# The source of the kulemt class*

Luc Van Eycken

## Contents



*This document corresponds to kulemt v2.0.0, dated 2025-04-03.

# File I
# kulemt-version.dtx
# The version

The kulemt file info is stored in a separate file so it can be input in style files, code descriptions and manuals.

\filename
\filedate
\fileversion
\fileinfo

These variables store the information one normally gets from `\GetFileInfo`. Make sure you don't use `\GetFileInfo` in your `.dtx` files or their contents will be overridden.

## 1 The `kulemt.cls` file info

\filename
\filedate
\fileversion
\fileinfo

Since we don't use the kulemt document class for typesetting the code, we can't use `\GetFileInfo`. Therefore we simply provide the information one normally gets from it. Plain TeX and LaTeX commands are used to make sure it always works.

```
⟨*fileinfo⟩
\def\filename{kulemt.cls}
\def\filedate{2025-04-03}
\def\fileversion{2.0.0}
\edef\fileinfo{KU\space Leuven\space Engineering\space Master's\space
  Thesis\space document\space class}
⟨/fileinfo⟩
```

(*End of definition for* `\filename` *and others. These variables are documented on page* 4.)

# File II
# kulemt-start.dtx
# Starting the preamble

We require at least the LaTeX version of TeX Live 2019 (LaTeX 2018-12-01) to ensure that the default input encoding is UTF-8 and all 'pdfTeX utilities' (incuding `\expanded`) are available in pdfTeX, LuaTeX and XeTeX. It also makes the rollback concept available for packages and classes. As a consequence, we assume that the required packages are not older than their TeX Live 2019 version (expl3 version 2019-04-06).

A rollback to the previous version of kulemt (dated 2022) is possible. However this also means that the configuration file of that version is used, which is no longer updated to the current situation.

`\legacy_if:nTF` This useful function seems to be missing from the 2019-04-06 kernel, so it is provided here.

`\IfFormatAtLeastTF` `\IfPackageLoadedTF` `\ProcessKeyOptions` Apart from starting the preamble, some compatibility functions are provided. These functions are present in recent LaTeX versions, but not yet in the 2019 version.

`\kulemt_at_end_preamble:n`
`\kulemt_after_begin_document:n`

Also hooks were defined later. We need the ones to postpone execution of commands to just before or after '`\begin{document}`'. So we provide our own emulation for older LaTeX versions.

`\kulemt_keys_key: ⋆` Since August 2020 the variable `\l_keys_key_str` replaces `\l_keys_key_tl`. The function `\kulemt_keys_key:` always expands to the key as a string.

## 1 Starting the kulemt preamble

The class starts with checking that the LaTeX is recent enough.

```
⟨*initial⟩
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
```

Next, we allow a rollback to version 1. Version 1.0 has been made available in March 2010. The current version 2 should be used from 2025 on.

```
\DeclareRelease{v1}{2010-03-02}{kulemt-v1.cls}
\DeclareCurrentRelease{}{2025-01-01}
```

The class then loads expl3 and xparse if needed. We also check the minimal release date of l3kernel. Since these packages are preloaded in recent LaTeX formats, `\ExplFileDate` must be used.
The minimal version of the L3 programming layer is the one where also the e-type argument is supported.

```
\@ifundefined{ExplFileDate}{\RequirePackage{expl3}}{}
\@ifl@t@r\ExplFileDate{2019-04-06}{}{%
```

```
  \ClassError{kulemt}%
    {The L3 programming layer is too old}%
    {You need to update your installation of 'l3kernel'.\MessageBreak
     Loading 'kulemt' will abort.}%
  \endinput}
\@ifundefined{NewDocumentCommand}{\RequirePackage{xparse}}{}
```
⟨/initial⟩

Then the class is identified. At this point the file info (from `kulemt-version.dtx`) must be known.

```
⟨*class⟩
⟨@@=kulemt⟩
\ProvidesExplClass{kulemt}{\filedate}{\fileversion}{\fileinfo}
```

# 2 Compatibility with more recent kernel versions

`\legacy_if:nTF` This function seems to be missing from the TeX Live 2019 version of expl3.

```
\cs_if_free:NT \legacy_if:nTF
  {
    \prg_new_conditional:Npnn \legacy_if:n #1 { TF }
      {
        \exp_after:wN \reverse_if:N
          \cs:w if#1 \cs_end:
          \prg_return_false:
        \else:
          \prg_return_true:
        \fi:
      }
  }
```

(*End of definition for* `\legacy_if:nTF`. *This function is documented on page 5.*)

`\kulemt_keys_key:` Since February 2020 the string `\l_keys_key_str` replaces the token list `\l_keys_key_tl`. The function `\kulemt_keys_key:` expands to the newer `\l_keys_key_str` or the older `\l_keys_key_tl`, whichever is available.

```
\cs_new:Nn \kulemt_keys_key:
  {
    \cs_if_exist_use:NF \l_keys_key_str
      { \tl_to_str:V \l_keys_key_tl }
  }
```

(*End of definition for* `\kulemt_keys_key:`. *This function is documented on page 5.*)

The October 2023 version switches from x- to e-type variants, although for some functions the x-type variants are kept. For no longer existing x-type variants we prefer to generate the e-type variant because this is more future-proof. To ease maintenance, it is done in each `.dtx` file as needed.

# 3 Compatibility with more recent LaTeX versions

`\IfFormatAtLeastTF` This command is available in LaTeX since October 2020.

```
\cs_if_exist:NF \IfFormatAtLeastTF
  { \cs_new:Npn \IfFormatAtLeastTF { \@ifl@t@r \fmtversion } }
```

\IfPackageLoadedTF This command is available in LaTeX since October 2021.

```
\cs_if_exist:NF \IfPackageLoadedTF
  { \cs_set_eq:NN \IfPackageLoadedTF \@ifpackageloaded }
```

\ProcessKeyOptions This command is available in LaTeX since June 2022. If not available, use the equivalent command from l3keys2e.

```
\cs_if_exist:NF \ProcessKeyOptions
  {
    \RequirePackage{l3keys2e}
    \NewDocumentCommand \ProcessKeyOptions { O{\@currname} }
      { \exp_args:No \ProcessKeysOptions {#1} }
  }
```

# 4 Using hooks

Since October 2020 a hook management system was added to LaTeX. For older versions of LaTeX we provide an emulation.

