep`	7	`indent`
4	`lfrheight`, `rfrheight`, `lineheight`	8	`bottomsep`
		9	Reference point

The dualslide macro computes the height of the construction to position text below the construction correctly. The computation is done by taking the maximum height of `lfrheight`, `rfrheight`, `lineheight` (if requested) and the left and right column content. Hence when frames nor a line is requested, `bottomsep` is the vertical space between the lowest line of text in the columns and the text below the columns (additional to `\baselineskip`).
Furthermore, if none of the optional arguments has been specified, `dualslide` will use the PSTricks defaults for creating frames and the line. Two exceptions: `linewidth=.25pt` and `linecolor=HAP@framecolor` by default, but these can of course again be changed using the optional arguments. When only `opt1` has been specified, this is used globally for both frames and the line. When both `opt1` and `opt2` have been specified, `opt1` will be used for both frames and `opt2` will be used for the line. In case you want to change the properties of the right frame only, call for instance

```
\dualslide[ ][ ][linecolor=red]{rfrheight=6cm}{left content}{right content}
```

More examples can be found in the example files of HA-prosper, see section 5.1. Also have a look at section 2.5 for information about animating content. The commands described there can also be used to animated content in the two columns.

2.5 Animating content

2.5.1 itemstep and enumstep

itemstep
enumstep

The `itemstep` and `enumstep` environments are based on the `itemize` and `enumerate` environments respectively, but with these environments, items will be displayed incrementally. Available options can be found in table 7. These environments can be nested into each other and lower level environments inherit options from higher level environments. The `sstart` option has effect only in the highest level stepping environment (which can be embedded in an `itemize` or `enumerate` environment). The items of an `itemize` environment, which is nested in a stepping environment, will inherit the state (active or not) from the item in the stepping environment to which the (sub)environment(s) belong to. If `itemize` or `enumerate` is used at the top level, all items are active on all overlays. Find an example below (for the meaning of `\xitem`, refer to section 2.5.2).

Table 7: Itemstep and enumstep options.

`sstart`	Starts the stepping environment on overlay `sstart`. Default: 1.
`iacolor`	Inactive color, used for inactive items. Default: lightgray.
`stype`	Type of stepping environment. Default: 0. Possible: 0 – items appear one by one, but remain active; 1 – items appear one by one, but turn inactive when moving to the next item; 2 – all items are displayed and only one is active at a time.

```
\overlays{6}{
\begin{slide}{xitem example}
Some text.
  \begin{itemstep}[sstart=2,stype=0,iacolor=gray]
    \xitem item 1
    \begin{enumstep}[stype=1] % inherits iacolor!
      \xitem item 1.1
      \xitem item 1.2
    \end{enumstep}
    \xitem item 2
    \xitem item 3
  \end{itemstep}
\end{slide}
}
```

See section 5.1 for more examples and section 2.5.3 for a tool for these environments. The options for the stepping environments can also be set globally using the `\HAPsetup` macro (see section 1.6).
Note that the use of type 1 or 2 stepping environments requires the definition of the color `HAP@textc` by the template that you use. See section 4.

2.5.2 xitem

\xitem

HA-prosper defines a new command, `\xitem`, to be used within `itemize`, `enumerate`, `itemstep` and `enumstep` environments. Within `itemstep` and `enumstep` environments it will take care of the animation effect discussed in section 2.5.1. Within `itemize` and `enumerate` environments it will just display the content. Defining this command is necessary since redefining `\item` would damage macros and environments using `\item` (for instance `center`, `flushright`, `flushleft`).

The `\xitem` macro also has a version in which it takes an optional value n by doing `\xitem<n>`. This value is the extra number of overlays that the item should remain active in a stepping environment and is ignored in an `itemize` or `enumerate` environment. This optional argument can be useful when you want to add content to the current item on a later overlay. Notice that this can make several items active at a particular overlay. To avoid this, add `\xitemwait` before the next item. See the example below and section 2.5.3.

```
\overlays{3}{
\begin{slide}{xitem example}
\begin{itemstep}[stype=1]
  \xitem<1> Yes?
  \onSlide{2-}{No!}
  \xitemwait
  \xitem Because...
\end{itemstep}
\end{slide}
}
```

Note that a possibly unfamiliar syntax has been chosen for `\xitem<n>`. This done to easily support the other optional argument that comes with `\xitem` since it is an extension of the standard `\item`. The optional argument allows you to overwrite the item label, for instance `\xitem[a]` or `\xitem<1>[b]`. See your favorite LaTeX manual for more information about the latter.

