The MRTbundle

L^AT_EX Templates for the MRT, University of Bayreuth

by

Jonathan P. Spratte

3rd May 2020

Contents

List of Figures

List of Tables

1 Introduction

This bundle provides three LaTeX classes, one for theses and one for presentations, which both aim to match the corresponding MS Office templates of the Chair of Measurement and Control Engineering (Lehrstuhl für Mess- und Regeltechnik; MRT) of the University of Bayreuth, hence the name. Along the two major classes $\mathrm{MRTthesis}$ and $\mathrm{MRTbeam}$ there are another class to create stand alone images and minor auxiliary packages contained in this distribution.

The classes are originally created for use with pdfLaTeX and give the best results with it. This is caused by the available fonts. The classes were created for use with the helvet font which is not a good choice for LuaLaTeX and X∃LaTeX. Therefore, if you're using those engines, you'll be encouraged to choose a system font matching Helvetica or Arial.

This bundle makes no claim to be complete, comprehensive, or correct. For formatting errors I don't take any responsibility. Each author takes full liability for his work and its formatting.

You're allowed to share this work with fellow students working at the MRT, though official distribution channels might be better suited as they assure up to date versions.

I'd feel guilty distributing this bundle without saying the following: I'm not responsible for the overall look of this. I tried to match the Word template of the institution where possible and as a result, this is non-optimal typography, in my humble opinion.

Of course this documentation is created with one of the provided classes, namely $\mathrm{MRTthesis}$, in use.

If you're not yet familiar with LaTeX you should stop reading at this point (meaning the end of this paragraph) and either read a *good* and *up-to-date* introduction to LaTeX and afterwards read on or use MS Word for your thesis. Personally I think the time reading an introduction in order to use LaTeX is well spend, but there certainly are different opinions on that – unfortunately opinions are prone to be biased, mine is no exception. A viable introduction is lshort which is available in several languages at the following link: https://www.ctan.org/pkg/lshort

1.1 Feature Requests and Bug Reports

You can request features or report bugs at gitlass: https://gitlass.de/jonathan/MRTbundle. Alternatively if you have a github account you can use the issue tracker there: https://github.com/Skillmon/MRTbundle/issues, the repository is not mirrored though, so the code there is out of date.

You can request features or report bugs if you find some via email, too: mrt_depp@yahoo.de. Please use a descriptive subject containing "MRTbundle" (e.g. "MRTbundle – bug report").

1.2 Support and Questions

I aimed to make this documentation as complete as possible and as short as reasonable. This might have the effect that some things are explained too briefly and are not comprehensible. In cases like these you might have questions which I'd be happy to answer. Since my spare time might be limited you should first try to understand everything you have problems with using this documentation and perhaps taking a look at the provided example documents.

If you have questions which you can't solve with the material provided and the question is related to this bundle and not a general LaTeX question, contact me via email: mrt_depp@yahoo.de. Make sure the subject is descriptive. My answer might redirect you to other resources if I don't think the issue is related to this bundle. You can test whether your issue is related to this bundle by trying to recreate it using another class and without any MRT packages. If that is possible, it

is *not* related. If your query is stated somewhere in this documentation, my answer might not be very kind (*RTFM*).

Unrelated questions might be asked on `topanswers.xyz/tex` (English), `tex.stackexchange.com` (English) or `texwelt.de` (German).[1] Since the vast majority of people there don't have the packages and classes provided in this bundle installed, make sure that your example code doesn't rely on those, so that the *volunteers* can actually help you.

1.3 Individual Versions

Table 1-1: Versions of Individual Classes and Packages

Class/Package	Date	Version
MRTthesis	2020-03-27	0.0.24
MRTbeam	2020-03-27	0.0.12
MRTalone	2020-03-02	0.0.10
MRTtab	2020-02-08	0.0.14
MRTfonts	2020-05-03	0.0.7
MRTif	2019-12-29	0.0.15
MRTwuline	2019-02-03	0.0.3
MRTsfacc	2019-11-23	0.0.7
MRTlmscale	2019-02-05	0.0.1
MRTutil	2019-04-07	0.0.5

---

[1]Funny thing, I might answer you there, too, if I got time and your question isn't answered until then.

2 Downwards Incompatibilities

Though I try to avoid it as much as possible, there might sometimes be a backwards incompatibility, which is necessary to implement new features I deem worth it or to fix bugs. Where possible I'll try to provide means to switch back to something close to the old behaviour.

This chapter should list all those ground breaking changes (which might be really minor, though). Bugfixes which only correct behaviour to match the documentation will not be listed here. For example there was a long standing bug in `\MRTifGroupTF` that expanded leading tokens until it hit an unexpandable one which was fixed in v0.0.13, clearly a behaviour change, but only to get the originally intended behaviour.

2.1 MRTthesis

Table 2-1: Downwards Incompatibilities in MRTthesis introduced by Date and Version

Date	Version	Affected	Description
2019-11-20	v0.0.18	`\affidavit`	From this version on the affidavit is available using singular and plural forms, with the default being to automatically decide which one to use based on the given `author`. You can deactivate this using `affidavit plural=false` in `\MRTthesisSetup` (see subsection 3.1.2).
2019-11-17	v0.0.20	`\ifNoWidthTF`	From this version on the macro specifies a font in the box so that `\nullfont` doesn't break its purpose. You can use the starred form for its previous behaviour.

2.2 MRTtab

Table 2-2: Downwards Incompatibilities in MRTtab introduced by Date and Version

Date	Version	Affected	Description
2019-02-09	v0.0.5	`\MRTcline`	From this version on by default the entire line is coloured first and then the effects of `\MRTcline` are applied easing the process of drawing interrupted lines. Reversible with the `cline version` key (see subsection 6.6.1).

2.3 MRTfonts

Table 2-3: Downwards Incompatibilities in MRTfonts introduced by Date and Version

Date	Version	Affected	Description
2019-05-02	v0.0.3	`\altlUnscaled`	The entire macro got removed and will not come back.

Table 2-3: Downwards Incompatibilities in MRTfonts introduced by Date and Version (*continued*)

Date	Version	Affected	Description
2019-05-02	v0.0.3	\scalemath	The macro is only available if the scale macro option got used (see section 7.1).
2019-05-02	v0.0.3	new maths	The bm package is no longer automatically loaded if you use the new maths option. Additionally MRTlmscale and the maths fonts of lmodern will not be loaded at all.
2019-05-02	v0.0.3	maths letter l	By default the alt l option will be used replacing the letter l in maths by a letter distinct from an upper case I. You can revert to the previous behaviour using the std l option (see section 7.1).
2020-05-03	v0.0.7	inputenc	The package used to load inputenc with option utf8. Since modern LaTeX installations don't need this anyway but it reduces the versatility if other encodings are used, this got removed.

3 The MRTthesis **class**

MRTthesis provides the template to write a thesis at the MRT. It sports a layout which looks confusingly similar to the MS Word template provided by the chair. Of course there are some minor differences and the typesetting algorithm of TeX should create better line breaking than Word's but if one doesn't know what to pay attention on or for an untrained eye the distinction won't be possible (at least I hope so, as that was the goal in the first place).

3.1 Options

3.1.1 Load time options

The class features a few load time options.

longtable —NoArgument—

Is forwarded to MRTtab see its description in section 6.6.

hidelinks —NoArgument—

If used the hyperref option of the same name will be used. By default this is used. You can negate it with `showlinks`.

minimal —NoArgument—

If this option is passed some packages are not loaded and therefore related configurations not set. See section 3.3.

no geometry —NoArgument—

If this option is passed the geometry package is not loaded (and of course the page dimensions passed to geometry otherwise are not set).

showlinks —NoArgument—

If used the hyperref option `hidelinks` will not be used. This is the negation of `hidelinks` of this package.

tikzunderline —NoArgument—
tUline

This option is forwarded to MRTwuline.

british —NoArgument—
english
UKenglish If used the document will be using the `british` definition of babel. Many strings used in the package will be in English, but some might be missed out. If you find any of which you think should be translated, please contact me as described in section 1.1. English simplified (US) is not supported by the class.

languages = ⟨*options*⟩
langs With this you can define additional languages which should be loaded by the babel package, `ngerman` and `british` will always be loaded and one of them will be set as the main language. But if you for some reason need more

language definitions than that, you can load them with this key (you could also pass other ⟨*options*⟩ to the babel package with this key, they are just forwarded as you enter them).

The following options (and their values) will be forwarded to MRTfonts (see for their description):

- sfacc
- font
- serif font
- mono font
- new maths
- scale maths
- scale macro
- alt l
- std l
- mathsizes
- no mathsizes
- pmb

Every other given option will be passed on to scrreprt.

3.1.2 Setup options

The following options are accessible with \MRThesisSetup.

advisor = ⟨*name*⟩

Sets the name of the advisor of this thesis. One typical value could be Dipl.-Ing. Alice Fischerauer

author = ⟨*name*⟩

Sets the name of the author or authors as the macro \author does. Separate authors with \and. You can give the surname first followed by a comma and the given name, in which case the parsing for the abbreviation works better (especially with name affixes). The following two options are fine: author={Duck, Donald \and Mouse, Mickey} or author={Donald Duck \and Mickey Mouse}; both should result in the abbreviation D. Duck, M. Mouse. Another example would be zu Guttenberg, Karl-Theodor or Karl-Theodor zu Guttenberg. Here the parsing would result in K.-T. zu Guttenberg or K.-T. z. Guttenberg – the first one seems correct, the second one fails. Remember to surround the argument with braces if you use a comma.

caption above –NoArgument–

Is forwarded to MRTtab. See .

caption below –NoArgument–

Is forwarded to MRTtab. See .

citation width = ⟨*dimen*⟩

The width of the citation indications on the title page. Default is .5\textwidth.

degree = ⟨*degree*⟩

The degree you aim to achieve with the thesis. If you don't use this option it is tried to be guessed from the type of thesis you can specify with the thesis key. An error is thrown if the degree can't be guessed. If you don't want to achieve any degree, use the option no degree. Typical values would be Bachelor of Science or Master of Science.

examiner = ⟨*name*⟩

The examiner of the thesis. The initial value is set to `Univ.-Professor Dr.-Ing. Gerhard Fischerauer`.

logoL = ⟨*file*⟩

The image file for the left logo on the titlepage. `MRTresources_logo_UBT2-.pdf` is the initial value. If ⟨*file*⟩ is an empty argument no left logo will be used.

logoL height = ⟨*dimen*⟩

The height the left logo is displayed in. Initial value is `10.85mm`.

logoR = ⟨*file*⟩

The image file for the right logo on the titlepage. `MRTresources_logo_MRT2-.pdf` is the initial value. If ⟨*file*⟩ is an empty argument no right logo will be used.

logoR height = ⟨*dimen*⟩

The height the right logo is displayed in. Initial value is `11.9mm`.

no advisor = ⟨*bool*⟩

If true no advisor will be displayed on the title page. Default is true, initially is false.

no citation = ⟨*bool*⟩

If true no citation indications are displayed at the bottom of the title page. Default is true, initially is false.

no degree = ⟨*bool*⟩

If true no degree will be displayed on the title page. Default is true, initially is false. Also the paragraph corresponding to the degree in the affidavit will be left out.

no examiner = ⟨*bool*⟩

If true no examiner will be displayed on the title page. Default is true, initially is false.

no chair = ⟨*bool*⟩

If true no chair will be displayed on the title page. Default is true, initially is false.

no logos –NoArgument–

If used `logoL={}`,`logoR={}` is used, which results in no logos on the title

page.

---

no thesis = ⟨*bool*⟩

If true no thesis type will be displayed on the title page. Default is true, initially is false.

---

no usage = ⟨*bool*⟩

If true no usage rights are given to the MRT in the affidavit text. Default is true, initially is false. If you need a custom paragraph and don't want to leave it out completely you should redefine `\affidavittext@usagerights`.

---

number = ⟨*number*⟩

The MRT report number displayed in the citation indications. Initially is empty. The typical pattern of these numbers is something like: `TT-yy-mm-nn` with `TT` the thesis type, e. g. `BA` or `MA`, `yy` the last two digits of the year, `mm` the month, and `nn` the number of the thesis in this month.

---

pos figure = ⟨*placement*⟩

The ⟨*placement*⟩ of floats of type `figure`.

---

pos float = ⟨*placement*⟩

The ⟨*placement*⟩ of floats of both types, `figure` and `table`. Initially set to `tbp`.

---

pos MRTtable = ⟨*placement*⟩

The ⟨*placement*⟩ of floating `MRTtable`s, forwarded to MRTtab's option `pos`. See subsection 6.6.1.

---

pos table = ⟨*placement*⟩

The ⟨*placement*⟩ of floats of type `table`.

---

short advisor
sadvisor = ⟨*abbreviation*⟩

The abbreviated name of the advisor. This is needed for the citation indications and not parsed automatically from the `advisor`, as the name contains academic titles, but the abbreviation should not and the parsing would be hard to do correctly.