\kulemt_at_end_preamble:n
\l__kulemt_begindocument_before_tl

This function appends information to the hook 'begindocument/before'. For older LaTeX versions we provide our own hook \l__kulemt_begindocument_before_tl at the beginning of the \document command.

```
\IfFormatAtLeastTF {2020-10-01}
  {
    \cs_new_protected:Nn \kulemt_at_end_preamble:n
      { \AddToHook {begindocument/before} {#1} }
  }
  {
    \tl_new:N \l__kulemt_begindocument_before_tl
    \cs_new_protected:Nn \kulemt_at_end_preamble:n
      { \tl_put_right:Nn \l__kulemt_begindocument_before_tl {#1} }
    \tl_put_left:Nn \document
      {
        \group_end:
        \l__kulemt_begindocument_before_tl
        \group_begin:
      }
  }
```

(*End of definition for* \kulemt_at_end_preamble:n *and* \l__kulemt_begindocument_before_tl. *This function is documented on page* 5.)

\kulemt_after_begin_document:n
\l__kulemt_begindocument_end_tl

This function appends information to the hook 'begindocument/end'. For older LaTeX versions we provide our own hook \l__kulemt_begindocument_end_tl at the end of the \document command.

```
\IfFormatAtLeastTF {2020-10-01}
  {
    \cs_new_protected:Nn \kulemt_after_begin_document:n
      { \AddToHook {begindocument/end} {#1} }
  }
  {
    \tl_new:N \l__kulemt_begindocument_end_tl
    \cs_new_protected:Nn \kulemt_after_begin_document:n
```

```
      { \tl_put_right:Nn \l__kulemt_begindocument_end_tl {#1} }
    \tl_put_right:Nn \document
      {
        \l__kulemt_begindocument_end_tl
        \ignorespaces
      }
}
```

(*End of definition for* `\kulemt_after_begin_document:n` *and* `\l__kulemt_begindocument_end_tl`. *This function is documented on page* 5.)

⟨/class⟩

File III

# kulemt-cfg.dtx
# Reading the configuration file

This module contains the user commands to get data from the configuration file or to print it.

## 1 Format of the configuration file

The configuration file is an INI file.[1] The file contains lines with key-value pairs separated by an equal sign '`=`' and grouped in sections. Each section name is on a line itself and surrounded by brackets '`[...]`'. The section name will be used as ⟨*master abbreviation*⟩. Section names, keys and values are case sensitive. LaTeX constructs are not allowed in the configuration file, but UTF-8 characters are. So, if you want to keep words together on a typeset line, you can use a "NO-BREAK SPACE" (aka "Nonbreaking space" in MS Word) and not a '`~`'.

Initial spaces on a line are ignored as well as empty lines and comment lines (the lines starting with a semicolon '`;`'). Leading and trailing spaces on keys and values are also ignored. If a line does not contain a '`=`', it is assumed to be a continuation line of the previous value. This implies that you cannot use a continuation line containing a '`=`'.

The section `[defaults]` contains the default values for the keys. This implies that '`defaults`' cannot be used as a ⟨*master abbreviation*⟩.

Keys consist of an alphanumeric word or words separated by a dot '`.`', without any intervening spaces. The word before the first dot is called the main key, the other ones are called subkeys. The following keys are currently recognized. Required keys are required for each master definition or they should be defined in the section `[defaults]`.

`name` (*conf. key*)

- `name`, with value ⟨*full official master name*⟩.
  The ⟨*full official master name*⟩ must be identical to the master's name in the program guide. This is a required key.

`type` (*conf. key*)

- `type`, with value '`initial`' or '`advanced`'.
  The type declares it to be either an initial master or an advanced master. This is a required key.

`language` (*conf. key*)

- `language`, with value ⟨*master language*⟩.
  The ⟨*master language*⟩ is the official language of the master. It is defined by the program guide. Currently the languages '`dutch`' and '`english`' are supported. This is a required key.

`option` (*conf. key*)
`option.`⟨*abbrev*⟩ (*conf. key*)

- `option`
  The subkeys of the `option` main key enumerate the abbreviations of the different options. The value gives the full official option name in the ⟨*master language*⟩. Whether the option is mentioned on the title page or not, depends on the requirements of the master program. A program can require it, allow it (in which case the

[1] https://en.wikipedia.org/wiki/INI_file

student decides) or forbid it. This requirement can be set by using the key `option` (without subkeys) with a value '`required`', '`allowed`' (the default) or '`forbidden`'.

`faculty` (*conf. key*)

- `faculty`, with value ⟨*full faculty name*⟩.
  Apart from the ⟨*full faculty name*⟩, the value can also be '`multi`' which indicates that the thesis is a combined work of multiple faculties.

`faculty.logo` (*conf. key*)
The subkey `.logo` can be used to refer to an image file of the faculty logo. At least a `.pdf` version should be available. If students also use `dvips`, a `.eps` should also be available. The logo will be used at its natural size, so the KU Leuven part of the logo must be 2 cm high.

`contact.address` (*conf. key*)
`contact.email` (*conf. key*)
`contact.phone` (*conf. key*)

- `contact`, always in combination with a subkey:
  - `contact.address`, with the full international address as a value;
  - `contact.email`, with a contact email address as value;
  - `contact.phone`, with an international contact phone number as value.

  The email address and the phone number are not used in the current template.

⟨*key*⟩`.from` (*conf. key*) Additionally the subkey `from` (as the only subkey) can be used to get information from another master definition, which appears earlier in the configuration file. The value is the ⟨*master abbreviation*⟩ we get the data from. This subkey is typically used for the key `contact`.

⟨*key*⟩.⟨*lang*⟩ (*conf. key*) An additional final subkey can be used to select different values for each ⟨*document language*⟩. E.g., `contact.address.dutch` defines the `contact.address` to be used when typeset in a Dutch context. Currently only '`dutch`' and '`english`' can be used as ⟨*document language*⟩. This ⟨*document language*⟩ is typically used for the keys `contact.address` and `faculty`.

Since masters and/or options can become obsolete after some time, please stick to the following rules about them.