2.5.3 xitemwait

`\xitemwait` The command `\xitemwait` tells HA-prosper to wait for one overlay before passing on to the next item in a stepping environment. The optional argument n tells HA-prosper to wait n overlays before passing on to the next item. Example:

```
\overlays{5}{
\begin{slide}{xitemwait example}
  \begin{itemstep}
    \xitem 1
    \xitem 2
    \xitemwait[2]
    \xitem 3
  \end{itemstep}
\end{slide}
}
```

This might come in handy to animate custom content using `\onSlide` (see section 2.5.4) in dualslides (section 2.4) in combination with stepping environments (section 2.5.1). Outside stepping environments, the macro has no effect.

2.5.4 onSlide and OnSlide

`\onSlide` The macro `\onSlide{overlays}{material}` displays `material` on the overlays listed in `overlays`. This can be a comma delimited list and the syntax explained in table 8 can be used (x and y are positive integers with $y \geq x$). When used inside an `itemstep` or `enumstep` environment, there is extra syntax available. This syntax follows the same rules as explained in table 8, but now x and y can start with a `+` sign. This sign can be used to animate content relative to the preceding `\xitem` in an `itemstep` or `enumstep` environment. Suppose that for instance the preceding `\xitem` of `\onSlide{+1-+2}{...}` will be displayed on overlay 6. Then the material of `\onSlide` will be displayed from overlay 7 to 8. Since this construct

Table 8: Notation for `\onSlide`, `\onSlide*` and `\OnSlide`.

x	Display material only on overlay x.
-x	Display material until and including overlay x.
x-	Display material from and including overlay x.
x-y	Display material from x until y, including x and y.

is relative, it is very easy to add to or delete items from the list without having to adapt all of the `\onSlide` commands that you used. See another example below.

```
\overlays{4}{
\begin{slide}{onSlide example}
\begin{itemstep}
  \xitem item 1
  \xitem item 2
  \begin{itemize}
    \xitem item 3 \onSlide{+1-}{and item 4}
  \end{itemize}
  \xitemwait
  \xitem item 5
\end{itemstep}
\end{slide}
}
```

The `\onSlide` macro collects and generalizes all the functionality of the prosper macros `\onlySlide`, `\untilSlide` and `\fromSlide`. This macro advances the cursor position even if the material is not displayed. The stared version `\onSlide*` does not advance the cursor position in case the material is not displayed and collects the functionality of the prosper macros `\onlySlide*`, `\untilSlide*` and `\fromSlide*`. Note that these macros can be nested into each other, which allows for the creation of more parsimonious structures (which will also become more complex). For instance,

```
\onSlide{-3,5-6,8,10-}{One}\onSlide{-2,6,8,10-}{Two}\onSlide{2,8,10-}{Three}
```

and

```
\onSlide{-3,5-6,8,10-}{One\onSlide{-2,6-}{Two\onSlide{2,8-}{Three}}}
```

produce the same output. The following example demonstrates the difference between `\onSlide` and `\onSlide*`:

```
\onSlide{1}{One}\onSlide{2}{Two}\onSlide{3-}{Three}\par
\onSlide*{1}{One}\onSlide*{2}{Two}\onSlide*{3-}{Three}
```

`\OnSlide` The macro `\OnSlide{overlays}` also accepts the same syntax as the `\onSlide` macro and can be used to display or not all of the following material on a slide after the `\OnSlide` macro. This macro collects the functionality of the prosper macros `\OnlySlide`, `\UntilSlide` and `\FromSlide`. Example:

```
\OnSlide{3-} All of this material will be displayed as from overlay 3.
```

2.6 Numbering on overlays

Mathieu Goutelle [11] contributed a patch for the `\overlays` macro which makes it possible to number equations, tables and figures on overlays. This patch has been

extended such that you can also protect custom counters on overlays. For general information about numbering, see your favorite LaTeX book for more information. The `counters` option for `\HAPsetup` (section 1.6) can be used to list the counters that should be protected. For instance,

```
\newtheorem{mytheorem}{Theorem}
\newtheorem{myproof}{Proof}
\HAPsetup{counters={mytheorem,myproof}}
```

in your preamble, will let you safely do

```
\overlays{3}{
\begin{slide}{Proof}
  \begin{myproof}
    Suppose that:
    \begin{equation}
      x^2+y^2=z^2
    \end{equation}
    then\dots
  \end{myproof}
\end{slide}
}
```

The counters `figure`, `equation` and `table` are protected on overlays by default. The `counters` option for `\HAPsetup` lists the additional counters that should be protected. Notice that the counter that you want to protect needs to exist when issuing `\HAPsetup` to set the protected counters.