---

short author
sauthor = ⟨*abbreviation*⟩

The abbreviated name or names of the author or authors. If you don't use this option it is tried to parse those automatically. If the parsing does something wrong you'll have to use this option giving the correct abbreviations with each name separated with commas from the others, e. g. `short author={D. Duck, M. Mouse}`.

short examiner **sexaminer**	$= \langle abbreviation \rangle$ The abbreviated name of the examiner. This is needed for the citation indications and not parsed automatically from the `examiner`, as the name contains academic titles, but the abbreviation should not and parsing would be hard to do correctly. Initial value is `G. Fischerauer`.
sign height	$= \langle dimen \rangle$ The height reserved for each signature below the affidavit text. Initial value is `9mm`.
sign separation **sign sep**	$= \langle dimen \rangle$ If `sign width max` is not given (or `0pt`) the maximum width is calculated from the text width and the width of the date and location. The minimum distance from the date and location to the signature lines is then enforced to be at least $\langle dimen \rangle$. Initial value is `2em`.
sign width max	$= \langle dimen \rangle$ You can enforce a maximum width for the signature lines below the affidavit using this option. If it is not used, the maximum width is calculated.
sign width min	$= \langle dimen \rangle$ You can enforce a minimum width for the signature lines using this option. Initially this is set to `7cm`.
stretch caption **stretch cap**	$= \langle float \rangle$ Uses `\setkomafont` to enforce a specific line spread using `\setstretch` for captions.
stretch tabular **stretch tab**	$= \langle float \rangle$ Is forwarded to MRTtab's option `stretch tab`. See subsection 6.6.1.
stretch text	$= \langle float \rangle$ Uses `\setstretch` to set a specific line spread in the document.
stretches	$= \langle float \rangle$ Sets `stretch cap`, `stretch tab`, and `stretch text` in one go. Initially set to `1.408`.
subtitle	$= \langle title \rangle$ The title page might include a subtitle. If you really want to use it, you'd have to use `with subtitle`. You can also use `\subtitle` to set it.
thesis	$= \langle thesis\ type \rangle$

Sets the ⟨*thesis type*⟩. Typical arguments would be `Bachelorarbeit` or `Bachelor Thesis` (the former if you're writing in German, the latter if you're writing in English).

`title` = ⟨*title*⟩

Sets the title of the thesis. You might also use `\title` to set this.

`toc ChapIndent` = ⟨*dimen*⟩

Sets the indentation of chapter entries in the table of contents. Initially set to `0.01em`.

`toc SecIndent` = ⟨*dimen*⟩

Sets the indentation of section entries in the table of contents. Initially set to `1.32em`. The width is also used for entries in the list of figures and list of tables.

`toc sSecIndent` = ⟨*dimen*⟩

Sets the indentation of subsection entries in the table of contents. Initially set to `3.38em`.

`toc ssSecIndent` = ⟨*dimen*⟩

Sets the indentation of subsubsection entries in the table of contents. Initially set to `6.38em`.

`with subtitle` = ⟨*bool*⟩

If true a subtitle can be used on the title page. Default is true, initially is false.

3.1.2.1 Options concerning automatically added contents

The following additional options can be set with `\MRTthesisSetup`. They all resolve around automatically added contents.

`backmatter` = ⟨*choice*⟩

A ⟨*choice*⟩ whether you want the back matter to be added automatically. Possible values are `auto` and `manual`. If set to `auto` the appendix will automatically be included at `\end{document}`. It might contain the following (dependent on the values of other keys; in correct order):

- bibliography (option `bib`)
- list of figures (option `lof`)
- list of tables (option `lot`)
- contents added with `\MRTthesisAddToBack`
- the contents of your appendix file (option `appendix`)
- contents added with `\MRTthesisAddAfterBack`
- the affidavit (option `affidavit`)

It also includes the necessary formatting switches otherwise contained in

\appendix. Default is `manual`.

---
frontmatter

= ⟨*choice*⟩

A ⟨*choice*⟩ whether you want the front matter to be added automatically. Possible values are `auto` and `manual`. If set to `auto` the front matter will automatically be included at `\begin{document}`. It might contain the following (dependent on the values of other keys; in correct order):

- title page
- the affidavit (option `affidavit`)
- the acknowledgements (option `acknowledgement`)
- table of contents (option `toc`)
- list of figures (option `lof`)
- list of tables (option `lot`)
- contents added with `\MRTthesisAddToFront`

It also includes the necessary formatting switches otherwise contained in `\mainpart`. Default is `manual`.

---
acknowledgement

= ⟨*file*⟩

Sets the acknowledgements file added if `frontmatter=auto` is used. If ⟨*file*⟩ (the argument) is empty no file will be added. By default it is empty. The ⟨*file*⟩ will be added using `\include`, so you should drop the `.tex` ending.

---
affidavit

= ⟨*choice*⟩

Sets where the `\affidavit` is added. Possible ⟨*choice*⟩s are `front`, `back` and `off`. If `off` is used it doesn't get added automatically. Default value is `front`. `front` and `back` will only take effect if `frontmatter` and `backmatter` are set to `auto`, respectively.

---
affidavit plural

= ⟨*choice*⟩

The `\affidavit`'s default text can be typeset using first person plural or singular forms. With this choice you can set whether you want this or not. Viable choices are `auto`, the class will try to parse this from the given `author`, `true`, meaning the affidavit will be typeset using plural forms, or `false`, meaning the affidavit will be typeset using singular forms.

---
appendix

= ⟨*file*⟩

Sets the appendix file added if `backmatter=auto` is used. If ⟨*file*⟩ (the argument) is empty no file will be added. By default it is empty. The ⟨*file*⟩ will be added using `\include`, so you should drop the `.tex` ending.

---
appendix ragged

= ⟨*bool*⟩

If set `true` the contents of the `appendix` file will be typeset `\raggedbottom`. Default is `true`.

---
bib
bibliography

= ⟨*bool*⟩

Sets whether the bibliography should be added automatically if `backmatter=auto` is used. It gets set to `false` if the class option `minimal` is used.

---

lof = ⟨*choice*⟩

Sets where the list of figures is added. Possible ⟨*choice*⟩s are `front`, `back` and `off`. If `off` is used it doesn't get added automatically. Default value is `front`. `front` and `back` will only take effect if `frontmatter` and `backmatter` are set to `auto`, respectively.

---

lot = ⟨*choice*⟩

Sets where the list of tables is added. Possible ⟨*choice*⟩s are `front`, `back` and `off`. If `off` is used it doesn't get added automatically. Default value is `front`. `front` and `back` will only take effect if `frontmatter` and `backmatter` are set to `auto`, respectively.

---

toc = ⟨*choice*⟩

Sets where the list of tables is added if `frontmatter=auto` is used. Possible ⟨*choice*⟩s are `front` and `off`. If `off` is used it doesn't get added automatically. Default value is `front`.

3.2 Macros

The following macros are provided:

\ifNoWidth*TF* Usage: \ifNoWidth*TF*⟨*∗*⟩{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Typesets the argument in a box (so the code is actually executed). If the produced box has a width of 0pt the ⟨*true*⟩ branch is executed, else the ⟨*false*⟩ branch. If the ⟨*∗*⟩ is given the surrounding font settings are used, if it is not given Computer Modern Roman at 10pt (`cmr10`) is used. This is done so that the macro gives its intended result in environments where the font has no width (TeX has a special \nullfont).

\vfillmult Usage: \vfillmult{⟨*num*⟩}

Same as if you'd use ⟨*num*⟩ instances of \vfill.

\hfillmult Usage: \hfillmult{⟨*num*⟩}

Same as if you'd use ⟨*num*⟩ instances of \hfill.

\MRTafterhyperref Usage: \MRTafterhyperref{⟨*content*⟩}

Places ⟨*content*⟩ after hyperref is loaded. This is important for the relatively few packages that need to be loaded after hyperref. So if you have one of these, you should use something like \MRTafterhyperref{\usepackage {cleveref}}. This macro has to be used prior to \begin{document}.

\MRTthesisAddToFront

Usage: `\MRTthesisAddToFront{⟨content⟩}`

Adds ⟨content⟩ to a hook executed during the front matter if `frontmatter=auto` was used. See subsubsection 3.1.2.1 for more information.

\MRTthesisAddToBack

Usage: `\MRTthesisAddToBack{⟨content⟩}`

Adds ⟨content⟩ to a hook executed during the back matter if `backmatter=auto` was used. See subsubsection 3.1.2.1 for more information.

\MRTthesisAddAfterBack

Usage: `\MRTthesisAddAfterBack{⟨content⟩}`

Adds ⟨content⟩ to a hook executed during the back matter if `backmatter=auto` was used. See subsubsection 3.1.2.1 for more information.

\MRTthesisAddToAppendixSwitch

Usage: `\MRTthesisAddToAppendixSwitch{⟨content⟩}`

Adds ⟨content⟩ to the `\appendix` switch. This is useful if you need the behaviour of some other packages changed during the appendix.

\DeclareTOCStyleEntryMRTChapterLike

Usage: `\DeclareTOCStyleEntryMRTChapterLike[⟨indent⟩]`
`{⟨entry-layer⟩}[⟨options⟩]`

See the description of `\DeclareTOCStyleEntryMRTSectionLike`. The difference is that this sets the entries how the chapters are formatted. Also the ⟨indent⟩ defaults to the one of chapters.

\DeclareTOCStyleEntryMRTSectionLike

Usage: `\DeclareTOCStyleEntryMRTSectionLike[⟨indent⟩]`
`{⟨entry-layer⟩}[⟨options⟩]`

The macro calls the KOMA macro `\DeclareTOCStyleEntry` and sets the options how they are used for the section entries in the table of contents. ⟨indent⟩ defaults to the indent length of section entries. It is possible to use an ⟨indent⟩ but give more options in that optional argument afterwards (comma separated). Table 3-1 shows an overview of the class's default indentations. With the ⟨options⟩ argument you can specify further options past the indent to `\DeclareTOCStyleEntry`.

\MRTthesisSetup

Usage: `\MRTthesisSetup{⟨options⟩}`

You can use this macro to set the options listed in subsection 3.1.2.

\sethead

Usage: `\sethead{⟨content⟩}`

Sets the head marks for both sides to ⟨content⟩. It is the same as `\markboth{⟨content⟩}{⟨content⟩}`. You might use this (or any similar macro provided by KOMA script) to manually set the head marks, e. g. if your section title gets too long.

Table 3-1: Indents of different ToC entry types and the macros they are stored in. Use the options described in subsection 3.1.2 to change the values.

entry-layer	macro name	default length
chapter	\l_MRTthesis_toc_chapter_indent_tl	0.01em
section	\l_MRTthesis_toc_section_indent_tl	1.32em
subsection	\l_MRTthesis_toc_subsection_indent_tl	3.38em
subsubsection	\l_MRTthesis_toc_subsubsection_indent_tl	6.38em
table	\l_MRTthesis_toc_section_indent_tl	1.32em
figure	\l_MRTthesis_toc_section_indent_tl	1.32em

\affidavit Usage: \affidavit

Prints a chapter "Eidesstattliche Erklärung" (stored in \affidavittitle, you might redefine it to change the title) and the affidavit text (as stored in \affidavittext) and the location and date, followed by a signature line for each author. \maketitle has to be used prior to it, else the lines won't be printed. This is a bug I might fix in the future.

\mainpart Usage: \mainpart
\mainmatter
Switches the formatting from the one at the beginning to the one used in the main part of the document. Should be used after \tableofcontents, \listoffigures, and \listoftables.

\appendix Usage: \appendix

Switches the formatting to the one used in the appendix. This includes switching to alphabetically numbered sections and setting the option no float in \MRTtabSetup.

Additionally the macros \author, \title, and \subtitle have been redefined to internally use \MRTthesisSetup to set the corresponding options. \author takes an optional argument with which you can set the abbreviated list of authors.

3.3 Dependencies

As this class is based on scrreprt, it depends on that class and all of its dependencies, of course. Additionally the following packages are loaded (used options given in brackets). Those are quite some but unfortunately most of these are required (or help a lot) to achieve certain formattings in order to match the MS Word template of the MRT best.

Some of the used packages are not necessarily needed to match the MS Word template, but provide useful features – e.g. hyperref which allows the use of \autoref and cross linking but is not needed to match any specific formatting.[1]

---

[1]Don't remove it though, the current code for section headings relies on it.

- expl3
- xparse
- MRTif
- MRTutil
- MRTsfacc
- MRTtab
- MRTwuline
- MRTfonts
- babel [ngerman] or if `british` is used with [main = british, ngerman]
- scrlayer-scrpage [singlespacing = true]
- geometry (with correct options)
- setspace
- xcolor
- graphicx
- enumitem

- mathtools
- mathastext [italic,defaultmathsizes]
- isomath
- hyperref
- if the `minimal` option is not used:
  - siunitx [detect-all, per-mode=reciprocal-positive-first]
    If babel's `british` is used [locale = UK] will be used, if `ngerman` [locale=DE]. Additionally the `range-phrase` will be set to either `to` or `bis` with spaces around it.
  - biblatex [backend = biber, natbib = true, citestyle = numeric, bibstyle = numeric, sorting=none, giveninits=true, sortcites] (with URLs being line breakable at any place)
  - csquotes

4 The MRTbeam **class**

The MRTbeam class is a class build upon beamer. It should mimic the style of the MS Powerpoint template of the MRT which was in use when I held my Bachelor's presentation. I heard the requirements to match a specific template are less strict today, but at least I'll still use this template.

Many of the features described here are also available if one uses \usetheme{MRTbeam} within a document using the beamer class. There however is no dedicated documentation for that possibility provided. You're encouraged to also use the corresponding MRTbeam class if you're using the eponymous theme.

If there is a new institution template which should be matched that doesn't match this beamer template please contact me as described in section 1.1.

4.1 Random chatter

The creation of a presentation using beamer is not everyone's cup of tea. Refer to the beamer manual to get a basic idea of how to use it, as MRTbeam only adds some stuff that is not basic beamer stuff. The main idea of creating a presentation remains that of beamer.

MRTbeam doesn't follow the way beamer does things everywhere. As a result some stuff may not work out as you expect if you're used to beamer. Especially the customization might require you to actually read the sources of MRTbeam and its beamer themes. Also MRTbeam is only for presentation mode as of now.

Special thanks are due to the TeX.SX user samcarter, who helped me ram my head through beamer's walls in order to get my will.