- Masters or options which are removed from the KU Leuven program guide are also removed from the configuration file, or at least commented out.
- When a master is changed considerably or the set of options is changed, a new program is set up and the old one is phased out. Students who started in the old program can still continue in it. So the old program, which we call the obsolete program, is still valid as program for the master's thesis. We differentiate between the new and the obsolete program by adding a dot '`.`' followed by a year to the master or option abbreviation. By convention the year is the starting year of the last academic year the master or option was not obsolete.

⟨*key*⟩.⟨*year*⟩ (*conf. key*)

- To speed up checking, a master or an option is considered obsolete as soon as it has a dot '`.`' in its name abbreviation.

`date` (*conf. key*)
`text.`⟨`...`⟩ (*conf. key*)
The key-value pairs before the first section change the configuration data. Typically it contains the key `date`, which holds the date of the configuration file in ISO-format. Currently the only other keys are `text` keys, which hold the language specific predefined texts. The values for English and Dutch are already known, so you should only need them for other languages or when the predefined texts must be changed later on. The currently used configuration data is enumerated in table 1.

For an example of a configuration file, see the file `kulemt.ini`.

Table 1: Configuration data. All values are strings.

| key | value |
|---|---|
| `date` | date of the configuration file in ISO-format |
| `text.and.`⟨*language*⟩ | translation of “and” |
| `text.assessor`[`.plural`]`.`⟨*language*⟩ | designation for the assessor(s) |
| `text.assistant`[`.plural`]`.`⟨*language*⟩ | designation for the assistant(s) |
| `text.author`[`.plural`]`.`⟨*language*⟩ | designation for the author(s) |
| `text.copyright.`⟨*language*⟩ | copyright text |
| `text.acyear.pre.`⟨*language*⟩ | text before the academic year on the title page |
| `text.title.pre.`⟨*language*⟩ | text before the master name on the title page |
| `text.promoter`[`.plural`]`.`⟨*language*⟩ | designation for the promoter(s) |
| `text.publisher.pre.`⟨*language*⟩ | text before the list of publishers |

# 2 Using the configuration file

## 2.1 Public variables and constants

`\l_kulemt_cfg_prop` The variable `\l_kulemt_cfg_prop` holds a property list with all key-value pairs of general configuration data, which are independent of the master selected. This data is either predefined or set at the start of the configuration file before the first section. The currently supported key-value pairs are enumerated in table 1. The variable `\l_kulemt_cfg_prop` is only valid after reading the configuration file.

`\l_kulemt_master_prop` The variable `\l_kulemt_master_prop` holds a property list with all key-value pairs of the configuration file, which describe the current master. The supported key-value pairs are enumerated in table 2. The variable `\l_kulemt_master_prop` is only valid after calling the function `\kulemt_set_master:n`.

`\l_kulemt_masters_seq` The variable `\l_kulemt_masters_seq` contains a list of master abbreviations, which are known until now. It only contains the complete list of master abbreviations after calling the function `\kulemt_read_config_file:` to read the entire configuration file.

## 2.2 Getting data from the configuration file

The option variable `\l_kulemt_opt_cfgfile_tl` holds the name of the configuration file (see file `kulemt-opt.dtx`).

`\kulemt_read_config_file:` The function `\kulemt_read_config_file:` reads the entire configuration file. It deletes any previously read defaults and master information.

Table 2: Possible master items. All items are strings except `options`, which is a sequence of strings.

| key | value |
|---|---|
| `abbreviation` | master abbreviation |
| `contact.address`[`.`⟨*language*⟩] | full international address, possibly ⟨*language*⟩ specific |
| `contact.email` | contact email address |
| `contact.phone` | international contact phone number |
| `faculty`[`.`⟨*language*⟩] | full faculty name or '`multi`', possibly ⟨*language*⟩ specific |
| `faculty.logo`[`.`⟨*language*⟩] | faculty logo |
| `language` | master language |
| `name` | full official master name |
| `option` | '`required`', '`allowed`' or '`forbidden`' |
| `option.`⟨*name*⟩ | full official name of master option ⟨*name*⟩ |
| `options` | list of option ⟨*name*⟩s |
| `type` | '`initial`' or '`advanced`' |

`\kulemt_read_config_file:n` `\kulemt_read_config_file:n {`⟨*master abbreviation*⟩`}`

The function `\kulemt_read_config_file:n` reads from the configuration file only the configuration data, the defaults and the information for the master ⟨*master abbreviation*⟩. A fatal error is raised if the section `[`⟨*master abbreviation*⟩`]` is not found in the configuration file. The function has no effect if the information for the ⟨*master abbreviation*⟩ already exists.

`\kulemt_set_master:n` `\kulemt_set_master:n {`⟨*master abbreviation*⟩`}`

This function initializes the property list `\l_kulemt_master_prop` for the master ⟨*master abbreviation*⟩. The initialization is a prerequisite for any function getting or using master information. If needed, it calls the function `\kulemt_read_config_file:n`.

`\kulemt_master_get_item:nN` `\kulemt_master_get_item:nN {`⟨*key*⟩`}` ⟨*var*⟩

This function returns the value of the ⟨*key*⟩ for the current master in the variable ⟨*var*⟩. First the current language is added as a final subkey to ⟨*key*⟩. If this doesn't exist, the key without that final language subkey is tried. The ⟨*var*⟩ is set to `\q_no_value` if no such item is found for the current master. The type of ⟨*var*⟩ depends on the ⟨*key*⟩ (see table 2).

`\kulemt_master_get_item_or_fallback:nnN` `\kulemt_master_get_item_or_fallback:nnN {`⟨*key*⟩`} {`⟨*fallback*⟩`}` ⟨*tl var*⟩

This function returns in the variable ⟨*tl var*⟩ the value of the ⟨*key*⟩ for the current master. If the item is not found for the current master, a ⟨*fallback*⟩ value is returned. If ⟨*fallback*⟩ is empty, an educated guess for the fallback is returned.

`\kulemt_master_get_required_item:nN` `\kulemt_master_get_required_item:nN {`⟨*key*⟩`}` ⟨*tl var*⟩

This function returns the value of the ⟨*key*⟩ for the current master in the variable ⟨*tl var*⟩. Since the item is required, an error is issued if the item is not found for the current master.