2.7 Portrait slides

`portrait` If a template supports portrait slides, you can access these by specifying the `portrait` option in the `\usepackage` command like

```
\usepackage[toc,portrait,HA]{HA-prosper}.
```

When requesting portrait slides, a template will give an error if it does not support portrait slides. See section 3.4 for the templates that support portrait slides. When using portrait slides, the `hlsections` option might be useful for you, see section 2.2.2.

2.8 Notes

`notes` The HA-prosper package provides a way to make notes in your presentation. To include notes in the presentation that go with a slide, start a `notes` environment after this slide. For instance

```
\begin{slide}{First slide}
...
\end{slide}

\begin{notes}{Notes for the first slide}
These are the notes for the first slide.
\end{notes}
```

`slidesonly` `notesonly` `notes` An additional option in the `\usepackage` command provides a way to control the printing of notes and slides. Include at most one of the following options.

- `slidesonly`: produces the slides only. This is the default option. Use this to create the presentation that you want to project.

- `notesonly`: produces the notes only.
- `notes`: includes the notes in the presentation.

For instance

```
\usepackage[toc,notesonly,HA]{HA-prosper}
```

Note: If you want the notes only, first run the presentation at least once with the option 'notes' to create correct labels for the notes page numbers. Once that is done, run the presentation with the 'notesonly' option to create the notes[7].

2.9 Bibliography

`thebibliography` HA-prosper redefines the standard article `thebibliography` environment to suppress the creation of a section heading and running headers. All other properties are maintained. Now you can do either of the next two (depending whether you are using BiBTEX or not):

```
\begin{slide}{slide 1}
\cite{someone}
\end{slide}
\begin{slide}{References}
\begin{thebibliography}{1}
\bibitem{someone} Article of someone.
\end{thebibliography}
\end{slide}
```

```
\begin{slide}{slide 1}
\cite{someone}
\end{slide}
\begin{slide}{References}
\bibliography{YourBib}
\end{slide}
```

In case you have a big reference list that you want to spread over multiple slides, have a look at the packages natbib and bibentry [14]. Using both packages allows you to do:

```
\begin{slide}{References (1)}
\nobibliography{YourBib}
\bibentry{someone1}
\bibentry{someone2}
\end{slide}
\begin{slide}{References (2)}
\bibentry{someone3}
\end{slide}
```

Have a look at your favorite LATEX documentation for more information about citations and bibliographies.

2.10 Bookmarks

HA-prosper nests bookmarks within sections created by `\tsection` and `\tsection*` (see section 2.3.1) and `\tsectionandpart` and `\tsectionandpart*` (see section 2.3.3). On the first pass of a document, bookmarks will be written to disk and will be read on the second pass to create the bookmarks. This is necessary to identify the number of slide bookmarks that should be nested within section bookmarks.

[7] In case you use landscape slides, do not forget to put Auto-Rotation of pages to 'Individually' if you are using the Acrobat Distiller to create the PDF. 'Collectively by File' will rotate will rotate notes to landscape just like the slides.

Notice that the bookmark of a `\tsection` (or `\tsection*`) points to the first slide within the section, just as the toc link generated by this section. Furthermore, as `\part` does not create a toc section, it does not create a bookmarks section. This allows for creating both a flat table of contents as well as a flat bookmarks list. The nesting of bookmarks in sections is fully compatible to the nesting of bookmarks of overlays subordinate to the bookmarks of the first overlay of a slide. Bookmarks created by section commands are open and the bookmarks of slides are closed by default. You can change the latter by issuing the prosper command `\collapsedBookmarksfalse` in the preamble of your presentation.

2.11 Black slide

`blackslide` The package option `blackslide` creates a completely black slide as the first slide of the presentation. The PDF-file will start at page 2 when opened, the first real slide, so the black slide will not be revealed. The slide has an embedded target called 'blackslide'. In the top left corner of the black slide, a black button is positioned which performs the Acrobat menu option 'GoBack' when clicking it. This allows to temporarily stop the presentation (for instance, to write something at the blackboard) by clicking a hyperlink that points to the target 'blackslide'. When the writing it done, click in the top left corner of the black slide to continue the presentation.