4.2 Frame contents

The class builds up frames as shown in Figure 4-1 (not true to scale).

You can specify the used logos using \uselogo. The default is the UBT logo on the left, no logo in the centre, and the MRT logo on the right side.

Figure 4-1: The basic layout of a frame in MRTbeam

If you specify no or an empty frame title the current section is used (with its numbering). If you don't want any frame title pass in \relax as the title. The frame's subtitle can be prepended with the current subsection (with its numbering) followed by a colon. This depends on the current value of \ifPrependSubsections.

The left footer contains the occasion, the shorttitle, and the shortauthor. If no short title or no short author is given, the title and the author, respectively, are used instead. If you give a * for the short author or the short title, they are left out (e.g. with using \title[*]{foo}).

The centre footer contains the frame number and if you want a progress bar. The progress bar is shown if \ifProgressBar is true.

In the right footer the following is displayed atop of each other: Persistent MRT footnotes, volatile MRT footnotes, citation MRT footnotes, normal footnotes. The right footer has enough space for three entries. If you need more they are scaled to the available vertical space. MRT footnotes might be displayed in a tabular manner with the labels in the left and the actual notes in the right column. This depends on the value of \ifTabularNotes.

Neither of the footers is restricted in horizontal size. As a result they might overlap if you specify really long contents.

4.3 Options

The class passes almost all options given to it on to beamer. The few handled differently are described here.

aspect $= \langle num_1 \rangle : \langle num_2 \rangle$

If this is used the aspect ratio of the slide gets set to the specified aspect. Typical usage would be aspect=16:9. $\langle num_1 \rangle$ and $\langle num_2 \rangle$ can be arbitrary unsigned integers.

The following options (and their values) will be forwarded to MRTfonts (see section 7.1 for their description):

- sfacc
- font
- serif font
- mono font
- new maths
- scale maths
- scale macro
- alt l
- std l
- mathsizes
- no mathsizes
- pmb

There are still some more class specific options which you can set with some macros instead of a key=value interface. The macros in this section are only provided to set specific options, other macros provided by MRTbeam are described in section 4.4.

\advisor   Usage: \advisor$\langle * \rangle$[$\langle title \rangle$]{$\langle name \rangle$}

Sets $\langle name \rangle$ as the current advisor. It also redefines itself, any consecutive call will not take any arguments but return the $\langle name \rangle$. The $\langle title \rangle$ shall be the title used on the title frame defaulting to 'Betreuerin' if the starred version is used, else it defaults to 'Betreuer'.

\occasion   Usage: \occasion{$\langle occasion \rangle$}

Defines the occasion of the presentation. If used the occasion will be displayed in the left footer.

\uselogo

Usage: \uselogo{⟨*pos*⟩}[⟨*options*⟩]{⟨*file*⟩}

Specifies the logo used at the position ⟨*pos*⟩. There are l, c, and r available. The ⟨*file*⟩ is included using \includegraphics with the specified ⟨*options*⟩ (defaulting to height=0.080864\paperheight). If ⟨*file*⟩ is an empty argument there is no logo used at the specified position. MRTresources_logo_-UBT2.pdf is used for the left logo and MRTresources_logo_MRT2.pdf for the right one by default. The centre logo is initially empty.

\setlogobox

Usage: \setlogobox{⟨*pos*⟩}{⟨*contents*⟩}

The logos are actually only typeset once inside a horizontal box and afterwards that box is used. While \uselogo only allows to place graphics files, you can use arbitrary ⟨*contents*⟩ (in horizontal mode) with this macro. The ⟨*pos*⟩ argument is the same as for \uselogo.

\ShowGrid

Usage: \ShowGrid[⟨*options*⟩]

Globally activates a TikZ grid displayed in the background of the frames. You can specify the TikZ-style used for the grid with ⟨*options*⟩. The default is: xstep=.05\paperwidth, ystep=.1\paperheight, help lines.

\HideGrid

Usage: \HideGrid⟨*∗*⟩

Globally deactivates the background grid and restores the package's default options for that grid. If the starred version is used, the options are not reset.

\ifPrependSubsections
\PrependSubsectionstrue
\PrependSubsectionsfalse

Usage: \ifPrependSubsections

If set true each frame's subtitle is prepended by the current subsection.

\ifOnlyOneTopRule
\OnlyOneTopRuletrue
\OnlyOneTopRulefalse

Usage: \ifOnlyOneTopRule

If set true in each frame the title and subtitle will not be displayed and the lower top rule will be omitted, significantly enlarging the content area. If you use \OnlyOneTopRuletrue or \OnlyOneTopRulefalse \contentheight will be adjusted.

\ifProgressBar
\ProgressBartrue
\ProgressBarfalse

Usage: \ifProgressBar

If set true a progress bar will be shown in the middle of the slides foot at the frame number. You can customize the progress bar shown using \SetProgressBar or \ProgressBarStyle.

\SetProgressBar

Usage: \SetProgressBar⟨*∗*⟩{⟨*align*⟩}{⟨*length*⟩}{⟨*height*⟩}{⟨*voffset*⟩}

This changes the default values of \ProgressBar. If the starred version is used the changes are made locally, else they are applied globally. Take a look at the description of \ProgressBar for an explanation what each of the parameters mean. If you use a ∗ as one of the arguments the corresponding

Table 4-1: Available Progress Bar Styles for \ProgressBarStyle

style	align	length	height	voffset	description
default	c	30pt	font size	-1.65ex	a thick and relatively short bar around the frame number
Spratte	c	\paperwidth	2pt	3pt	A thin line spanning the whole page width at the bottom of the frame

default value will remain unchanged.

\ProgressBarStyle

Usage: \ProgressBarStyle⟨*⟩{⟨*style*⟩}

This sets the progress bar options to a predefined ⟨*style*⟩ using the unstarred version of \SetProgressBar. If the starred version of \ProgressBarStyle is used \ProgressBartrue is issued. Available styles are listed in Table 4-1.

\ProgressBarColors

Usage: \ProgressBarColors
[⟨*model-list₁*⟩]{⟨*color₁*⟩}[⟨*model-list₂*⟩]{⟨*color₂*⟩}

Sets the colours of the progress bar in one macro. If you use either of the optional arguments the respective colour is defined using \definecolor and the ⟨*color*⟩ arguments are the ⟨*spec-list*⟩ arguments of xcolor, if the optional arguments aren't used the definition is done using \colorlet and the ⟨*color*⟩ arguments match the colour expressions of xcolor. ⟨*model-list₁*⟩ and ⟨*color₁*⟩ set the progressed colour and ⟨*model-list₂*⟩ and ⟨*color₂*⟩ set the noprogress colour. For example to set the progressed colour to 85 % grey scale and the noprogress colour to a 50 : 50 mixture of red and green, you can use \ProgressBarColors[gray]{0.85}{red!50!green}.

\StretchyLists

Usage: \StretchyLists⟨!⟩⟨*⟩[⟨*factor*⟩]

Redefines the internals \@listi, \@listii, and \@listiii, such that lists (itemize, enumerate, description) are stretchy. Inside a beamer frame that has vertically centred contents the used order of infinity must be fill, but inside of constructs such as minipages you need fil. By default fill is used by \StretchyLists, but if you use the starred version fil will be used. In addition you can specify the stretch ⟨*factor*⟩, which defaults to 0.3. If the ! is used, instead of appending fill or fil to ⟨*factor*⟩ nothing will be appended, such that one might specify another unit, e.g., with \StretchyLists![5pt].

\ResetLists

Usage: \ResetLists

Reset the list internals set by \StretchyLists to their default values.

\FixedColumnHeight

Usage: \FixedColumnHeight⟨!{⟨*skip*⟩}⟩⟨*⟩[⟨*height*⟩]

If you want to use \StretchyLists inside of a beamer column, you'll note that they don't stretch. That's because those columns have no fixed height. With this macro you can locally redefine column to use a fixed ⟨*height*⟩, defaulting to \contentheight. Depending on the column's vertical alignment \vfills will be added on top and below. If you use the starred form \vfil will be used

instead of `\vfill`. If you provide the ⟨*!*⟩ you have to give another argument specifying the ⟨*skip*⟩ which should be used.

4.3.1 Footnote related

`\ifTabularNotes`
`\TabularNotestrue`
`\TabularNotesfalse`
	Usage: `\ifTabularNotes`
	
	If set true the MRT footnotes will be displayed in a tabular manner with two columns. MRT footnotes are those footnotes set with the footnote related macros in subsection 4.4.1.

`\ColumnsTabularNotes`	Usage: `\ColumnsTabularNotes{`⟨*specification*⟩`}` With this macro you can specify the column specifications used by MRT footnotes. Your definition should contain two columns.

4.3.2 Bibliography related

`\ifExplicitCiteOnce`
`\ExplicitCiteOncetrue`
`\ExplicitCiteOncefalse`
	Usage: `\ifExplicitCiteOnce`
	
	If set to true for every used key the citation is in an explicit manner only once. For each following citation of the same key only the number is used.

`\ifNoExplicitCite`
`\NoExplicitCitetrue`
`\NoExplicitCitefalse`
	Usage: `\ifNoExplicitCite`
	
	If set to true there will never be an explicit citation at the frame, only the citation number will be used.

`\ifNoJournalCite`
`\NoJournalCitetrue`
`\NoJournalCitefalse`
	Usage: `\ifNoJournalCite`
	
	If set to true there will be no journal title in the citations produced by `\insertcite`█ and `\insertframecite`.

4.4 Macros

`\PlaceAt`	Usage: `\PlaceAt`⟨*∗*⟩`(`⟨*pos*⟩`)[`⟨*node options*⟩`]{`⟨*content*⟩`}` The starred version differs fundamentally from the unstarred one. The unstarred one places ⟨*content*⟩ at the specified position ⟨*pos*⟩ in the background inside a TikZ node with the optionally specified ⟨*node options*⟩. The coordinates default to multiples of `\pagewidth` and `\pageheight` for x and y, respectively. You can use anything TikZ understands as coordinates for ⟨*pos*⟩. The starred version places the `tikzpicture` where you currently are. It uses `remember picture` and `overlay` as options. The ⟨*pos*⟩ must match the pattern (⟨*x*⟩,⟨*y*⟩). ⟨*x*⟩ is in multiples of `\pagewidth` and ⟨*y*⟩ in multiples of `\pageheight` and you can't change that. The node still gets ⟨*node options*⟩. In both cases (0,0) is the bottom left corner of the slide.

`\AddToPlaced`	Usage: `\AddToPlaced{`⟨*TikZ code*⟩`}`

Adds the specified ⟨*TikZ code*⟩ to the background of the current slide. (0,0) is the bottom left corner of the slide. Coordinates are by default in multiples of \pagewidth and \pageheight. It uses the same tikzpicture as \PlaceAt and is stored in the same macro.

\ProgressBar

Usage: \ProgressBar[⟨*align*⟩]<⟨*length*⟩>[⟨*height*⟩][⟨*voffset*⟩]

Prints a progress bar. ⟨*align*⟩ is the horizontal alignment as you would pass it to a \makebox, the initial default is c. ⟨*length*⟩ is the overall length the progress bar should have (defaulting to 30pt), ⟨*height*⟩ its height, defaulting to the current font size. ⟨*voffset*⟩ allows you to offset the progress bar vertically. With positive values the shift is downwards, the default is -1.65ex. The progress bar uses the xcolor colours progressed and noprogress. All arguments are optional.

\StartOfProgress
\EndOfProgress

Usage: \StartOfProgress

Denotes the start and end of the progress bars gauge. The frame after the call of \StartOfProgress is the first frame filling the gauge, the frame prior to \EndOfProgress is the first frame in which the gauge is fully filled. The macros should be used outside of the frame environment. If \StartOfProgress is not used the first frame starts filling the gauge, if \EndOfProgress is not used the last frame is the only one with a completely filled gauge.

\contentwidth
\contentheight

Usage: \contentwidth

These are lengths which are set to match the height and width of the content block of a frame (the space between the bottom rule and the lower top rule). \textwidth should match \contentwidth if you're outside of a minipage or similar, but \textheight will most likely not match the actual height of the content area.

\UseAndIfEmptyTF

Usage: \UseAndIfEmptyTF[⟨*pre*⟩]{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

The ⟨*arg*⟩ is expanded inside a box. If that box has a width not equal 0pt ⟨*pre*⟩ is used followed by the contents of the box. Then the ⟨*false*⟩ branch is executed. If the box's width equals 0pt the ⟨*true*⟩ branch is used instead and neither ⟨*pre*⟩ is used nor the box containing ⟨*arg*⟩ placed.

\cursec

Usage: \cursec⟨***⟩

If the current section is starred or you used the optional * for \cursec, this macro inserts the current sections name, else the name is prepended by the current sections number.

\curssec

Usage: \curssec⟨***⟩

This macro is very similar to \cursec. If you used the starred version of it or the current subsection is starred, this macro inserts the current subsections name, else the name is prepended by the current subsections number.

\maketitle	Usage: \maketitle[⟨*options*⟩]{⟨*title*⟩}{⟨*subtitle*⟩}

The \maketitle macro is redefined compared to the definition made by beamer. It accepts arguments, all of which are optional, even the ones enclosed by curly braces. If you use \maketitle inside of a frame environment, the arguments are gobbled and the same behaviour as in beamer is used. If you're using it outside of frame, it'll issue a frame environment with the ⟨*options*⟩, ⟨*title*⟩ and ⟨*subtitle*⟩ arguments forwarded to it. The title page will be set with \OnlyOneTopRuletrue if you didn't use the ⟨*title*⟩ argument.

whiteframes	Usage: \begin{whiteframes} ... \end{whiteframes}

In this environment \ifwhiteframes is set true.