\kulemt_master_get_faculty_name:N \kulemt_master_get_faculty_name:N ⟨*tl var*⟩

This function returns the faculty name for the current master in the variable ⟨*tl var*⟩. In case no faculty name is provided or it is 'multi', ⟨*tl var*⟩ is made empty.

\kulemt_master_obsolete_item:nTF \kulemt_master_obsolete_item:nTF {⟨*item*⟩} {⟨*true code*⟩} {⟨*false code*⟩}

Determines if the ⟨*item*⟩ is obsolete (i.e., contains a dot). This item can be a master abbreviation or an option subkey.

### 2.3 Typesetting data from the configuration file

\kulemt_titlecase_first:n \kulemt_titlecase_first:n {⟨*text*⟩}

\kulemt_titlecase_first:n prints the ⟨*text*⟩ with the first character converted to uppercase. It takes into account special forms in the current language, such as ij in Dutch which is converted to IJ. However the latter only works with a recent (2020) kernel.

\kulemt_cfg_print_text:n \kulemt_cfg_print_text:n {⟨*subkey*⟩}
\kulemt_cfg_print_text_ucfirst:n \kulemt_cfg_print_text_ucfirst:n {⟨*subkey*⟩}

\kulemt_cfg_print_text:n prints the configuration text based on the configuration key 'text.⟨*subkey*⟩.⟨*current language*⟩'. If the configuration key is not found, an error is raised. \kulemt_cfg_print_text_ucfirst:n does the same but also makes the first character uppercase.

\kulemt_cfg_print_text_from_opt:n \kulemt_cfg_print_text_from_opt:n {⟨*subkey*⟩}
\kulemt_cfg_print_text_from_opt_ucfirst:n \kulemt_cfg_print_text_from_opt_ucfirst:n {⟨*subkey*⟩}

\kulemt_cfg_print_text_from_opt:n calls \kulemt_cfg_print_text:n with ⟨*subkey*⟩ if the option ⟨*subkey*⟩ variable holds exactly one item and with '⟨*subkey*⟩.plural' otherwise. \kulemt_cfg_print_text_from_opt_ucfirst:n does the same but also makes the first character uppercase.

\kulemt_master_print_required_item:n \kulemt_master_print_required_item:n {⟨*key*⟩}

This function prints the value of the ⟨*key*⟩ for the current master. Since the item is required, an error is issued if the item is not found for the current master.

## 3 Implementation

⟨*class⟩
⟨@@=kulemt_cfg⟩

Some x-variants are since October 2023 version no longer available. We generate here the e-type variants for functions which don't exist yet and are used in this file.

```
\cs_generate_variant:Nn \exp_last_unbraced:Nn { Ne }
\cs_generate_variant:Nn \msg_error:nnn { nne }
\cs_generate_variant:Nn \msg_error:nnnn { nnee }
\cs_generate_variant:Nn \msg_error:nnnnn { nneee }
```

```
\cs_generate_variant:Nn \msg_fatal:nnn { nne }
\cs_generate_variant:Nn \msg_warning:nnnn { nnee }
\cs_generate_variant:Nn \prop_put:Nnn { Nne }
\cs_generate_variant:Nn \str_put_right:Nn { Ne }
\cs_generate_variant:Nn \str_set:Nn { Ne }
\cs_generate_variant:Nn \tl_set:Nn { Ne }
```

## 3.1 Public variables and constants

\l_kulemt_masters_seq

This variable holds a list of the master abbreviations known until now. It contains the list of all master abbreviations after reading the entire configuration file with `\kulemt_read_config_file:`.

```
\seq_new:N \l_kulemt_masters_seq
```

(*End of definition for* `\l_kulemt_masters_seq`. *This variable is documented on page 11.*)

\l_kulemt_master_prop

This variable holds the property list describing the current master. It is set by the function `\kulemt_set_master:n` to `\l__kulemt_cfg_master_`⟨*abbrev*⟩`_prop`, which stores the key-value pairs for the master with abbreviation ⟨*abbrev*⟩.
The keys are the full keys from the configuration file section `[`⟨*abbrev*⟩`]` with defaults from the configuration file section `[defaults]`. Apart from the full keys allowed by `\c__kulemt_cfg_allowed_key_seq`, the key `abbreviation` is added.

```
\prop_new:N \l_kulemt_master_prop
```

(*End of definition for* `\l_kulemt_master_prop`. *This variable is documented on page 11.*)

\l_kulemt_cfg_prop

This variable holds the property list describing the general configuration data. See table 1 on page 11 for a list of supported keys. It is initialized with the default data as strings. The initialization is needed before the configuration file is read!

```
\prop_new:N \l_kulemt_cfg_prop
\prop_set_from_keyval:Nn \l_kulemt_cfg_prop
  {
    date = ,
    text.and.dutch   = en ,
    text.and.english = and ,
    text.assessor.dutch   = evaluator ,
    text.assessor.english = assessor ,
    text.assessor.plural.dutch   = evaluatoren ,
    text.assessor.plural.english = assessors ,
    text.assistant.dutch   = begeleider ,
    text.assistant.english = assistant-supervisor ,
    text.assistant.plural.dutch   = begeleiders ,
    text.assistant.plural.english = assistant-supervisors ,
    text.author.dutch   = auteur ,
    text.author.english = author ,
    text.author.plural.dutch   = auteurs ,
    text.author.plural.english = authors ,
    text.copyright.dutch   = { Alle~ rechten~ voorbehouden.~ Niets~ uit~ deze~
      uitgave~ mag~ worden~ vermenigvuldigd~ en/of~ openbaar~ gemaakt~ worden~
      door~ middel~ van~ druk,~ fotokopie,~ microfilm,~ elektronisch~ of~ op~
      welke~ andere~ wijze~ ook~ zonder~ voorafgaande~ schriftelijke~
      toestemming~ van~ de~ uitgever.} ,
    text.copyright.english = { All~ rights~ reserved.~ No~ part~ of~ the~
      publication~ may~ be~ reproduced~ in~ any~ form~ by~ print,~ photoprint,~
```

```
      microfilm,~ electronic~ or~ any~ other~ means~ without~ written~
      permission~ from~ the~ publisher.} ,
    text.acyear.pre.dutch   = academiejaar ,
    text.acyear.pre.english = academic~ year ,
    text.title.pre.dutch   = thesis~ voorgedragen~ tot~ het~ behalen~
                             van~ de~ graad~ van ,
    text.title.pre.english = thesis~ submitted~ for~ the~ degree~ of ,
    text.promoter.dutch   = promotor ,
    text.promoter.english = supervisor ,
    text.promoter.plural.dutch   = promotoren ,
    text.promoter.plural.english = supervisors ,
    text.publisher.pre.dutch   = uitgegeven~ in~ eigen~ beheer~ door ,
    text.publisher.pre.english = published~ by
  }
\prop_map_inline:Nn \l_kulemt_cfg_prop
  { \prop_put:Nne \l_kulemt_cfg_prop {#1} { \tl_to_str:n {#2} } }
```

(*End of definition for* `\l_kulemt_cfg_prop`. *This variable is documented on page* 11.)