2.12 Footers

The footers can be defined by the `\HAPsetup` command (see section 1.6)[8], for instance

```
\HAPsetup{lf={Left footer},rf={Right footer}}
```

The default values of `lf` and `rf` are empty strings. The slide number layout can be specified by the `sn` option. The default value of `sn` is[9]

```
{-~p.~\thepage\ifallPages/\totalpages\fi}
```

which resembles the standard prosper layout. HA-prosper adds a space in between the text specified by `rf` and the text specified by `sn` to create the entire right footer. Another example of the slide number layout could be

```
\HAPsetup{sn={slide~\#\thepage}}.
```

2.13 e-mail, institution and the and-command

`\author` `\email` `\institution` HA-prosper changes the behavior of the prosper commands `\email` and `\institution`. The package does not anymore put them directly onto the title page. Instead, you can use them inside the `\author` command to use the fonts that the template specifies for the e-mail address and the institution. This is done to offer support for the `\and` command[10]. Example:

```
\author{%
  Me\\
  \institution{My institution}\\
```

[8] Notice that backward compatibility is provided for `\LeftFoot` and `\RightFoot`.

[9] `\thepage` contains the number of the current slide, `\totalpages` inserts the total number of slides in the presentation and `\ifallPages` is a conditional from prosper which is false if the prosper class option `nototal` is specified by the user.

[10] This was not possible in the construction used by prosper.

```
  \email{My e-mail address}
  \and
  You\\
  \institution{Your institution}\\
  \email{Your e-mail address}
}
```

3 Style specific features

This section documents several style specific features. Check table 9 in section 3.4 to find which features are supported by each template.

3.1 maketitle

`\maketitle` This adapted command replaces the usual `\maketitle` and takes care of defining a title slide with a different layout as the first slide, include the title on that slide and create the rest of the slides as normal slides (unless specified differently by using for instance another slide environment, see section 3.2).
This command has the same options as the `slide` environment (see section 2.1. By default, the title slide will appear in neither the bookmarks list nor the table of contents. You can change this by specifying respectively the `bm` option and/or the `toc` option (see section 2.1).
If the `\maketitle` feature is not supported by a style, then the `\maketitle` uses a normal slide to display the title on.

3.2 Wideslide and partslide

`wideslide` A `wideslide` produces a slide with an extended text area in case you need to display very wide formulas or figures. In a lot of templates, this means that the table of contents will not be displayed on a wideslide.
As a remark: the optional arguments possible for the `\maketitle` (see section 3.1) are also available for the modified slide environments. So

```
\begin{wideslide}[toc=Toc slide title,trans=Replace]{Slide title}
...
\end{wideslide}
```

creates a `wideslide` which does not have transition effects itself and uses 'Toc slide title' as the entry for the (optional) table of contents. See section 2.2 for more information about the optional table of contents.

`partslide` The `partslide` environment is available in some styles. This environment uses a template which has been designed especially for use with the `\part` (see section 2.3.2) and the `\sectionandpart` commands (see section 2.3.3).
See section 3.4 to find which styles support `wideslide` or `partslide` environments.

3.3 Hidden navigation

If you use Acrobat (Reader) to project the slides, it is possible to embed hidden navigational components. The navigational components are accessed by clicking the logo on a slide (unless stated otherwise in style specific documentation). These components are always hidden and hence removing their definitions when you want to print the slides is not necessary. Using `\HAPsetup` (see section 1.6), you can set the following options[11]:

- `tsnav`
  Navigation possibilities of the logo on the title slide.
  For instance: `tsnav=FullScreen`: when starting the presentation, you can make the presentation full screen by clicking the logo.

[11] Notice that backward compatibility for `\TitleSlideNav` and `\NormalSlideNav` is provided.

- `nsnav`
  Navigation possibilities on all other slides.
  For instance: `nsnav=ShowBookmarks` or `nsnav=GoToPage`: the former will provide a list of slide titles when viewing the presentation in full screen mode. In this case, proper slide titles might help you navigate your presentation. The latter will show a menu where you can type the slide number of the slide that you want to display.

For more possibilities, see the documentation of the hyperref package.

3.4 Support table

Table 9 displays an overview of the available extra features in the styles. In between brackets you can find links to sections explaining the features.

Table 9: Available features in styles.