4.4.1 Footnote related

The term footnotes relates to the special MRT footnotes in this subsection.

\AddToRightFoot	Usage: \AddToRightFoot⟨*⟩⟨+⟩<⟨*overlay*⟩>[⟨*pre*⟩]{⟨*note*⟩}

This macro adds stuff to the right footer. If ⟨*⟩ is given, the content is added to the persistent footnotes, else if ⟨+⟩ is given added to the cite related footnotes, else to the ordinary ones. ⟨*overlay*⟩ is used for any overlay specifications using \uncover. ⟨*pre*⟩ is added left to ⟨*note*⟩. If tabular footnotes are used ⟨*pre*⟩ is in the left, ⟨*note*⟩ in the right column. If tabular footnotes are not used the distance between ⟨*pre*⟩ and ⟨*note*⟩ is 0.5\tabcolsep. The starred variant should only be used outside of the frame environment. If you get strange errors during compilation a \noexpand in front of some macros (e.g. stuff like \href) you give as arguments might help.

\ClearRightFoot	Usage: \ClearRightFoot⟨*⟩

Clears the footnotes. If the * is given only the volatile footnotes are cleared, else all of them.

\SetLeftFoot	Usage: \SetLeftFoot{⟨*contents*⟩}

Sets the ⟨*contents*⟩ of the left footer locally.

\ClearLeftFoot	Usage: \ClearLeftFoot

Clears the left footer locally. This might be handy to clear the footers on the title page, you could use it with \begingroup\ClearLeftFoot\maketitle \endgroup.

4.4.2 Bibliography related

\cite	Usage: \cite<⟨*overlay*⟩>⟨*⟩[⟨*opt₁*⟩][⟨*opt₂*⟩]{⟨*key*⟩}

⟨*overlay*⟩ is handled by \uncover, which affects only the footnote not the footnote mark. The usage of the two optional arguments and ⟨*key*⟩ match

those known from biblatex's \cite. The citation's contents are dependent on \ifNoExplicitCite and \ifExplicitCiteOnce, an explicit citation contains the citation number, authors' names, the journal, and the year. If the star is given, no explicit citation will be used.

\framecite   Usage: \framecite⟨*⟩<⟨overlay⟩>[⟨pre⟩]{⟨key⟩}[⟨post⟩]

Places a citation in the footnotes, if the starred version is used using the persistent footnotes, else the volatile non-cite related footnotes. Use the starred version prior to the frame it should first be shown in. ⟨overlay⟩ specifications are interpreted by \uncover. ⟨pre⟩ is put in front of the citation with a distance of \, (unaffected by tabular footnotes options), ⟨post⟩ with a distance of \, after the citation. The citation contains the authors' names, the journal, and the year.

\bibliographyframe   Usage: \bibliographyframe⟨*⟩⟨!⟩[⟨bibfont⟩]{⟨title⟩}{⟨subtitle⟩}

Prints the bibliography. The starred variant uses \whiteframestrue. The ⟨bibfont⟩ defaults to \small, you might give any font related commands here. Both ⟨title⟩ and ⟨subtitle⟩ are optional though delimited by curly braces. ⟨title⟩ defaults to 'Quellen', ⟨subtitle⟩ is initially empty. The \bibliographyframe is printed using allowframebreaks by default, if you give the optional ! it will be printed without allowframebreaks.

\inlinecite   Usage: \inlinecite[⟨opt₁⟩][⟨opt₂⟩]{⟨key⟩}

Gives the citation which would be placed in the text by \cite without any \textcolor. In fact \cite uses this internally.

\insertcite   Usage: \insertcite{⟨key⟩}

Gives the citation which would be placed in the footnote by \cite. In fact \cite uses this internally.

\insertframecite   Usage: \insertframecite{⟨key⟩}

Gives the citation which would be placed in the footnote by \framecite. In fact \framecite uses this internally.

4.5 Dependencies

The class uses beamer as its basis. Additionally the following packages are loaded:
- MRTif
- MRTsfacc
- MRTutil
- xparse
- MRTfonts
- TikZ
- biblatex (with URLs being line breakable at any place)

biblatex uses biber as its backend.

5 The MRTalone **class**

The standalone version of MRTthesis. The aim is to provide a class to produce simple LATEX based images which match the look of MRTthesis.

5.1 Options and Setup Files

MRTalone allows you to use a shared options file and a shared setup file for your project. The files get sourced if they are available.

The shared options might be placed into the file ./MRTalone.option.tex. This file can include any number of \MRTaloneOptions calls. It is loaded in the midst of the class file. See section 5.3 for a description of \MRTaloneOptions and subsection 5.2.1 for the available options.

The additional setup file should be ./MRTalone.setup.tex. It is sourced at the end of the class and might contain any valid LATEX code, of course including some \MRTaloneSetup instructions. See section 5.3 for \MRTaloneSetup and subsection 5.2.2 for the available setup options.

5.2 Options

5.2.1 Load time options

The class features a few load time options.

longtable –NoArgument–
Is forwarded to MRTtab see its description in section 6.6.

minimal –NoArgument–
If this option is passed some packages are not loaded and therefore related configurations not set. See section 5.4.

tikzunderline –NoArgument–
tUline This option is forwarded to MRTwuline. See its description in section 9.1.

british –NoArgument–
english If used the document will be using the british definition of babel. Many
UKenglish strings used in the package will be in English, but some might be missed out. If you find any of which you think should be translated, please contact me as described in section 1.1. English simplified (US) is not supported by the class.

languages = ⟨options⟩
langs With this you can define additional languages which should be loaded by the babel package, ngerman and british will always be loaded and one of them will be set as the main language. But if you for some reason need more language definitions than that, you can load them with this key (you could also pass other ⟨options⟩ to the babel package with this key, they are just forwarded as you enter them).

The following options (and their values) will be forwarded to MRTfonts (see section 7.1 for

their description):

- sfacc
- font
- serif font
- mono font
- new maths
- scale maths
- scale macro
- alt l
- std l
- mathsizes
- no mathsizes
- pmb

Every other given option will be passed on to standalone.

5.2.2 Setup options

The following options are accessible with \MRTaloneSetup.

caption above   –NoArgument–

Is forwarded to MRTtab and its \MRTtabSetup. See its description in subsection 6.6.1.

caption below   –NoArgument–

Is forwarded to MRTtab and its \MRTtabSetup. See its description in subsection 6.6.1.

stretch caption
stretch cap $= \langle float \rangle$

Currently does nothing.

stretch tabular
stretch tab $= \langle float \rangle$

Is forwarded to MRTtab and its \MRTtabSetup. See its description in subsection 6.6.1.

stretch text $= \langle float \rangle$

Uses \setstretch to set a specific line spread in the document.

stretches $= \langle float \rangle$

Sets stretch cap, stretch tab, and stretch text in one go. Initially set to 1.408.

5.3 Macros

\MRTaloneSetup   Usage: \MRTaloneSetup{⟨options⟩}

You can use this macro to set the options listed in subsection 5.2.2.

\MRTaloneOptions   Usage: \MRTaloneOptions{⟨options⟩}

You can use this macro to set the options listed in subsection 5.2.1. It is only available inside of the ./MRTalone.option.tex file (see section 5.1).

5.4 Dependencies

The class is based on standalone, therefore it naturally depends on that and all its dependencies. Additional dependencies are:

- expl3
- xparse
- MRTtab for which `in text sep` is set to 0pt and the option `no float` is set. Take a look at subsection 6.6.1 to see what those do.
- MRTwuline
- MRTsfacc
- MRTfonts
- babel [ngerman] or if `british` is used with [main=british, ngerman]
- setspace
- enumitem

- mathtools with the `fleqn` option
- mathastext with the `defaultmathsizes` and `italic` options
- isomath
- if the `minimal` option is not used:
  - siunitx [detect-all, per-mode=reciprocal-positive-first]
    If babel's `british` is used [locale=UK] will be used, if `ngerman` [locale=DE]. Additionally the `range-phrase` will be set to either `to` or `bis` with spaces around it.

6 The MRTtab **package**

MRTtab provides means to typeset tables in a style similar to the ones in the scripts of the MRT. This includes:

- delimited by horizontal rules on top and below
- head rows are light grey and delimited by horizontal rules
- all horizontal rules have the same thickness
- no vertical rules (though not enforced)

The package provides an environment similar to tabular (section 6.1), an enhanced version of \cline (section 6.2), and an environment to typeset displayed tables with many options available (section 6.3).

6.1 The MRTtabular **environment**

The MRTtabular environment calls a patched tabular environment. The following differences exist:

- a hook is provided at the beginning and the end of each line
- above and below of it a \hline is placed
- it has an additional optional argument specifying the number of rows to be formatted as head rows.
- you can access the current row number
- automatic application of a stretch factor based on the stretch tabular key in subsection 6.6.1.

Any tabular environments inside of an MRTtabular are ordinary tabulars which neither have hooks nor row numbers. They might be affected by an outer \rowcolor or similar, though.

An ordinary description as done with other environments in this documentation:

MRTtabular Usage: \begin{MRTtabular}[⟨*valign*⟩]{⟨*preamble*⟩}[⟨*head rows*⟩] ...
                    \end{MRTtabular}

The first optional argument as well as the mandatory argument match the ones of a regular tabular environment. ⟨*head rows*⟩ specifies how many rows at the beginning of the environment should be formatted as head rows. If ⟨*head rows*⟩ is not specified, no head row will be formatted. No further markup is required for this formatting to take place. You should end your rows only with \\ to make the hook mechanism work (on which the head row markup relies).

MRTarray Usage: \begin{MRTarray}[⟨*valign*⟩]{⟨*preamble*⟩}[⟨*head rows*⟩] ...
                  \end{MRTarray}

This is to MRTtabular what array is to tabular, so all in all the same but every cell is formatted as maths and it should be used only inside of a maths context. All the hooks from MRTtabular work here, too.

\head Usage: \head{⟨*num*⟩}

Additionally to the optional argument of MRTtabular to set the first *n* rows as

head rows, you can use \head to set the next ⟨*num*⟩ rows as head rows. This does not only work at the beginning of the environment but anywhere you want. Alternatively you can use the macros described in section 6.4. It is not supported in other implementations of tables. If you use it in MRTtable it'll throw an undefined error if MRTtabular isn't used internally (e.g., when you use the long option or an alternative inner tabular-like environment via the env keys).

\MRTtabAddtoBoLHook

Usage: \MRTtabAddtoBoLHook{⟨*content*⟩}

You can add ⟨*content*⟩ to the Begin-of-Line hook with this macro. Bear in mind that the ⟨*content*⟩ should be fully expandable and not produce any text, if you want to use stuff like \multicolumn, \rowcolor, or \cline at the beginning of the line – as this hook will be executed prior to that and \noalign and \omit won't work in that case. If you need something unexpandable that doesn't produce text you can enclose it in \noalign. The addition is made locally.

\MRTtabClearBoLHook

Usage: \MRTtabClearBoLHook

Clears the Begin-of-Line hook locally.

\MRTtabAddtoEoLHook

Usage: \MRTtabAddtoEoLHook{⟨*content*⟩}

You can also add ⟨*content*⟩ to the End-of-Line hook. Here it should not matter whether the contents are expandable or not, as it is impossible that something follows in the same row which can't follow something unexpandable. The addition is made locally.

\MRTtabClearEoLHook

Usage: \MRTtabClearEoLHook

Clears the End-of-Line hook locally.

\MRTtabCurrentRow

Usage: \MRTtabCurrentRow

Returns the current row number in an MRTtabular expandably.

6.1.1 Known Bugs

Currently only one bug is known: If after the last head row there is only one additional row the bottom \hline will only be drawn if you end that last row with \\. If you have more rows following the last head row, it won't matter whether you end the last row with \\ or not.

6.2 The \MRTcline macro

\MRTcline

Usage: \MRTcline⟨*!*⟩[⟨*color*⟩]{⟨***⟩[⟨*color*⟩]⟨*<[⟨left skip⟩]*⟩⟨*>[⟨right skip⟩]*⟩⟨*cols*⟩}

Sets something like a \cline in the specified ⟨*cols*⟩.

In the mandatory argument the only mandatory element is the affected ⟨*cols*⟩.

The mandatory argument can include a comma separated list in which you

can repeat every optional argument you like as many times as you like. Additionally you can enclose the ⟨*cols*⟩ in curly braces and give another comma separated list there which then can only contain column specifications and none of the optional arguments using the optional arguments specified before that list. A valid column specification is a single column, or a column range separated by a -, so something like ⟨*start-end*⟩.

Both ⟨*color*⟩ arguments have the same effect, but the first applies to every specification in the list, while the second only affects the current list item. The ⟨*color*⟩ doesn't change the color of the line, but the color of the optional fill arguments. It defaults to either `tabulargray` if used inside the scope of head rows, or `white` else. If you give a ⟨*⟩ the current list item will be completely in the specified ⟨*color*⟩.

You can introduce a small skip on the left side if you specify a ⟨<⟩ which defaults to `.5\tabcolsep`, with the optional ⟨*left skip*⟩ you can customize that length. A small skip to the right can be introduced with ⟨>⟩, again of customizable width using ⟨*right skip*⟩.

You should only use one `\MRTcline` per line and specify every column you want in that.

If you don't give the optional ⟨!⟩ after `\MRTcline`, before anything else something like a `\hline` using ⟨*color*⟩ will be used to cover the full width of the tabular. This way you don't have to specify every column you want to color with ⟨*color*⟩ using the ⟨*⟩ type argument.

I hope you got that rather cryptic description (if you can supply a better description, message me as noted in section 1.1).