## 3.2 Helper functions

We provide some variants of LaTeX3 functions.

```
\cs_generate_variant:Nn \prop_put:Nnn { Ne }
\cs_generate_variant:Nn \seq_set_split:Nnn { Nee }
\prg_generate_conditional_variant:Nnn \seq_if_in:Nn { Ne } { T, F, TF }
```

`\__kulemt_cfg_str_seq_const_from_clist:Nn`

The configuration file lines are read as strings. However, `\seq_if_in:...` compares tokens not strings. Therefore we need to store the values also as strings in the constants with values.

```
\cs_new_protected:Nn \__kulemt_cfg_str_seq_const_from_clist:Nn
  { \exp_args:NNe \seq_const_from_clist:Nn #1 { \tl_to_str:n {#2} } }
```

(*End of definition for* `\__kulemt_cfg_str_seq_const_from_clist:Nn`.)

Strings in the configuration file are assumed to be UTF-8. But `pdflatex` reads the UTF-8 sequences as bytes and converts each byte separately to a string character: '`â`' becomes '`Ã¢`' in T1 font encoding. For LuaTeX or XeTeX this is not a problem.

`\__kulemt_cfg_str_with_utf:N`

To solve this problem, a helper function is provide to replace the UTF-8 string bytes in `#1` back to normal characters.

```
\cs_new_protected:Nn \__kulemt_cfg_str_with_utf:N
  {
    \tl_set_rescan:Nno #1
      {
        \ExplSyntaxOff
        \seq_map_inline:Nn \l_char_special_seq
          { \char_set_catcode_other:N ##1 }
        \char_set_catcode_space:n {9}
        \char_set_catcode_space:n {32}
      }
      { #1 }
  }
```

(*End of definition for* `\__kulemt_cfg_str_with_utf:N`.)

__kulemt_cfg_str_with_utf:nN

Of course we only want to do this for string values. This helper function first checks whether the key `#1` stores a string value before trying to replace characters in `#2`.

```
\cs_new_protected:Nn \__kulemt_cfg_str_with_utf:nN
  {
    \seq_if_in:NnF \l__kulemt_cfg_non_str_items_seq {#1}
      { \__kulemt_cfg_str_with_utf:N #2 }
  }
```

(*End of definition for* `\__kulemt_cfg_str_with_utf:nN`.)

\l__kulemt_cfg_non_str_items_seq

This variable holds a list of keys whose values does not contain a simple string.

```
\seq_const_from_clist:Nn \l__kulemt_cfg_non_str_items_seq { options }
```

(*End of definition for* `\l__kulemt_cfg_non_str_items_seq`.)

But all this is not needed for LuaTeX or XeTeX.

```
\bool_lazy_any:nT { \sys_if_engine_luatex_p: \sys_if_engine_xetex_p: }
  {
    \cs_set_eq:NN \__kulemt_cfg_str_with_utf:N \use_none:n
    \cs_set_eq:NN \__kulemt_cfg_str_with_utf:nN \use_none:nn
  }
```

## 3.3 Parsing the configuration file

### 3.3.1 Private variables and constants

\l__kulemt_cfg_current_section_name_str
\l__kulemt_cfg_current_key_str
\l__kulemt_cfg_current_value_str
\l__kulemt_cfg_current_section_prop

The variable `\l__kulemt_cfg_current_section_name_str` holds the name of the current section, the variable `\l__kulemt_cfg_current_key_str` the name of the current key being parsed, and the variable `\l__kulemt_cfg_current_value_str` the value of the current key so far. The property list `\l__kulemt_cfg_current_section_prop` holds the key-value pairs of the current section, parsed so far. These variables have only meaningful content during the parsing of a section of the configuration file.

```
\str_new:N  \l__kulemt_cfg_current_section_name_str
\str_new:N  \l__kulemt_cfg_current_key_str
\str_new:N  \l__kulemt_cfg_current_value_str
\prop_new:N \l__kulemt_cfg_current_section_prop
```

(*End of definition for* `\l__kulemt_cfg_current_section_name_str` *and others.*)

\l__kulemt_cfg_current_options_seq

A list of the `option` subkeys of the current section so far.

```
\seq_new:N \l__kulemt_cfg_current_options_seq
```

(*End of definition for* `\l__kulemt_cfg_current_options_seq`.)

\l__kulemt_cfg_section_defaults_prop

The property list `\l__kulemt_cfg_section_`⟨*name*⟩`_prop` stores the key-value pairs for the section `[`⟨*name*⟩`]`, which is either a `[`⟨*master abbreviation*⟩`]` or `[defaults]`. Each key is a main key of the section and its value is again a property list with as key the concatenated subkeys and as value the value of the section key. The reason for having this complex organization, is that we want to easily replace entire main key contents. E.g., we want to avoid combining a specific contact address with a default contact phone number. (If you want to do this anyway, use "`contact.from=defaults`" first.)

```
\prop_new:N \l__kulemt_cfg_section_defaults_prop
```

(*End of definition for* `\l__kulemt_cfg_section_defaults_prop`.)

\c__kulemt_cfg_allowed_key_seq

This constant holds the valid keys, including subkeys. Additionally the key `option` can have subkeys, namely the abbreviation of an option name and the last year before coming obsolete. The subkey `from` as a single subkey is also allowed for all keys.
The list must be converted to strings because `\seq_if_in:...` compares token lists.

```
\__kulemt_cfg_str_seq_const_from_clist:Nn \c__kulemt_cfg_allowed_key_seq
  { name, type, language, option,
    faculty, faculty.dutch, faculty.english,
    faculty.logo, faculty.logo.dutch, faculty.logo.english,
    contact.address, contact.address.english, contact.address.dutch,
    contact.email, contact.phone }
```

(*End of definition for* `\c__kulemt_cfg_allowed_key_seq`.)