Features \ Styles	Aggie	Capsules	CentER	Ciment	Fyma	HA	Lakar	Simple	TCS	TCSgrad	TCSTealBlue	Tycja
table of contents (2.2)	x	x	x	x	x	x	x	x	x	x	x	x
portrait slides (2.7)	x					x	x	x				x
A4 support					x							
modified `\maketitle` (3.1)	x	x	x	x	x	x	x	x	x	x	x	x
wideslide environment (3.2)	x	x	x	x	x	x	x	x	x	x	x	x
partslide environment (3.2)		x		x	x		x					
tsnav (3.3)	x		x						x	x	x	
nsnav (3.3)	x		x			x			x	x	x	

Currently only the `Fyma` style supports the `A4` papersize. To use it, issue the command `\FymaAFour` in your preamble and use `-t A4` in the `dvips` command line to get slides in A4 dimensions.

4 Information for style developers

4.1 General tools

This section provides some information in case you want to develop your own style. Remember that this package is still based on prosper, so for the standard prosper features, see the prosper documentation. Table 10 lists additional features of HA-prosper available to control the layout, fonts and colors of your style. The controls in table 11 are HA-prosper internals and should only be used in style files. In between brackets you can find links to sections explaining the related user interface commands. Do not forget to take a look at existing styles since these are the ideal starting point to create a new style. Note that if you use `\HAP@NSNav` or `\HAP@TSNav` in a style, make sure to check whether it has been defined by the user or not.

Table 10: HA-prosper features for style development.

`\FontLeftFoot{C}{BW}`,	This command is used to assign a font to the left footer (2.12). The first argument is for color slides, the second for black and white ones. Default: font: `\fontText`; text height: 5pt; line height: 5pt.
`\fontLeftFoot{xx}`	This command writes `xx` in the font defined by the former command.
`\FontRightFoot{C}{BW}`, `\fontRightFoot{xx}`	Idem for right footer (2.12). Default: font: `\fontText`; text height: 5pt; line height: 5pt.
`\FontAuthor{C}{BW}`, `\fontAuthor{xx}`	Idem for author (2.13). Default: font: `\fontText`.
`\FontInst{C}{BW}`, `\fontInst{xx}`	Idem for institution (2.13). Default: font: `\fontText`; text height: 7pt; line height: 7pt.
`\FontEmail{C}{BW}`, `\fontEmail{xx}`	Idem for e-mail address (2.13). Default: font: `\fontText`; text height: 7pt; line height: 7pt.
`\FontToc{C}{BW}`, `\fontToc{xx}`	Idem for table of contents (2.2). Default: font: `\fontText`; text height: 4pt; line height: 6pt.

Table 11: Internal HA-prosper controls for style development.

`\HAP@PutLF{A}{C}`	Uses `\rput` to position the left footer using anchor `A` and coordinates `C`.
`\HAP@PutRF{A}{C}`	Uses `\rput` to position the right footer using anchor `A` and coordinates `C`.
`\HAP@NSNav`	Contains argument of `\NormalSlideNav`, if defined (3.3).
`\HAP@TSNav`	Contains argument of `\TitleSlideNav`, if defined (3.3).
Color `HAP@textc`	Text color, used for switching back from inactive colors; default: black (2.5.1).
Color `HAP@framecolor`	Color for dualslide frames and lines; default: black (2.4).
`\ifHAP@active`	True if item is active, for use with `\myitem` (see prosper docs and 2.5.2).
`\ifHAP@highlight`	True if highlighting has been requested by the user (2.2.2).
`\ifHAP@portrait`	True if portrait slides have been requested by the user (2.7).
`\ifHAP@notes` `\ifHAP@notesonly` `\ifHAP@slidesonly`	If's that contain the state of requested notes, slides only or notes only. These can take the value 0 or 1 (false or true) and it holds that the sum is equal to one and the product is equal to zero (2.8).

4.2 Table of contents typesetting

HA-prosper uses the macros `\HAP@tocentry` and `\HAP@hltocentry` to typeset table of contents entries. The latter macro will be used in case the entry needs to

be highlighted. By default, HA-prosper puts the toc entry at hand in a \parbox to allow for multi-line toc entries. For \HAP@hltoctext, the latter is inserted in a \psframebox. You can redefine the former macros to typeset the table of contents for your style in a different way. Use \HAP@toctext in your definitions. This will at run time contain the entry at hand in the markup specified by either \HAP@tline, \HAP@tlineonly, \HAP@tsection or \HAP@tsectiononly depending on the type of table of contents and the type of the entry. See for instance the Fyma style for an alternative table of contents markup. See further table 12 for other macros related to typesetting the table of contents and section 2.2 for more information about the table of contents.