Here are a few examples of usage with comparison to a correct `\cline` usage. The source of each table is printed below it. The last example of `\MRTcline` is not possible with the standard `\cline` as far as I know.

a	b	c
d	e	f
g	h	i
j	k	l

```
\begin{MRTtabular}{lll}
  a & b & c\\
  \MRTcline{1-2}
  d & e & f\\
  g & h & i\\
  j & k & l\\
\end{MRTtabular}
```

a	b	c
d	e	f
g	h	i
j	k	l

```
\begin{MRTtabular}{lll}
  a & b & c\\
  \cline{1-2}
  \clineReveal
  d & e & f\\
  g & h & i\\
  j & k & l\\
\end{MRTtabular}
```

a	b	c
d	e	f
g	h	i
j	k	l

```
\begin{MRTtabular}{lll}[2]
  a & b & c\\
  \MRTcline{1-2}
  d & e & f\\
  g & h & i\\
  j & k & l\\
\end{MRTtabular}
```

a	b	c
d	e	f
g	h	i
j	k	l

```
\begin{MRTtabular}{lll}[2]
  a & b & c\\
  \MRTcline{<>1-2}
  d & e & f\\
  g & h & i\\
  j & k & l\\
\end{MRTtabular}
```

a	b	c
d	e	f
g	h	i
j	k	l

```
\begin{MRTtabular}{lll}[2]
  a & b & c\\
  \cline{1-2}
  \arrayrulecolor{tablegray}
  \cline{3-3}
  \arrayrulecolor{black}
  \clineReveal
  \rowcolor{tablegray}
  d & e & f\\
  g & h & i\\
  j & k & l\\
\end{MRTtabular}
```

6.3 The `MRTtable` environment

The `MRTtable` environment is a wrapper around an `MRTtabular` inside of a `table` environment. There might be a severe difference in the implementation of the `long` version but for the user this shouldn't be noticeable. Most importantly the provided hooks at the start and end of each line should work in the `long` version, too, however they won't work if you use another inner `tabular`-like environment via the `env` keys.

`MRTtable` Usage: \begin{MRTtable}[⟨*key=value*⟩] ... \end{MRTtable}

`MRTtable` sets its contents in an `MRTtabular` environment. It features several ⟨*key*⟩s you are encouraged to use.

All available ⟨*key*⟩s are listed in subsection 6.6.1. An example can be seen in section 6.7.

6.4 Explicit head rows

It is possible to mark head rows explicitly. For this the following macros are provided:

`\headS` Usage: \headS

Start of the head rows. Sets a \hline above the current row except if the current row is the first row in a `MRTtabular` environment. Additionally the current row is coloured with \rowcolor{tablegray}.

\headR   Usage: \headR

An additional head row should be started with this macro. It sets the current row's colour to `tablegray`.

\headE   Usage: \headE

The end of the head rows. Should be used after the last row of the table's head but prior to the next row (immediately after \\).

\MRTtabDeclareHeadMacros   Usage: \MRTtabDeclareHeadMacros

By default the above macros are only available inside of `MRTtabular` and in the body of `MRTtable`. \MRTtabDeclareHeadMacros will make them locally available.

6.5 Other package macros

\MRTtabSetup   Usage: \MRTtabSetup{⟨*key=value*⟩}

This is the interface to set the options listed in subsection 6.6.1 outside of `MRTtable`.

\clineReveal   Usage: \clineReveal

As you can see in section 6.2 the macro \clineReveal is used. This is done because a \cline doesn't take up any vertical space (by issuing \noalign{\vskip-\arrayrulewidth}) as opposed to a \hline. This is done so that multiple \clines can be used in the same row. As a result the spacing is inconsistent and a \cline is overlapped by a following \rowcolor or \cellcolor. \clineReveal does introduce a vertical skip which reveals the lines (issuing \noalign{\vskip\arrayrulewidth}). It is also used by \MRTcline.

\MRTtabRepeatCols   Usage: \MRTtabRepeatCols

This macro is to be used in column definitions of `tabular`s or `array`s and other macros and environments using these internally (e.g. `MRTtabular` and `MRTtable`). The effect is that the column definitions which follow this macro are repeated indefinitely to match the required columns for the tables body. E.g., `l \MRTtabRepeatCols c` does set the first column left aligned and every following column centred. It has to be preceded by at least one valid column definition.

⚠ It is known that the used approach doesn't work with `longtable` and as such, it also doesn't work if you use it inside of `MRTtable` if it is using the `long` option.

6.6 Options

The package only features one load time option, which is `longtable`. If it is specified the `longtable` package is loaded and some more options of `MRTtable` become available which are

focused around the usage of `longtable` inside of `MRTtable`.

6.6.1 Setup Options

The following options are available for `\MRTtabSetup` and `MRTtable`.

---
align   = ⟨*align*⟩

If `no float` has been used, a `minipage` is used around the `MRTtable`. With the `align` option you can specify the vertical alignment of that `minipage`.

---
caption above   –NoArgument–
above

If specified the caption will be put above the `MRTtabular` in `MRTtable`. If `\KOMAoptions` is available the KOMA option `captions=tableheading` is used.

---
caption below   –NoArgument–
below

If specified the caption will be put below the `MRTtabular` in `MRTtable`. If `\KOMAoptions` is available the KOMA option `captions=tablesignature` is used.

---
bare   = ⟨*bool*⟩

If set to true the potential caption and the tabular like environment in `MRTtable` are neither surrounded by a `minipage` nor a `figure`. Only a `\centering` is issued.

BoL   = ⟨*content*⟩

Sets the `MRTtabular` Begin-of-Line hook using `\MRTtabAddtoBoLHook`

EoL   = ⟨*content*⟩

Sets the `MRTtabular` End-of-Line hook using `\MRTtabAddtoEoLHook`

---
caption   = ⟨*caption*⟩
cap

Specifies the content of the caption in an `MRTtable`. If it is blank, no caption will be used.

---
cline version   = ⟨*choice*⟩

set the behaviour of `\MRTcline`. Choices are `1` and `2`. `2` is the behaviour currently described in section 6.2. If you specify `1` the behaviour of the optional ⟨*!*⟩ of `\MRTcline` is reversed.

---
columns   = ⟨*preamble*⟩
col

Specifies the `MRTtabular` preamble (the column specifications). Defaults to first column `l`, others `c`.

---
env   = ⟨*name*⟩

Uses the tabular like environment ⟨*name*⟩ instead of `MRTtabular`. If an empty

argument is provided, no inner environment will be used. This is useful if you want to use an environment that grabs its contents and has to be explicitly used, e.g. `tabularx` can only be used like this. If you specify `MRTarray` as the argument, the math mode will automatically be set.

env begin = ⟨*begin*⟩

Uses ⟨*begin*⟩ as the start of the tabular like environment. This way you can specify some options. Note that any outer braces are stripped. If you want to use an environment you have to include `\begin` in the argument. Note that if the argument you provide is not empty, the column specification as defined with `columns` is inserted in braces after ⟨*begin*⟩.

env end = ⟨*end*⟩

Uses ⟨*end*⟩ as the end of the tabular like environment. This way you can specify some options. Note that any outer braces are stripped. If you want to use an environment you have to include `\end` in the argument.

float = ⟨*bool*⟩

If set true (the default and initial value) the `MRTtable` floats.

head rows = ⟨*num*⟩
head

The number of rows which should be formatted as head rows as in `MRTtabular`. In each `MRTtable` it is initially `1` – this differs from the behaviour of a stand alone `MRTtabular` which defaults to `0` rows.

in text sep = ⟨*skip*⟩

This controls the vertical space around a non-floating `MRTtable`. It is initially set to `\intextsep`. If it is equal to 0pt the `\vskip` is not issued.

label = ⟨*label*⟩

If `caption` is used the `MRTtable` will get the specified ⟨*label*⟩.

no float = ⟨*bool*⟩

The opposite of `float`. If set true the `MRTtable` will not float which is the default (but not initial) value.

no inner env –NoArgument–

Same result as `env begin={}`,`env end={}`, so no tabular like environment is used at all.

pos = ⟨*pos*⟩

The placement of a floating `MRTtable`. Initially `tbp`.

post tab = ⟨*content*⟩
post

A hook which is executed right after the `\end` of the inner `MRTtabular`.

pre tab pre	= ⟨*content*⟩ A hook which is executed right before the `\begin` of the inner `MRTtabular` (well, actually there is no explicit `\begin` following, but the next thing which is done is dependent of the used variant and other keys, so this is a close enough aproximation).
short caption short cap scap	–NoArgument– If caption and this option are used the list of tables will get this short caption instead of the caption.
stretch tabular stretch tab stretch	= ⟨*float*⟩ Sets the stretch in `MRTtabular` to the specified ⟨*float*⟩ using `\setstretch`. Should also apply to the contents of `MRTtable`, regardless of the inner environment.
stretch caption stretch cap cstretch	= ⟨*float*⟩ Sets the stretch in the caption using `\setkomafont` and `\setstretch`. Doesn't work if KOMA script is not used but issues a warning in that case.
striped	= ⟨*bool*⟩ If set to true the inner `MRTtabular` will be striped with `stripe color 1` and `stripe color 2`, beginning in line `stripe start`. It uses `\rowcolors` internally.
stripe color 1 stripe 1 scolor 1 scolor1	= ⟨*color*⟩ Defines the ⟨*color*⟩ of the first color argument of `\rowcolors` if `striped` is true. Initially set to `tablegray!50`.
stripe color 2 stripe 2 scolor 2 scolor2	= ⟨*color*⟩ Defines the ⟨*color*⟩ of the second color argument of `\rowcolors` if `striped` is true. Initially set to `white`.
stripe invert sinvert	–NoArgument– Exchanges the current values of `stripe color 1` and `stripe color 2`.
stripe start sstart	= ⟨*row*⟩ Defines the starting row of a potentially striped `MRTtabular`. Initially set to 2.

6.6.1.1 longtable **related options**

The following options are only available if the `longtable` option was used during package load time.

long   = ⟨*bool*⟩

If set true the `MRTtable` uses `longtable` internally. It doesn't float and gets page breakable. You should specify the `columns` of `MRTtable` manually as the automatic detection might fail terribly in conjunction with `longtable`.

continue caption   = ⟨*caption*⟩
cont cap
ccap

If specified following pages use this ⟨*caption*⟩ instead of the `short caption` or the normal `caption`.

continue with caption   = ⟨*bool*⟩
cont with cap
cont w cap

If set true, the following pages use the `caption` and not the `short caption` or `continue caption`. Defaults to true and initially is set to false.

continue text   = ⟨*text*⟩
cont text
ctext

The caption on following pages will be appended by this ⟨*text*⟩, this is true regardless of wether `caption`, `short caption` or `continue caption` is used. Initially this is set to (\emph{Fortsetzung}) (if MRTthesis with the English language is used, this will be set to (\emph{continued})).

6.7 Example

Table 6-1 shows an example usage of the `MRTtable` environment. The code to produce it is shown below. The `bare` option is used since I placed it manually inside of a `minipage` right of the verbatim listing.

```
\begin{MRTtable}
  [
    cap=Boring Table,
    label=tab:tab:example,
    bare
  ]
  This & is & the & boring & head \\
  This & is & the & first  & line \\
  This & is & the & second & line \\
  This & is & the & third  & line \\
  This & is & the & fourth & line \\
  This & is & the & fifth  & line \\
\end{MRTtable}
```

Table 6-1: Boring Table

This	is	the	boring	head
This	is	the	first	line
This	is	the	second	line
This	is	the	third	line
This	is	the	fourth	line
This	is	the	fifth	line

6.8 Dependencies

The package requires the following packages and their dependencies:

- expl3
- xparse
- array
- setspace
- xcolor with option `table`
- potentially longtable

7 The MRTfonts **package**

MRTfonts loads the fonts as they are used by the classes of this bundle, giving a uniform look to the documents.

7.1 **Options**

mathsizes	–NoArgument–

Opposite of `no mathsizes`. If used (which it by default is) the maths sizes are set according to the MS Word template. Note that those weren't set by mrtarbeit and if you alter the default font size won't be set.

no mathsizes	–NoArgument–

Opposite of `mathsizes`. If used the maths sizes are not changed from the defaults of the base class in use (so scrreprt, standalone or beamer).

sfacc	= ⟨*choice*⟩

⟨*choice*⟩ must be `height` or `list`. Sets the approach used by MRTsfacc (see chapter 10) and if `list` is in use the shift list for helvet will also be loaded. If it is not specified the `list` variant is used.

scale macro	–NoArgument–

If you use this option the macro `\scalemath` will be available. It is no longer needed for `\altl` but if you really need it this option can provide downward compatibility.

pmb	–NoArgument–

If you use this option the macro `\pmb` will be redefined to give, imho, better looking results. Of course, one shouldn't use `\pmb` anyway, but unfortunately the Greek letters used with `new maths` don't feature bold glyphs, so if you want to use bold Greek letters, you'll have to use it. If bm is loaded in the preamble its version of *poor man's bold* will be redefined, too. The effect is that instead of three overlapping slightly shifted symbols a whooping fourteen will be used. Compare and guess which one is the original and which one is the altered version:

$$\alpha \, \boldsymbol{\alpha} \, \boldsymbol{\alpha}$$

alt l	–NoArgument–

If this option is used the letter `l` in maths will result in the same as the `\altl` macro (see section 7.2). This option is used by default.

std l	–NoArgument–

If this option is used the letter `l` in maths will result in the same as the `\stdl` macro (see section 7.2).

<u>font</u>   = ⟨*font*⟩

This is only available if you're using LuaTeX or X∃TeX. With this you can set the used sans serif font, which will be used as the default font. It should be a font resembling Helvetica or Arial. Per default the font `TeX Gyre Heros` will be used.