\c__kulemt_cfg_language_values_seq
\c__kulemt_cfg_type_values_seq
\c__kulemt_cfg_option_values_seq

The following constants hold the allowed values for the keys `language`, `type` and `option` without a subkey. It is assumed that for each of the languages hyphenation patterns are available, either loaded on demand (LuaTeX) or preloaded (the other formats).
The first list item is used instead of an incorrect value or when a value is missing for a required item.

```
\__kulemt_cfg_str_seq_const_from_clist:Nn \c__kulemt_cfg_language_values_seq
  { english, dutch }
\__kulemt_cfg_str_seq_const_from_clist:Nn \c__kulemt_cfg_option_values_seq
  { allowed, forbidden, required }
\__kulemt_cfg_str_seq_const_from_clist:Nn \c__kulemt_cfg_type_values_seq
  { initial, advanced }
```

(*End of definition for* `\c__kulemt_cfg_language_values_seq`, `\c__kulemt_cfg_type_values_seq`, *and* `\c__kulemt_cfg_option_values_seq`.)

\l__kulemt_cfg_tmp_tl

Temporary internal variable, used when printing an item.

```
\tl_new:N \l__kulemt_cfg_tmp_tl
```

(*End of definition for* `\l__kulemt_cfg_tmp_tl`.)

### 3.3.2 Reading the configuration file

\kulemt_read_config_file:

Public function to read the entire configuration file. It removes any previously read configuration information.
The name of the configuration file is taken from `\l_kulemt_opt_cfgfile_tl`.

```
\cs_new_protected:Nn \kulemt_read_config_file:
  {
    \prop_clear:N \l__kulemt_cfg_section_defaults_prop
    \seq_clear:N \l_kulemt_masters_seq
    \exp_args:NV \__kulemt_cfg_read_file:nn \l_kulemt_opt_cfgfile_tl {}
    \seq_if_empty:NT \l_kulemt_masters_seq
      { \msg_fatal:nne {kulemt} {cfg/no-masters} { \l_kulemt_opt_cfgfile_tl } }
  }
\msg_new:nnnn {kulemt} {cfg/no-masters}
  { No~ master~ definitions~ are~ found~ in~ the~ configuration~ file~ '#1'. }
  { Please~ correct~ the~ configuration~ file~ or~ select~ another~ one. }
```

(*End of definition for* `\kulemt_read_config_file:`. *This function is documented on page* 11.)

\kulemt_read_config_file:n

Public function to read from the configuration file the necessary information about one master as fast as possible. It adds this to the existing configuration information. If the section [⟨*master abbreviation*⟩] is not found in the configuration file, a fatal error is raised.

```
\cs_new_protected:Nn \kulemt_read_config_file:n
  {
    \seq_if_in:NeF \l_kulemt_masters_seq { \tl_to_str:n {#1} }
      {
        \exp_args:NV \__kulemt_cfg_read_file:nn \l_kulemt_opt_cfgfile_tl {#1}
        \seq_if_in:NeF \l_kulemt_masters_seq { \tl_to_str:n {#1} }
          { \msg_fatal:nnn {kulemt} {cfg/missing-master} {#1} }
      }
  }
\msg_new:nnnn {kulemt} {cfg/missing-master}
  { The~ master~ abbreviation~ '#1'~ is~ not~ defined~
    in~ the~ configuration~ file~ '\l_kulemt_opt_cfgfile_tl'. }
  { Perhaps~ you~ mistyped~ the~ abbrevation~ or~
    the~ configuration~ file~ needs~ updating. }
```

(*End of definition for* `\kulemt_read_config_file:n`*. This function is documented on page* 12.)

__kulemt_cfg_read_file:nn

This function reads the information for a specific ⟨*master abbreviation*⟩ (#2) from the configuration file ⟨*file name*⟩ (#1). If #2 is empty, the entire configuration file is read. Empty lines and comment lines (starting with a ';') in the configuration file are ignored. A section starts with a line containing a [⟨*section name*⟩]. Other lines define a new key or continue an existing key.

Due to the possibility of a subkey `from`, no sections can be skipped until the section [⟨*master abbreviation*⟩]. After that section, reading stops. We assume no relevant `[defaults]` sections can be found further down the line.

Note: `\l__kulemt_cfg_current_section_name_str` holds the name of the previous section when a '[' is encountered.

```
\cs_new_protected:Nn \__kulemt_cfg_read_file:nn
  {
    \file_if_exist:nF {#1}
      { \msg_fatal:nnn {kulemt} {cfg/file-not-found} {#1} }
    \str_clear:N \l__kulemt_cfg_current_section_name_str
    \ior_open:Nn \g_tmpa_ior {#1}
    \ior_str_map_inline:Nn \g_tmpa_ior
      {
        \str_set:Ne \l_tmpa_str { \str_head_ignore_spaces:n {##1} }
        \str_case:VnF \l_tmpa_str
          {
            {}  {}
            {;} {}
            {[} {
                  \__kulemt_cfg_finish_read_section:
                  \tl_if_empty:nTF {#2}
                    { \__kulemt_cfg_start_read_section:nn {##1} {#2} }
                    {
                      \str_if_eq:VnTF
                        \l__kulemt_cfg_current_section_name_str {#2}
                        { \ior_map_break: }
                        { \__kulemt_cfg_start_read_section:nn {##1} {#2} }
```

```
                    }
                }
            }
            { \__kulemt_cfg_read_key:w ##1 == \q_stop }
        }
      \__kulemt_cfg_finish_read_section:
      \ior_close:N \g_tmpa_ior
    }
  \msg_new:nnn {kulemt} {cfg/file-not-found}
    { The~ configuration~ file~ '#1'~ is~ missing.}
```

(*End of definition for* `\__kulemt_cfg_read_file:nn`.)