Table 12: Table of contents typesetting macros.

\HAP@toc	Command that includes the toc file in \fontToc.
\HAP@tsections	Includes sections of the toc file in \fontToc.
\HAP@tcontent	Contains the content of the current section.
\ifHAP@toc	True if a table of contents has been requested by the user.
\HAP@tline	Layout of a table of contents entry in a full toc.
\HAP@tlineonly	Layout of a table of contents entry in \HAP@tcontent.
\HAP@tsection	Layout of a table of contents section in a full toc.
\HAP@tsectionm	Section marker, inserted just before creating a section.
\HAP@tsectiononly	Layout of a table of contents section in \HAP@tsections.
\HAP@tsectionskip	Vertical skip above a section; default: 1.5em.
\HAP@titemskip	Vertical skip in between items, not sections; default: 0.1em.
\HAP@tocentry	Construct for entry, use \HAP@toctext inside, see leading text.
\HAP@hltocentry	Construct for highlighting, use \HAP@toctext inside, see leading text.
Color HAP@hcolor	Color for PostScript frame used for highlighting; default: black.
Color HAP@htcolor	Text color for highlighted toc entry; default: white.
\HAP@twidth	Width of toc, used for \parbox in \HAP@tocentry and \HAP@hltocentry.
\HAP@tborder	Border size in \psframebox of \HAP@hltocentry; default: 0.05cm

Note that the macros \HAP@hcolor, \HAP@htcolor, \HAP@twidth and \HAP@tborder are only used in the default definitions of \HAP@tocentry and \HAP@hltocentry. If you decide to redefine the latter two macros, defining and using the former four might not be necessary. The default toc typesetting uses these macros to provide an easy way for style designers to tweak the default markup of the table of contents.

4.3 Reserved commands

HA-prosper knows a couple of reserved commands. If these commands are defined by a style, they will be used by HA-prosper to create new slide environments. See table 13.

Table 13: Reserved commands in HA-prosper.

\HAPR@normalSlide	Should contain the layout of a standard slide.
\HAPR@wideSlide	Should contain the layout of a wide slide; used for `wideslide` environment (3.2).
\HAPR@partSlide	Should contain the layout of a part slide; used for `partslide` environment (3.2).
\HAPR@titleSlide	Should contain the layout of a title slide; used to create title slide, which will be embedded in the \maketitle command (3.1).

Note that `\HAPR@normalSlide` is used to switch back to after creating a special slide environment like `wideslide`. If this command has not been defined in your style, then HA-prosper will not be able to define the environments mentioned in section 3.2 for you. As a last note: don't forget the power of the PSTricks package for designing styles.

5 Miscellaneous issues

5.1 Examples

Examples that use the HA-prosper package are contained in the `HAPIntroduction.tex`, `HAPBigtest.tex` and `HAPDualslide.tex` files in this distribution. The examples assume that you have the `HA` style. More examples can be found at the HA-prosper website [8].
Do not forget to take a look at the documentation of the prosper class, the hyperref package and the PSTricks package to fully understand the possibilities of slides created by HA-prosper.

5.2 Required packages

The most important requirements of the prosper class are:

- seminar,
- hyperref,
- pstricks,
- graphicx.

The most important requirements of the HA-prosper package are:

- prosper (`prosper.cls` version 1.24),
- xkeyval [26],
- xcomment,
- verbatim.

5.3 Resulting PDF

The PDF produced by PS2PDF (GhostScript) is usually very big. Unfortunately, it includes a copy of the logos on the slides every time they appear. In case this logo is a rendered picture, the PDF will become very big. There is not much you can do about that. However, you can compress a generic presentation PDF file very much (often even to 10% of the original size) using for instance WinZip. And you could of course delete the logo or replace it by a non-rendered picture, for instance created by PSTricks, see the PSTricks website [18]. Compare for example the `HA` style and the `Lakar` style.
Another possibility is to use 'Reduce file size' in the file menu of Acrobat (available from version 6). You can also use the PDF Optimizer in this version of Acrobat (Tools - PDF Optimizer). Using this option, you can compress the pictures inside the presentation and make the presentation considerably smaller. Hint: do not use JPEG or JPEG2000 for the compression of images. Even at the highest level of quality, this produces poor results for projecting purposes. Instead, use the ZIP compression. Also, do not downsize the resolution lower than 300 pixels per inch (for retaining printing quality).
However, to get the most efficient PDF document, use the Acrobat Distiller to convert the PS file into a PDF file. Do not forget to include the 'distiller' option in the document class in this case (see section 1.7).