<u>serif font</u>   = ⟨*font*⟩

This is only available if you're using LuaTeX or X∃TeX. With this you can set the used serif font. This isn't too important as the default fonts will suffice. You shouldn't have too much text with a Roman font anyway. Per default the font `Latin Modern Roman` will be used.

<u>mono font</u>   = ⟨*font*⟩

This is only available if you're using LuaTeX or X∃TeX. With this you can set the used mono font. This isn't too important as the default fonts will suffice. You shouldn't have too much text with a mono font anyway. Per default the font `Latin Modern Mono` will be used.

<u>new maths</u>   = ⟨*choice*⟩

This is only available if you're using pdfTeX. With this you can specify whether some special maths fonts are loaded. The result looks closer to the MS Word template for Greek letters and operators.

Available ⟨*choice*⟩s are `off` or `false` to turn this off, `on` or `true` to turn this on, and a valid float, to set the scale of the Greek letters and activate the feature. By default `1.05` will be used.

The number of usable maths fonts will be used exhaustively. newpxmath will be loaded with its options `upint`, `smallerops`, `nosymbolsc` and `noamssymbols` to get operators and the like, the maths letters of mathptmx will be loaded with a scale factor to get the Greek letters. The lmodern package will be loaded with the `nomath` option, and MRTlmscale will not be loaded at all.

Unfortunately newpxmath v1.401 from 2019-10-02 has a bug (that is already reported) in which the symbols of `\forall` and `\exists` aren't the correct ones (instead $\hbar$ and $\hbar$ are used). There might be other affected symbols, too, MRTfonts fixes these two for you.

<u>scale maths</u>   = ⟨*choice*⟩

This is only available if you're using pdfTeX. With this you can specify whether the MRTlmscale package should be loaded. Available ⟨*choice*⟩s are no argument, resulting in MRTlmscale being used with its default, `on` or `true` resulting in the same, `off` or `false` resulting in MRTlmscale not being used, and any valid float, resulting in MRTlmscale being used with the specified float as its scale factor. See chapter 11 for more about MRTlmscale. By default, MRTlmscale will be used with its default scale factor if lmodern is loaded with its maths fonts.

7.2 Macros

> **\stdl** Usage: `\stdl`
>
> `\stdl` will result in the lower case l from the helvet font in maths (`$\stdl$` results in *l*).

> **\altl** Usage: `\altl`
>
> `\altl` provides an alternative lower case l for use in maths which is distinct from an upper case I. Compare: *Il* (that is `$\stdl I$`) and *lI* (that is `$\altl I$`). There is no bold version of `\altl` provided by the package, nor any other maths alphabet version. Instead the standard fonts will be the ones used there.

> **\arev** Usage: `\arev{⟨symbols⟩}`
>
> This is another maths font (similar to `\mathbf` or `\mathcal`), that will use the maths font of arevmath, from which `\altl` is taken. Take a look at the following (the first group uses the standard maths fonts, the second is typeset using `\arev`):
>
> *abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ* $0123456789\alpha\beta\gamma\delta\varepsilon\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\varsigma\tau\upsilon\phi\chi\psi\omega\Gamma\Delta\Theta\Lambda\Xi\Sigma\Phi\Psi\Omega$
>
> *abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ* $0123456789\alpha\beta\gamma\delta\varepsilon\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\varsigma\tau\upsilon\phi\chi\psi\omega\Gamma\Delta\Theta\Lambda\Xi\Sigma\Phi\Psi\Omega$

> **\scalemath** Usage: `\scalemath{⟨float⟩}{⟨text⟩}`
>
> This is a version of `\scalebox` to be used in maths. It is only available if you use the `scale macro` option.

7.3 Dependencies

- expl3
- MRTif
- MRTutil
- MRTsfacc
- If the `scale macro` option is used graphics
- If X∃TEX or LuaTEX are used
    - fontspec [no-math] and additionally `\defaultfontfeatures{Ligatures= TeX}`
    - the fonts `TeX Gyre Heros`, `Latin Modern Roman` and `Latin Modern Mono`, dependent on the `font` keys.
  else
    - fontenc [T1]

- inputenc [utf8]
- helvet
- If `new maths` is used
    * lmodern [nomath]
    * newpxmath [upint, smallerops, nosymbolsc, noamssymbols]
    * The `ztmcm` font
  else
    * lmodern
    * MRTlmscale dependent on the `scale maths` key
- mathastext [italic, defaultmathsizes]
- isomath

8 The MRTif **package**

The MRTif package provides a number of expandable tests. In the following macros *TF* is used to specify that the macros exist with the endings T, F, and TF. The T ending stands for a ⟨*true*⟩ branch, F for the ⟨*false*⟩ branch.

If a macro name contains a G prior to *TF*, it strips any outermost groups prior to the test using \MRTifGroupTF. An N denotes that the first token in the argument is expanded once prior to any test. If a macro which takes two arguments ends with NN prior to the *TF* in both arguments the first token is expanded once, Nn and nN mean that only for the first and second argument, respectively, an expansion is made.

MRTif uses a special marker in some of its tests which expands to the undefined control sequence

                    \MRTif Error. Please report.

If you ever see this in your log or console output, please contact me as stated in section 1.1 and include a minimal example producing this behaviour in your contacting. Please do the same if you get any other undefined control sequence errors containing MRTif in the control sequence's name.

8.1 Macros

\MRTifCreateBranchingIfs

Usage: \MRTifCreateBranchingIfs{⟨*base*⟩}{⟨*branches*⟩}{⟨*args*⟩}{⟨*if*⟩}

This macro creates different branching if-tests from a TEX-like ⟨*if*⟩. ⟨*base*⟩ is the base name of the new macros (e.g. MRTifEmpty), ⟨*branches*⟩ is a comma separated list of different name extensions and branches (a detailed description follows), ⟨*args*⟩ is the number of arguments the TEX-like ⟨*if*⟩ will take and ⟨*if*⟩ should be a TEX-like ⟨*if*⟩, meaning an \if... test which would need a \fi.

The ⟨*branches*⟩ can be an arbitrary number (though more than four don't make sense) of ⟨*key*⟩=⟨*value*⟩ pairs separated by commas, you should make sure that they don't contain any spaces as those aren't stripped. ⟨*key*⟩ can be anything and will append the ⟨*base*⟩ name to form the individual macro name while ⟨*value*⟩ should be one of the values shown in Table 8-1. Each resulting macro name built from ⟨*base*⟩ and ⟨*key*⟩ must be undefined else it will be skipped (the other names might still become defined).

For example the \MRTifNumToken... macros are defined using:

\MRTifCreateBranchingIfs{MRTifNumToken}{TF=ab,T=yn,F=ny}{2}
  {\ifnum#1=\MRTtllength{#2} }

The space after {#2} is intended there to end the number parsing of \ifnum and does no harm (just in case you wondered). With TF=ab we define that the macro ending with TF should execute the first branch if the \ifnum yields true and else execute the second branch. Similarly T=yn defines the T macro to execute the next branch if \ifnum is true and else gobbles it.

\MRTifMathMode*TF*

Usage: \MRTifMathMode*TF*{⟨*true*⟩}{⟨*false*⟩}

Tests if you're currently in math mode. This is just a wrapper around the primitive \ifmmode.

Table 8-1: Possible ⟨*value*⟩s in \MRTifCreateBranchingIfs's ⟨*branches*⟩ argument, how many branches they'll create and which branch will be executed.

⟨*value*⟩	number of branches	if true	if false
ab	2	first	second
ba	2	second	first
yn	1	first	gobble
ny	1	gobble	first

\MRTifEmpty*TF*
\MRTifEmptyG*TF*
\MRTifEmptyN*TF*
\MRTifEmptyGN*TF*

Usage: \MRTifEmpty*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if ⟨*arg*⟩ is completely empty.

\MRTifBlank*TF*
\MRTifBlankG*TF*
\MRTifBlankN*TF*
\MRTifBlankGN*TF*

Usage: \MRTifBlank*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if ⟨*arg*⟩ is completely empty or contains only spaces.

\MRTifGroup*TF*
\MRTifGroupN*TF*

Usage: \MRTifGroup*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if ⟨*arg*⟩ is a single group no matter what the contents of that group are. It ignores spaces around the group.

\MRTifGroupNoSpaces*TF*
\MRTifGroupNoSpacesN*TF*

Usage: \MRTifGroupNoSpaces*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if ⟨*arg*⟩ is a single group no matter what the contents of that group are. It doesn't ignore spaces around the group.

\MRTifStringsMatch*TF*
\MRTifStringsMatchNN*TF*
\MRTifStringsMatchNn*TF*
\MRTifStringsMatchnN*TF*
\MRTifStringsMatchG*TF*
\MRTifStringsMatchGNN*TF*
\MRTifStringsMatchGNn*TF*
\MRTifStringsMatchGnN*TF*

Usage: \MRTifStringsMatch*TF*{⟨*string₁*⟩}{⟨*string₂*⟩}{⟨*true*⟩}
          {⟨*false*⟩}

Tests if ⟨*string₁*⟩ and ⟨*string₂*⟩ match, the strings are \detokenized prior to the comparison.

\MRTifStringsMatchXX*TF*
\MRTifStringsMatchXXG*TF*

Usage: \MRTifStringsMatchXX*TF*{⟨*string₁*⟩}{⟨*string₂*⟩}{⟨*true*⟩}
          {⟨*false*⟩}

Tests if ⟨*string₁*⟩ and ⟨*string₂*⟩ match, the strings are fully expanded.

\MRTifOneToken*TF*
\MRTifOneTokenG*TF*
\MRTifOneTokenN*TF*
\MRTifOneTokenGN*TF*

Usage: \MRTifOneToken*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if ⟨*arg*⟩ is only a single token or group, leading spaces are ignored.

\MRTifOneTokenNoGroup*TF* \MRTifOneTokenNoGroupN*TF*	Usage: \MRTifOneTokenNoGroup*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is only a single token. A single group is also ⟨*false*⟩. A G version is not supplied for obvious reasons.

\MRTifTwoToken*TF* \MRTifTwoTokenG*TF* \MRTifTwoTokenN*TF* \MRTifTwoTokenGN*TF*	Usage: \MRTifTwoToken*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is exactly two tokens or groups, leading spaces and spaces between the first two tokens are ignored.

\MRTifNumToken*TF* \MRTifNumTokenG*TF* \MRTifNumTokenN*TF* \MRTifNumTokenGN*TF*	Usage: \MRTifNumToken*TF*{⟨*num*⟩}{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is exactly ⟨*num*⟩ tokens long. It uses \MRTtllength internally. Compared to \MRTifOneToken and \MRTifTwoToken this macro takes longer and the longer the tested ⟨*arg*⟩ the longer it takes. The G and N variants only work on ⟨*arg*⟩, ⟨*num*⟩ will not be changed.

\MRTifNumTokenS*TF* \MRTifNumTokenSG*TF* \MRTifNumTokenSN*TF* \MRTifNumTokenSGN*TF*	Usage: \MRTifNumTokenS*TF*{⟨*num*⟩}{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Like \MRTifNumTokenTF but this one uses \MRTtllengthS and therefore captures spaces.

\MRTifLetter*TF* \MRTifLetterG*TF* \MRTifLetterN*TF* \MRTifLetterGN*TF*	Usage: \MRTifLetter*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is a letter, meaning of category code 11.

\MRTifTokensMatch*TF* \MRTifTokensMatchNN*TF* \MRTifTokensMatchNn*TF* \MRTifTokensMatchnN*TF* \MRTifTokensMatchG*TF* \MRTifTokensMatchGNN*TF* \MRTifTokensMatchGNn*TF* \MRTifTokensMatchGnN*TF*	Usage: \MRTifTokensMatch*TF*{⟨*arg₁*⟩}{⟨*arg₂*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg₁*⟩ and ⟨*arg₂*⟩ are single tokens and if so compares them whether both tokens match. The variants without G test if one of the arguments is contained in a group. If that's the case the ⟨*false*⟩ branch is executed.

\MRTifDigit*TF* \MRTifDigitG*TF* \MRTifDigitN*TF* \MRTifDigitGN*TF*	Usage: \MRTifDigit*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is a single token and a digit.

\MRTifNumber*TF* \MRTifNumberG*TF* \MRTifNumberN*TF* \MRTifNumberGN*TF*	Usage: \MRTifNumber*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is a number, meaning it consists out of an optional + or − sign and digits.

\MRTifNumberNoSign*TF* \MRTifNumberNoSignG*TF* \MRTifNumberNoSignN*TF* \MRTifNumberNoSignGN*TF*	Usage: \MRTifNumberNoSign*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Same as \MRTifNumberTF but also returns ⟨*false*⟩ for a leading sign.