### 3.3.3 Reading a section

`\__kulemt_cfg_start_read_section:nn`
`\__kulemt_cfg_start_read_section_aux:w`

Start reading a section from ⟨*line*⟩ (`#1`) for ⟨*master abbreviation*⟩ (`#2`). To allow reading the configuration file multiple times for different masters, a redefinition of a section does only results in an error if the entire configuration file is read (`#2` is empty).

```
\cs_new_protected:Nn \__kulemt_cfg_start_read_section:nn
  {
    \exp_last_unbraced:Nf \__kulemt_cfg_start_read_section_aux:w
      { \str_tail_ignore_spaces:n {#1} } ] \q_stop
    \tl_if_empty:nT {#2}
      {
        \seq_if_in:NVT \l_kulemt_masters_seq
          \l__kulemt_cfg_current_section_name_str
          {
            \msg_error:nne {kulemt} {cfg/section-redefined}
              { \l__kulemt_cfg_current_section_name_str }
          }
      }
    \prop_clear:N \l__kulemt_cfg_current_section_prop
    \str_clear:N \l__kulemt_cfg_current_key_str
    \str_clear:N \l__kulemt_cfg_current_value_str
    \seq_clear:N \l__kulemt_cfg_current_options_seq
  }
\cs_new_protected:Npn \__kulemt_cfg_start_read_section_aux:w #1] #2 \q_stop
  {
    \tl_if_empty:nT {#2}
      { \msg_warning:nnn {kulemt} {cfg/section-name-error} {#1} }
    \str_set:Nn \l__kulemt_cfg_current_section_name_str {#1}
  }
\msg_new:nnnn {kulemt} {cfg/section-redefined}
  { Section~ '#1'~ defined~ more~ than~ once~ in~ the~ configuration~ file.}
  { The~ previous~ definition~ will~ be~ ignored. }
\msg_new:nnnn {kulemt} {cfg/section-name-error}
  { '[#1'~ misses~ a~ trailing~ ']'~ in~ the~ configuration~ file.}
  { An~ extra~ ']'~ is~ assumed. }
```

(*End of definition for* `\__kulemt_cfg_start_read_section:nn` *and* `\__kulemt_cfg_start_read_section_aux:w`.)

`\__kulemt_cfg_finish_read_section:`

Finish up reading a section. A `[defaults]` section is merged with existing ones. Any other section describes a master and it replaces an existing section with the same name.

```
\cs_new_protected:Nn \__kulemt_cfg_finish_read_section:
  {
    \__kulemt_cfg_finish_read_key:
    \str_if_eq:VnTF \l__kulemt_cfg_current_section_name_str {defaults}
      {
        \prop_map_tokens:Nn \l__kulemt_cfg_current_section_prop
          { \prop_put:Nnn \l__kulemt_cfg_section_defaults_prop }
      }
      {
        \prop_clear:N \l_tmpa_prop
        \prop_put:NnV \l_tmpa_prop {} \l__kulemt_cfg_current_options_seq
        \prop_put:NnV \l__kulemt_cfg_current_section_prop {options} \l_tmpa_prop
        \str_if_empty:NF \l__kulemt_cfg_current_section_name_str
          { \seq_put_right:NV \l_kulemt_masters_seq
              \l__kulemt_cfg_current_section_name_str }
        \prop_set_eq:cN
          { l__kulemt_cfg_section_ \l__kulemt_cfg_current_section_name_str _prop }
          \l__kulemt_cfg_current_section_prop
        \prop_clear:c
          { l__kulemt_cfg_master_ \l__kulemt_cfg_current_section_name_str _prop }
      }
  }
```

(*End of definition for* `\__kulemt_cfg_finish_read_section:`.)

### 3.3.4 Reading a key-value pair

`\__kulemt_cfg_read_key:w`
`\__kulemt_cfg_read_key_aux:w`

Usage: `\__kulemt_cfg_read_key:w` ⟨*line*⟩ `==\q_stop`

The function `\__kulemt_cfg_read_key:w` splits a ⟨*line*⟩ in a key and a value, based on the first '`=`' in the ⟨*line*⟩. The value may contain '`=`' characters. In that case, `#3` contains more than a '`=`'. If no '`=`' is present in the ⟨*line* ⟩(when `#3` is empty), the ⟨*line*⟩ is added to the previous value. The auxiliary function `\__kulemt_cfg_read_key_aux:w` is used to remove the `==` before the `\q_stop`.

```
\cs_new_protected:Npn \__kulemt_cfg_read_key:w #1 = #2 = #3 \q_stop
  {
    \str_case:nnF {#3}
      {
        {}  {
              \str_put_right:Ne \l__kulemt_cfg_current_value_str
                { \c_space_tl \tl_trim_spaces:n {#1} }
            }
        {=} {
              \__kulemt_cfg_finish_read_key:
              \str_set:Ne \l__kulemt_cfg_current_key_str
                { \tl_trim_spaces:n {#1} }
              \str_set:Ne \l__kulemt_cfg_current_value_str
                { \tl_trim_spaces:n {#2} }
            }
      }
      {
        \__kulemt_cfg_finish_read_key:
        \str_set:Ne \l__kulemt_cfg_current_key_str { \tl_trim_spaces:n {#1} }
        \str_set:Ne \l__kulemt_cfg_current_value_str
          { \tl_trim_spaces:o { \__kulemt_cfg_read_key_aux:w #2=#3\q_stop } }
```

```
      }
  }
\cs_new:Npn \__kulemt_cfg_read_key_aux:w #1 ==\q_stop { #1 }
```

(*End of definition for* `\__kulemt_cfg_read_key:w` *and* `\__kulemt_cfg_read_key_aux:w`.)