5.4 Credits

I want to thank the following people for their contribution(s).

- **Style contributors**
  Chris Ellison: `TCS`, `TCSgrad` and `TCSTealBlue` styles.
  Jack Stalnaker: `Aggie` style.

Laurent Jacques [10]: `Fyma` style.
Mathieu Goutelle [11]: `Capsules` and `Ciment` styles.

- **Beta testers**
  At this moment the beta testers are (in alphabetical order): Mark Davidson, Patrick Drechsler, Murali Krishnan Ganapathy, Mathieu Goutelle, Michael Hallgren and Mark Senn.

- **Bug reports**
  Bugs have been detected and reported by Chris Ellison, Victor Francisco Fonte, Johan Joubert, Frédéric Mayot, and Johan Segers.

- **Contributions and ideas**
  A lot of thanks go Mathieu Goutelle [11], Herbert Voß [9], Murali Krishna Ganapathy [13] and Laurent Jacques for great help, code contributions and/or ideas for improvements.

5.5 Contributions

Contributions are always welcome. If you want to submit a new style, template or piece of code, please make sure that it does not contain bugs. In case you want to contribute some code, section 4 might offer some useful code documentation. Please insert some documentation in case you added features that are not documented yet in this documentation. After submission of the template or code, it will be tested and if it is approved, it will be included in the HA-prosper package. For more information or if you have questions related to submitting your style or code, have a look at the contact information on my website [4].

5.6 Questions

If you have questions, please first consult the documentation of the package that could possibly answer your question. Probably, you can find the answer in the documentation of either the HA-prosper package, prosper class, the hyperref package or the PSTricks package. Also have a look at WikiProsper [24] for a prosper FAQ, prosper open questions, known prosper bugs, etcetera.
If you cannot solve the problem yourself, or if you have comments or requests for additional features, post a message to the HA-prosper mailinglist [6]. Do not forget to include a *minimal* example when you want help with an error message.

5.7 Development snapshots

Development snapshots are available from the HA-prosper website [8]. If you have comments or requests for new features, please check the page with contact information [4]. You can also contact me if you want to become a regular beta tester. Beta testers have great influence on the development process and are expected to spot bugs, but can also reject or propose new features.

5.8 Links

This section contains addresses of websites which have been referred to in this documentation.

[1] Acrobat. `http://www.adobe.com/products/acrobat`.

[2] Acrobat Reader. `http://www.adobe.com/products/acrobat/readstep2.html`.

[3] AFPL GhostScript. `http://www.cs.wisc.edu/~ghost`.

[4] Contact information. http://stuwww.uvt.nl/~hendri/personal/contact.html.

[5] GhostView. http://www.cs.wisc.edu/~ghost/gsview.

[6] HA-prosper mailinglist. http://listserv.surfnet.nl/archives/ha-prosper.html.

[7] HA-prosper on CTAN. http://www.ctan.org/tex-archive/macros/latex/contrib/ha-prosper.

[8] HA-prosper website. http://stuwww.uvt.nl/~hendri/downloads/haprosper.html.

[9] Herbert Voß. http://www.perce.de.

[10] Laurent Jacques. http://www.fyma.ucl.ac.be/~ljacques.

[11] Mathieu Goutelle. http://webperso.easyconnect.fr/goutelle.

[12] MiKTeX. http://www.miktex.org.

[13] Murali Krishna Ganapathy. http://www.cs.uchicago.edu/people/gmkrishn.

[14] Natbib on CTAN. http://www.ctan.org/tex-archive/macros/latex/contrib/natbib.

[15] ppr-prv. http://www.ctan.org/tex-archive/macros/latex/contrib/ppr-prv.

[16] Prosper on CTAN. http://www.ctan.org/tex-archive/macros/latex/contrib/prosper.

[17] Prosper website. http://prosper.sourceforge.net.

[18] PSTricks website. http://www.pstricks.de.

[19] Scientific WorkPlace. http://www.mackichan.com.