\MRTifFloat*TF* \MRTifFloatG*TF* \MRTifFloatN*TF* \MRTifFloatGN*TF*	Usage: \MRTifFloat*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ is a float, meaning it consists out of an optional + or – sign, optional digits, an optional decimal marker (.) and digits (which are again optional if there were digits prior to a decimal marker).
\MRTifFloatNoSign*TF* \MRTifFloatNoSignG*TF* \MRTifFloatNoSignN*TF* \MRTifFloatNoSignGN*TF*	Usage: \MRTifFloatNoSign*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Same as \MRTifFloatTF but also returns ⟨*false*⟩ for a leading sign.
\MRTifContainsGroup*TF* \MRTifContainsGroupG*TF* \MRTifContainsGroupN*TF* \MRTifContainsGroupGN*TF*	Usage: \MRTifContainsGroup*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ contains any braced groups.
\MRTifContainsSpace*TF* \MRTifContainsSpaceG*TF* \MRTifContainsSpaceN*TF* \MRTifContainsSpaceGN*TF*	Usage: \MRTifContainsSpace*TF*{⟨*arg*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests if ⟨*arg*⟩ contains spaces which are not enclosed by inner groups.
\MRTifTokenIn*TF* \MRTifTokenInNN*TF* \MRTifTokenInNn*TF* \MRTifTokenInnN*TF* \MRTifTokenInG*TF* \MRTifTokenInGNN*TF* \MRTifTokenInGNn*TF* \MRTifTokenInGnN*TF*	Usage: \MRTifTokenIn*TF*{⟨*token*⟩}{⟨*token list*⟩}{⟨*true*⟩}{⟨*false*⟩} Tests whether ⟨*token list*⟩ contains ⟨*token*⟩. The group variant only strips outer groups for ⟨*token list*⟩. Any inner group in ⟨*token list*⟩ is skipped (so one can hide tokens from this search). The test is slower than non-expandable alternatives doing the same because it scans ⟨*token list*⟩ recursively. ⟨*token*⟩ should be a single token, if it's empty the test is true, if it is a space \MRTifContainsSpaceTF is used, if it is more than a single token the test is false, also a single group as an argument yields false.
\MRTifTokenInDeep*TF* \MRTifTokenInDeepNN*TF* \MRTifTokenInDeepNn*TF* \MRTifTokenInDeepnN*TF*	Usage: \MRTifTokenInDeep*TF*{⟨*token*⟩}{⟨*token list*⟩}{⟨*true*⟩}{⟨*false*⟩} This does the same as \MRTifTokenInTF, except that there is no hiding! Therefore no G version is provided.
\MRTtllength \MRTtllengthN	Usage: \MRTtllength{⟨*arg*⟩} Expands to the number of tokens or groups inside of ⟨*arg*⟩. Unprotected spaces are ignored. The ordinary version needs two expansions while the N version needs four. A group is counted as one Token.
\MRTtllengthS \MRTtllengthSN	Usage: \MRTtllengthS{⟨*arg*⟩} Like \MRTtllength but this version counts spaces, too.
\MRTifFexp \MRTifFexpI \MRTifFexpII	Usage: \MRTifFexp{⟨*MRTif test*⟩}⟨*branches*⟩ These macros take an arbitrary expandable test and expand it in exactly two

steps of expansion. `\MRTifFexp` can be applied to any test, while `\MRTifFexpI` is meant to be used for tests having only one – a true or a false – branch and `\MRTifFexpII` is meant to be used for tests having two branches. ⟨*MRTif test*⟩ doesn't necessarily have to be a test provided by MRTif but can be any fully expandable test. Inside of ⟨*MRTif test*⟩ all the arguments necessary for the test should be contained but not the true or false branch. An example:

```
\MRTifFexp{\MRTifEmptyF{abc}}{false}
\MRTifFexp{\MRTifEmptyTF{abc}}{true}{false}
\MRTifFexpI{\MRTifEmptyF{abc}}{false}
\MRTifFexpII{\MRTifEmptyTF{abc}}{true}{false}
```

all expand to `false` after exactly two steps of expansion. `\MRTifFexpI` and `\MRTifFexpII` are more than thrice as fast as `\MRTifFexp` and of course each test takes longer with these added than without.[1] The advantage is the control over the needed expansion steps.

8.2 Dependencies

MRTif loads the pdftexcmds package to make the pdfTeX primitive `\pdfstrcmp` available as `\pdf@strcmp` for LuaTeX. Additionally it uses MRTutil.

---

[1] Benchmarking done with pdfTeX, version 3.14159265-2.6-1.40.19 (TeX Live 2018), on an Intel® Core™ i5-2540M with `\MRTifEmptyT` and `\MRTifEmptyTF` utilizing the l3benchmark package. To give some numbers: 2 `\MRTifFexpI` and 2 `\MRTifFexpII` added roughly 1.2 μs to the compile time of 1.8 μs for 2 `\MRTifEmptyT` and 2 `\MRTifEmptyTF` tests, each once empty and once not with empty branches, while 4 `\MRTifFexp` added 4.2 μs.

9 The MRTwuline **package**

The package provides a MS Word like looking line breakable underlining. It does so by using ulem or stackengine.

9.1 Options

tUline
tikzunderline

–NoArgument–

If this option is passed TikZ will be added as a required package and an alternative underlining macro defined called \tUline, see its description in .

9.2 Macros

\WUline

Usage: \WUline[⟨height⟩]{⟨text⟩}

This sets ⟨text⟩ and underlines it in a way that looks like MS Word underlining – at least in the headings. It is usable both in math mode and in text mode. Though in math mode you should use \underline.

In text mode the ulem package is used for the underline. In math mode stackengine is employed. In both cases you can use ⟨height⟩ to change the default height of the underlining. In text mode and math mode the needed ⟨height⟩ to achieve the same height of the line differs quite a lot. By default in math mode is used, in text mode .

While the text mode version is line breakable it disables automatic hyphenation in its argument, you can still use \- to set hyphenation points.

\tUline

Usage: \tUline[⟨height⟩][⟨overhang⟩][⟨thickness⟩]{⟨text⟩}

This macro can be used to underline bigger portions of text. You should never need it, I guess. Just use \WUline instead. If you need it, you'll have to use the package option tUline.

If you think you can use this one instead: It underlines ⟨text⟩ at the given ⟨height⟩ (default -0.35ex) with the given ⟨thickness⟩ (default 0.185ex). You can specify ⟨overhang⟩ (default 0pt) which is the width the line should be wider than a text line on each side. If you let any optional argument empty, the default is used. It is assumed that the lines are equally separated with \baselineskip – so if your material does stretch the baseline skip, you can't use \tUline. It needs at least two runs to be displayed correctly.

9.3 Dependencies

- expl3
- xparse
- stackengine
- scalerel
- MRTif
- MRTutil

- ulem with the normalem option
- if the tUline option is used:
    - TikZ
    - tikzpagenodes
    - The TikZ library calc

10 The MRTsfacc **package**

This package is provided to remedy an issue related with sans serif maths, to be more precise to fix the placement of `\mathaccentV`, which is internally used by macros such as `\hat` and `\dot` with amsmath loaded. It is therefore loaded by all three, MRTthesis, MRTbeam and MRTalone. The beamer class provides a fix for the same issue which is unfortunately only working for beamer's default font by fixing the font metrics (and as far as I know only works with pdfLATEX).

MRTsfacc has two different approaches by patching `\mathaccentV` to move the accent horizontally – either depending on the height of the accented character or a defined offset in a list of possible arguments.

The package is designed with mathastext with the `italics` option in mind. It might work for other sans serif maths solutions as well. It requires amsmath to be loaded. It is incompatible with the accents package, as that one changes the accents to no longer use `\mathaccentV` internally.

Independent on the used approach the accent macros check whether their argument is one meeting a special criterion (a character of category 11 or a known element). Furthermore both versions should detect whether the argument is just another accent macro nested so that in `\dot {\bar{a}}` the `\dot` would still find the a as a known argument. This nested usage works only if the nested macro uses `\mathaccentV` internally and each level of nesting is an exact match of the approach's criterion or does only contain two tokens or groups (so in above example the `\bar` and the `{a}`) with the first one being a `\mathaccentV` using macro.

10.1 Options

The package has the following options:

> <u>height</u> –NoArgument–
>
> > If this option is used the offset is dependent on the height of the accented character. Read the description in section 10.2.

> <u>list</u> –NoArgument–
>
> > If this option is used the offset is defined by a list of known arguments. Read the description in section 10.3.

> <u>notest</u> –NoArgument–
>
> > By default the package does test whether the definition of `\mathaccentV` meets the known definition from the amsmath package. If something does redefine `\mathaccentV` or the definition has changed but you're sure that MRTsfacc still works with (it redefines it anyway) you can deactivate that test with this option. If amsmath's definition of `\mathaccentV` has changed, please contact the me as described in section 1.1.

Every other option is passed on to `\MRTsfaccSet`, its description is included in subsection 10.2.1. This will have no effect if the `list` option is used.

10.2 `height` **Variant**

This variant checks whether the argument is a single character with category code 11. If this test does not return true, the shift isn't applied.

Table 10-1: Comparison of shifted accents against original placement with the use of the `height` variant.

original		shifted		original		shifted	
â	Â	â	Â	n̂	N̂	n̂	N̂
b̂	B̂	b̂	B̂	ô	Ô	ô	Ô
ĉ	Ĉ	ĉ	Ĉ	p̂	P̂	p̂	P̂
d̂	D̂	d̂	D̂	q̂	Q̂	q̂	Q̂
ê	Ê	ê	Ê	r̂	R̂	r̂	R̂
f̂	F̂	f̂	F̂	ŝ	Ŝ	ŝ	Ŝ
ĝ	Ĝ	ĝ	Ĝ	t̂	T̂	t̂	T̂
ĥ	Ĥ	ĥ	Ĥ	û	Û	û	Û
î	Î	î	Î	v̂	V̂	v̂	V̂
ĵ	Ĵ	ĵ	Ĵ	ŵ	Ŵ	ŵ	Ŵ
k̂	K̂	k̂	K̂	x̂	X̂	x̂	X̂
l̂	L̂	l̂	L̂	ŷ	Ŷ	ŷ	Ŷ
m̂	M̂	m̂	M̂	ẑ	Ẑ	ẑ	Ẑ

Table 10-1 shows the results of this approach. While this approach is easier to adapt to other fonts – one has to change only one parameter – it is always a compromise trying to match every character as good as possible.

10.2.1 Macros

`\<accent>`   Usage: `\<accent>`⟨*/!⟩{⟨*arg*⟩}

`\<accent>` can be any of the maths accent macros using `\mathaccentV` internally (e.g. `\bar`, `\dot`, etc.).
The ⟨*/!⟩ can either be * or ! or omitted entirely. If the starred version is used, the shift is enforced regardless of the argument, if the exclamation mark is given it is prohibited.

`\MRTsfaccSet`   Usage: `\MRTsfaccSet{⟨*float*⟩}`

The shift width depends on a multiple of the box's height. The multiple can be set with this macro and should be a valid float. This is tested using `\MRTifFloatTF`. The package default for this share is `0.25`.

10.3 `list` Variant

This variant checks whether the argument is a known element from a list in which the offset is defined in the unit of `mu`.

It has the advantage that you can define individual offsets for every argument. In addition not only characters can be added to the list but almost arbitrary stuff. The drawback is that everything has to be added that you want to be recognized. Table 10-2 shows the results of this

Table 10-2: Comparison of shifted accents against original placement with the use of the `list` variant.

original	shifted	original	shifted	original	shifted	original	shifted
$\hat{a}$	$\hat{A}$	$\hat{a}$	$\hat{A}$	$\hat{n}$	$\hat{N}$	$\hat{n}$	$\hat{N}$
$\hat{b}$	$\hat{B}$	$\hat{b}$	$\hat{B}$	$\hat{o}$	$\hat{O}$	$\hat{o}$	$\hat{O}$
$\hat{c}$	$\hat{C}$	$\hat{c}$	$\hat{C}$	$\hat{p}$	$\hat{P}$	$\hat{p}$	$\hat{P}$
$\hat{d}$	$\hat{D}$	$\hat{d}$	$\hat{D}$	$\hat{q}$	$\hat{Q}$	$\hat{q}$	$\hat{Q}$
$\hat{e}$	$\hat{E}$	$\hat{e}$	$\hat{E}$	$\hat{r}$	$\hat{R}$	$\hat{r}$	$\hat{R}$
$\hat{f}$	$\hat{F}$	$\hat{f}$	$\hat{F}$	$\hat{s}$	$\hat{S}$	$\hat{s}$	$\hat{S}$
$\hat{g}$	$\hat{G}$	$\hat{g}$	$\hat{G}$	$\hat{t}$	$\hat{T}$	$\hat{t}$	$\hat{T}$
$\hat{h}$	$\hat{H}$	$\hat{h}$	$\hat{H}$	$\hat{u}$	$\hat{U}$	$\hat{u}$	$\hat{U}$
$\hat{i}$	$\hat{I}$	$\hat{i}$	$\hat{I}$	$\hat{v}$	$\hat{V}$	$\hat{v}$	$\hat{V}$
$\hat{j}$	$\hat{J}$	$\hat{j}$	$\hat{J}$	$\hat{w}$	$\hat{W}$	$\hat{w}$	$\hat{W}$
$\hat{k}$	$\hat{K}$	$\hat{k}$	$\hat{K}$	$\hat{x}$	$\hat{X}$	$\hat{x}$	$\hat{X}$
$\hat{l}$	$\hat{L}$	$\hat{l}$	$\hat{L}$	$\hat{y}$	$\hat{Y}$	$\hat{y}$	$\hat{Y}$
$\hat{m}$	$\hat{M}$	$\hat{m}$	$\hat{M}$	$\hat{z}$	$\hat{Z}$	$\hat{z}$	$\hat{Z}$

approach.

10.3.1 Macros

\\<accent>

Usage: `\<accent>[⟨opt⟩]{⟨arg⟩}`

`\<accent>` can be any of the maths accent macros using `\mathaccentV` internally (e.g. `\bar`, `\dot`, etc.).

⟨opt⟩ can either be a defined element from the list or a length in the unit of `mu`. So with `foo` a defined list element, both `\hat[foo]{bar}` and `\hat[4mu]{bar}` would be valid. If ⟨opt⟩ is a known element the offset of that element is used regardless of the given ⟨arg⟩ else (if it is used) the given length is used as the offset. If the optional argument isn't used at all, it'll be tested whether ⟨arg⟩ is a known element and if so the appropriate offset will be used. Else no offset will be applied.

\MRTsfaccShift

Usage: `\MRTsfaccShift{⟨element⟩}{⟨shift⟩}`

Adds ⟨element⟩ to the list of known arguments and saves ⟨shift⟩ for it. If ⟨element⟩ is already known it'll get redefined. ⟨shift⟩ has to be given in `mu`.

\MRTsfaccShiftLet

Usage: `\MRTsfaccShiftLet{⟨element₁⟩}{⟨element₂⟩}`

Adds ⟨element₁⟩ to the list of known arguments and defines the offset to be the one currently used by ⟨element₂⟩. ⟨element₂⟩ has to be known, if it isn't an error will be thrown.

Table 10-3: Available shift definition lists

List	To be used with
helvet	helvet and [italic,defaultmathsizes]mathastext

\MRTsfaccLoadShiftList

Usage: \MRTsfaccLoadShiftList{⟨*list*⟩}

The package comes with definitions for some fonts (see Table 10-3). With this macro you can load them. If you define a list for a font (or font combination) not listed in the table you might contact me as described in section 1.1 and I'll gladly add it to the package.

10.4 Additional macros

The package provides macros to use the accents used in text mode additionally in maths. Since the placement proves somewhat difficult – this might be caused by the bundle's author's insufficient knowledge – there is no really automated way to do so with a few macros. Instead you can define macros which will produce a symbol which is accented by one of the text accents.

\newsfhatmacro
\defsfhatmacro
\newsfcheckmacro
\defsfcheckmacro
\newsftildemacro
\defsftildemacro
\newsfacutemacro
\defsfacutemacro
\newsfgravemacro
\defsfgravemacro
\newsfdotmacro
\defsfdotmacro
\newsfddotmacro
\defsfddotmacro
\newsfbrevemacro
\defsfbrevemacro
\newsfbarmacro
\defsfbarmacro

Usage: \newsfhatmacro<⟨*horizontal*⟩>[⟨*vertical*⟩]{⟨*cs*⟩}[⟨*type*⟩]
           {⟨*symbol*⟩}

The difference between the \def... and the \new... variant is that the former will not check whether the macro ⟨*cs*⟩ is already defined or not. With these macros you can locally create a ⟨*cs*⟩ that gets displayed as ⟨*symbol*⟩ with an accent based on the text font's variant of the accents. hat uses \^, check uses \v, tilde uses \~, acute uses \', grave uses \`, dot uses \., ddot uses \", breve uses \u and bar uses \=.
You can control the horizontal positioning of the accent using ⟨*horizontal*⟩, which should be a length in mu. If you don't provide ⟨*horizontal*⟩ the offset will be determined based on the rules of the used variant (see section 10.2 and section 10.3). ⟨*vertical*⟩ specifies the vertical shift of the accent and should be given in ex. If ⟨*vertical*⟩ is not given nothing special will happen (this might change in the future – for now it is best if you specify 0ex if you don't want to change the accents vertical placement).
⟨*type*⟩ is the math atom type to be used for the newly created ⟨*cs*⟩. You could use \mathord, \mathop, \mathbin, \mathrel, \mathopen, \mathclose, \mathpunct, \mathinner, or any other macro taking one argument.

\newsfaccmacro
\defsfaccmacro

Usage: \newsfaccmacro<⟨*horizontal*⟩>[⟨*vertical*⟩]{⟨*cs*⟩}[⟨*type*⟩]
           {⟨*accent*⟩}{⟨*symbol*⟩}

This is a more general variant of \newsfhatmacro and the like. With this macro you can specify the macro responsible for typesetting the accent using the ⟨*accent*⟩ argument. The specified ⟨*accent*⟩ should take at most one argument and this one will be empty.

The results of these macros heavily depend on the used font. For MRTthesis, using pdfLATEX, the results don't look too bad. For example one can define a \hateq:

```
\defsfhatmacro[-0.3ex]{\hateq}[\mathrel]{=}
```

We use `\mathrel` since a = is a relation and should be spaced like that. Additionally we move the accent down by `0.3ex`, which should give a good result in this case. The following formula uses this `\hateq` definition, `\mathrel{\hat{=}}` and the default = for comparison:

$$a \doteq a \,\hat{=}\, a = a$$

Unfortunately these accents don't look too good in combination with Greek letters (see for yourself: $\hat{\alpha}$ vs. $\hat{\alpha}$), and one probably shouldn't mix the two types of accents in a document. The decision which approach you use is up to you but you'd have to define a whole lot of custom macros for every character you might want to use accented.

10.5 Dependencies

MRTsfacc loads the MRTif package and uses its tests `\MRTifMathModeTF`, `\MRTifLetterGTF`, `\MRTifFloatTF`, `\MRTifStringsMatchXXTF` and `\MRTifTwoTokenTF`. It also depends on amsmath being loaded. Additionally it uses the MRTutil package for some of its macros' definitions.

11 The MRTlmscale **package**

This package provides the option to scale the maths font of lmodern in a similar way helvet is scalable. It is only meant to be used with the pdfTEX engine.

If you load it it applies the default scale of 1.176 47. This scale leads to a matching height of the Greek maths font compared to helvet when used in combination with mathastext. You can provide any other scale as a package option. The package only takes this one option and it is checked whether this option is a valid float with \MRTifFloatTF.

Here are the results of this scale:

Figure 11-1: Effects of the MRTlmscale package. On the left the unscaled font, right with the default scale applied.

11.1 Dependencies

The package uses the MRTutil package and \MRTifFloatTF from the MRTif package.

12 The MRTutil **package**

This package provides some utility functions. Those are meant to aid people defining their own macros and are used throughout other packages of this bundle. Every macro this package provides is at the code level so there are no real user facing macros. As a general rule of thumb the user level is therefore moved one layer down, user facing macros have a single @ in their names, while internal macros have at least two.

12.1 Defining Macros

Since the author of this bundle often finds the possibilities of the LATEX 2$_\varepsilon$ macro family of \newcommand too restricting, the package provides some macros which use the syntax of TEX's \def but still check whether the macro is already defined.

\MRTutil@def
\MRTutil@edef

Usage: \MRTutil@def[⟨*prefixes*⟩]⟨*cs*⟩⟨*args*⟩{⟨*definition*⟩}

Those are versions of \def and \edef. You can define ⟨*prefixes*⟩ like \long or \protected. ⟨*cs*⟩ is the new control sequence's name, ⟨*args*⟩ is the argument specification and ⟨*definition*⟩ is the replacement text of the macro. Both check whether the macro is already defined and will raise an error if they are. They are like a fusion of \newcommand and \def in that they only define a new command but keep the versatility of \def. There is a usage example in subsection 12.2.1.

12.2 Optional Argument Parsing

Since the author really likes what xparse allows in defining macros with many optional arguments but doesn't want to force the complete expl3 onto the user (since it's huge), if a user is only interested in one or two of the small packages of this bundle there are some macros in those packages which have multiple optional arguments. To define those the following macros were created to provide a very limited subset of xparse's functionality. Note that none of the provided macros allow expandable definitions and none checks for matched delimiters (so if you want to pass in a [and a] to a macro with an optional argument delimited by [] you'll need to use [{ [foo]}], this is the same behaviour encountered in LATEX 2$_\varepsilon$'s optional arguments).

\MRTutil@Oarg

Usage: \MRTutil@Oarg{⟨*default*⟩}{⟨*continue*⟩}

checks for a following optional argument in []. If there is none it provides the ⟨*default*⟩. ⟨*continue*⟩ will be executed after the argument has been parsed. The value of the optional argument will be provided to ⟨*continue*⟩ in braces ({}).

\MRTutil@oarg

Usage: \MRTutil@oarg{⟨*continue*⟩}

Like \MRTutil@Oarg, but provides a special marker if there is no optional argument. You can check whether the special marker was provided with \MRTutil@ifmark. There is a usage example in subsection 12.2.1.

\MRTutil@Darg

Usage: \MRTutil@Darg⟨*token₁*⟩⟨*token₂*⟩{⟨*default*⟩}{⟨*continue*⟩}

Like \MRTutil@Oarg, but the optional argument is delimited by ⟨*token₁*⟩ and

⟨*token₂*⟩. So `\MRTutil@Darg<>{}\foo` will check whether there is an optional argument delimited by `<>` and if there is none will use an empty one. The result is provided to `\foo`.

\MRTutil@darg   Usage: `\MRTutil@darg⟨token₁⟩⟨token₂⟩{⟨continue⟩}`

Like `\MRTutil@Darg`, but provides a special marker if there is no optional argument. You can check whether the special marker was provided with `\MRTutil@ifmark`.

\MRTutil@Earg   Usage: `\MRTutil@Earg⟨token⟩{⟨default⟩}{⟨continue⟩}`

checks for a following optional argument which is indicated by a leading ⟨*token*⟩. If there is a ⟨*token*⟩ following, the following argument will be grabbed according to TeX's rules (so a single token might follow or a group delimited by `{}`). If there is no ⟨*token*⟩ following it provides ⟨*default*⟩. ⟨*continue*⟩ will be executed after the argument has been parsed. The value of the optional argument will be provided to ⟨*continue*⟩ as it is present in the input stream (so if it is braced there will be braces, if it's just a single token, there will be only a single token).

\MRTutil@earg   Usage: `\MRTutil@earg⟨token⟩{⟨continue⟩}`

Like `\MRTutil@Earg`, but if there is no ⟨*token*⟩ following instead of a default, a special marker will be provided. You can check whether the special marker was provided with `\MRTutil@ifmark`.

\MRTutil@targ   Usage: `\MRTutil@targ⟨token⟩{⟨continue⟩}`

checks for a following ⟨*token*⟩. If that ⟨*token*⟩ follows it will be gobbled and ⟨*continue*⟩ will get a special marker as its argument. Else another marker will be provided that would result in `false` in `\MRTutil@ifmark`. This is similar to LaTeX 2ε's `\@ifstar`.

\MRTutil@ifmark   Usage: `\MRTutil@ifmark{⟨test⟩}{⟨true⟩}{⟨false⟩}`

Tests whether ⟨*test*⟩ is the special marker provided by `\MRTutil@oarg` and similar. If so expands to ⟨*true*⟩, if not it expands to ⟨*false*⟩.

\MRTutil@defOargpair   Usage: `\MRTutil@defOargpair{⟨cs₁⟩}{⟨cs₂⟩}⟨token₁⟩⟨token₂⟩`

This defines ⟨*cs₁*⟩ to be similar to `\MRTutil@Oarg` and ⟨*cs₂*⟩ to be similar to `\MRTutil@oarg`, but using ⟨*token₁*⟩ and ⟨*token₂*⟩ as delimiters. The resulting macros should be faster than `\MRTutil@Darg` and `\MRTutil@darg`. Additionally to ⟨*cs₁*⟩ and ⟨*cs₂*⟩ a macro called ⟨`\cs₁`⟩ (with an additional backslash in its name) will be defined that does the argument grabbing. ⟨*cs₁*⟩ and ⟨*cs₂*⟩ should include the leading backslash.

\MRTutil@defOarg
\MRTutil@defoarg   Usage: `\MRTutil@defOarg{⟨cs⟩}⟨token₁⟩⟨token₂⟩`

These can be used if only one of the equivalents of `\MRTutil@Oarg` or

\MRTutil@oarg is needed. Additionally to ⟨cs⟩ a macro called ⟨\cs⟩ (with an additional backslash in its name) will be defined that does the argument grabbing. ⟨cs⟩ should include the leading backslash. There is a usage example in subsection 12.2.1.

12.2.1 Example

In the following example we'll define a macro that takes two optional arguments – one delimited by [] and one delimited by <> – and a mandatory one. We'll need one auxiliary macro per optional argument.

First we define the front facing macro named \ourmacro. It is defined \protected, because it isn't expandable since \MRTutil@oarg isn't.

```
\makeatletter
\MRTutil@def[\protected]\ourmacro{\MRTutil@oarg{\ourmacro@a}}
```

Next we define the second step that looks for the second optional argument, again defined \protected, but this time also \long, because it takes arguments and those might contain an explicit or implicit \par. We pipe through the first optional argument. Since we think that \ourmacro is gonna be used pretty often we want the argument grabbing for the second optional argument to be fast, so instead of using \MRTutil@darg<> we define our own test macro \ourmacro@opt (the speed gain is really small though).

```
\MRTutil@defoarg\ourmacro@opt<>
\MRTutil@def[\protected\long]\ourmacro@a#1{\ourmacro@opt{\ourmacro@b{#1}}}
```

The last step is the one which takes both optional arguments and the mandatory one. This one doesn't have to be defined \protected, because it is expandable.

```
\MRTutil@def[\long]\ourmacro@b#1#2#3%
  {%
    \MRTutil@ifmark{#1}
      {No 1st optional argument provided}
      {1st optional argument: #1}%
    \par
    \MRTutil@ifmark{#2}
      {No 2nd optional argument provided}
      {2nd optional argument: #2}%
    \par
    Mandatory argument: #3%
  }
\makeatother
```

As you can see, the macro doesn't do anything special, it just lists its arguments. A few usage examples are shown in Table 12-1.

Table 12-1: Usage examples of `\ourmacro`

Macro call	Output
`\ourmacro{baz}`	No 1st optional argument provided No 2nd optional argument provided Mandatory argument: baz
`\ourmacro[foo]{baz}`	1st optional argument: foo No 2nd optional argument provided Mandatory argument: baz
`\ourmacro<bar>{baz}`	No 1st optional argument provided 2nd optional argument: bar Mandatory argument: baz
`\ourmacro[foo]<bar>{baz}`	1st optional argument: foo 2nd optional argument: bar Mandatory argument: baz