[20] TeX FAQ (Installing packages). http://www.tex.ac.uk/cgi-bin/texfaq2html?label=instpackages.

[21] TeX FAQ (Top-aligning imported graphics). http://www.tex.ac.uk/cgi-bin/texfaq2html?label=topgraph.

[22] TeXLive. http://www.tug.org/texlive.

[23] TeXnicCenter. http://www.toolscenter.org.

[24] WikiProsper. http://wikiprosper.bbclone.de.

[25] WinEdt. http://www.winedt.com.

[26] xkeyval on CTAN. http://www.ctan.org/tex-archive/macros/generic/xkeyval.

5.9 Copyright

5.10 Version history

This section lists the main changes in this package as from version 3.5. Note that each version change also contains solutions to known (minor) bugs and improvements in programming and user interface.

3.5 Solved a bug with toc entries in combination with highlighting. This bug was mainly caused by the keyval package not undefining variables when they are not used. Thanks to Mathieu Goutelle for the help; added support for the `\part` command. Made HA-prosper load slide style definitions itself at the final stage; changed style names to `HAP[name].sty` to identify HA-prosper style definitions from prosper style definitions; made HA-prosper undefine `\slidetitle` before loading the style definitions;

3.6 Improved writing of toc entries; solved an inconsistency in writing definitions between `seminar` and `latex.ltx`; introduced `\section` and `\sectionandpart`; completely changed `notes` environment, not using `seminar` note constructions anymore; added support for `\and` in `\author`; added `\fontToc`;

3.61 Removed redundant grouping;

3.62 Adapted writing definitions a bit to be more consistent with `seminar`;

3.7 Prepared HA-prosper for compatibility to the `prosper preview` class [15]; integrated specific style features into HA-prosper; recoded `\figureitem` command; renamed `\section` to `\tsection`, `\sectionandpart` to `\tsectionandpart` to provide compatibility to for instance `\bibliography`; restructured HA-prosper internals;

3.71 Solved a small bug writing final toc entry AtEndDocument; added `TCS` style by Chris Ellison.

3.8 Added `\tsection*`, `\tsectionandpart*` and nesting of bookmarks; created `\dualslide`; added `\hiddenitem`, `\putfigure`, `\figureitem` and `\putfigureitem`. Revised the documentation; revised the examples; added the `\dualslide` example; revised the styles.

3.9 Deleted `\putfigureitem` and `\figureitem` and replaced `\putfigure` by the more general `\topbl`; improved positioning of bottom text when using `\dualslide`; added `indent` option for `\dualslide`; added support for a split toc; added package option `hlsections`; added package option `blackslide`; solved a bug occurring when `\tsectionandpart` was used to create the last slide; added `enumstep`; made it possible to nest `itemstep` and `enumstep` environments; added `\HAPsetup`, yet only used for stepping environments.

4.0 Changed to `.dtx` format and wrote code documentation. Added slide options `bm` and `sound`. Added default behavior of stepping environments with the `ps2pdf` prosper option to be type 1, namely all active. Changed bookmarking process to a system that allows for deletion of bookmarks. Added global control of features through `HA-prosper.cfg`. Added `Aggie` style by Jack Stalnaker. Made some code more efficient. Added entrance point for ppr-prv to make redefining macros easier for that class. Solved a bug in combination with loading babel. Added `\onSlide` and `\OnSlide`.

4.1 Moved documentation to `.dtx` file. Added package option `counters`. Restored original `\item` and created `\xitem`. Changed `\hiddenitem` to `\xitemwait` which is a bit more general. Removed a bug in `\onSlide`. Rewrote a lot of code. Made HA-prosper use xkeyval. Changed `emptyslide` into more `partslide`. Added `Fyma` style by Laurent Jacques.

4.11 Corrected small bug in ppr-prv compatibility check. Made HA-prosper use original LaTeX macros `\title` and `\author`. Added missing '=' in `\HAP@contentsline` definition.

4.2 Improved table of contents typesetting. Added `\xitem<n>`. Added `thebibliography` environment. Improved positioning of the slide title in the styles. Added handle for state of list items. Updated styles. Added `TCSgrad` and `TCSTealBlue` styles by Chris Ellison. Added extra syntax for `\onSlide` and `\OnSlide` to make displaying material relative to the preceding `\xitem`.

4.21 Added `Capsules` and `Ciment` styles by Mathieu Goutelle. Updated for xkeyval version 1.4.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.