`\__kulemt_cfg_finish_read_key:` After checking for invalid keys, add the key-value pair to the current section property list. Before the first section, any key is allowed and is stored in the configuration property list. In any other section keys enumerated in `\c__kulemt_cfg_allowed_key_seq` are allowed as well as `option.`⟨*option abbrev*⟩.

```
\cs_new_protected:Nn \__kulemt_cfg_finish_read_key:
  {
    \str_if_empty:NF \l__kulemt_cfg_current_key_str
      {
        \str_if_empty:NTF \l__kulemt_cfg_current_section_name_str
          {
            \prop_put:NVV \l_kulemt_cfg_prop \l__kulemt_cfg_current_key_str
              \l__kulemt_cfg_current_value_str
          }
          {
            \seq_set_split:NnV \l_tmpa_seq {.} \l__kulemt_cfg_current_key_str
            \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str
            \str_set:Ne \l_tmpb_str { \seq_use:Nn \l_tmpa_seq {.} }
            \str_if_eq:VnTF \l_tmpb_str {from}
              {
                \__kulemt_cfg_get_section_item_from:VVN
                  \l__kulemt_cfg_current_value_str \l_tmpa_str \l_tmpa_prop
              }
              {
                \seq_if_in:NVTF \c__kulemt_cfg_allowed_key_seq
                  \l__kulemt_cfg_current_key_str
                  {
                    \seq_set_eq:Nc \l_tmpa_seq
                      {c__kulemt_cfg_ \l_tmpa_str _values_seq}
                    \seq_if_exist:NT \l_tmpa_seq
                      {
                        \seq_if_in:NVF \l_tmpa_seq
                          \l__kulemt_cfg_current_value_str
                          {
                            \msg_error:nneee {kulemt} {cfg/unknown-value}
                              { \l__kulemt_cfg_current_value_str }
                              { \l__kulemt_cfg_current_key_str }
                              { ' \seq_use:Nnnn \l_tmpa_seq
                                { '~ and~ ' } { ',~ ' } { ',~ and~ ' } ' }
                            \tl_set:Ne \l__kulemt_cfg_current_value_str
                              { \seq_item:Nn \l_tmpa_seq {1} }
                          }
                      }
                    \bool_set_true:N \l_tmpa_bool
                  }
                  {
                    \str_if_eq:VnTF \l_tmpa_str {option}
                      {
                        \tl_if_empty:NF \l_tmpb_str
```

```
                        {
                          \seq_put_right:NV
                            \l__kulemt_cfg_current_options_seq \l_tmpb_str
                        }
                      \bool_set_true:N \l_tmpa_bool
                    }
                    { \bool_set_false:N \l_tmpa_bool }
                }
              \bool_if:NTF \l_tmpa_bool
                {
                  \prop_get:NVNF
                    \l__kulemt_cfg_current_section_prop
                    \l_tmpa_str
                    \l_tmpa_prop
                    { \prop_clear:N \l_tmpa_prop }
                  \prop_put:NVV \l_tmpa_prop \l_tmpb_str
                    \l__kulemt_cfg_current_value_str
                }
                {
                  \msg_warning:nnee {kulemt} {cfg/invalid-key}
                    { \l__kulemt_cfg_current_key_str }
                    { \l__kulemt_cfg_current_section_name_str }
                  \prop_set_eq:NN \l_tmpa_prop \q_no_value
                }
            }
          \quark_if_no_value:NF \l_tmpa_prop
            {
              \prop_put:NVV \l__kulemt_cfg_current_section_prop \l_tmpa_str
                \l_tmpa_prop
            }
        }
    }
  }
\msg_new:nnnn {kulemt} {cfg/invalid-key}
  { Invalid~ key~ '#1'~ in~ section~ '#2'. }
  { The~ invalid~ key~ is~ ignored. }
\msg_new:nnnn {kulemt} {cfg/unknown-value}
  { '#1'~ is~ not~ a~ valid~ value~ for~ key~ '#2'.}
  { Valid~ choices~ for~ '#2'~ are:~ #3. }
```

(*End of definition for* `\__kulemt_cfg_finish_read_key:`.)

`\__kulemt_cfg_get_section_item_from:nnN`
`\__kulemt_cfg_get_section_item_from:VVN`

Return the value of ⟨*key*⟩ (`#2`) from the section named ⟨*section*⟩ (`#1`) in the ⟨*variable*⟩ (`#3`), normally a property list variable. Return `\q_no_value` if the ⟨*key*⟩ is not present in ⟨*section*⟩.

Note: the ⟨*section*⟩ must be located before the current section in the configuration file.

```
\cs_new_protected:Nn \__kulemt_cfg_get_section_item_from:nnN
  {
    \seq_if_in:NeTF \l_kulemt_masters_seq { \tl_to_str:n {#1} }
      { \prop_get:cnN { l__kulemt_cfg_section_ #1 _prop } {#2} #3 }
      {
        \msg_error:nneee {kulemt} {cfg/unknown-from}
          { \l__kulemt_cfg_current_key_str }
          { \l__kulemt_cfg_current_section_name_str } { #1 }
```

```
        \prop_set_eq:NN #3 \q_no_value
      }
  }
\cs_generate_variant:Nn \__kulemt_cfg_get_section_item_from:nnN { VV }
\msg_new:nnn {kulemt} {cfg/unknown-from}
  { The~ configuration~ key~ '#1'~ in~ section~ '#2'~ refers~ to~ '#3'.~
    But~ this~ section~ has~ not~ been~ defined~ before. }
```

(*End of definition for* `\__kulemt_cfg_get_section_item_from:nnN`.)

## 3.4 Getting information of a master

`\kulemt_set_master:n`
`\kulemt_set_master:V`

Combine the default settings (from section `[defaults]`) and the settings for a master ⟨*master abbreviation*⟩ and return the result in `\l_kulemt_master_prop`. If needed the configure file is read. If this file does not contain a section named ⟨*master abbreviation*⟩, a fatal error is issued by `\kulemt_read_config_file:n`.

```
\cs_new_protected:Nn \kulemt_set_master:n
  {
    \seq_if_in:NeF \l_kulemt_masters_seq { \tl_to_str:n {#1} }
      { \kulemt_read_config_file:n {#1} }
    \bool_lazy_or:nnT
      {
        ! \tl_if_eq_p:Nc \l_kulemt_master_prop
          { l__kulemt_cfg_master_ #1 _prop }
      }
      { \prop_if_empty_p:N \l_kulemt_master_prop }
      {
        \prop_set_eq:Nc \l_kulemt_master_prop { l__kulemt_cfg_master_ #1 _prop }
        \prop_if_empty:NT \l_kulemt_master_prop
          {
            \prop_set_eq:NN \l_tmpa_prop \l__kulemt_cfg_section_defaults_prop
            \prop_map_tokens:cn { l__kulemt_cfg_section_ #1 _prop }
              { \prop_put:Nnn \l_tmpa_prop }
            \prop_map_inline:Nn \l_tmpa_prop
              {
                \tl_set:Nn \l_tmpb_prop {##2}
                \prop_map_inline:Nn \l_tmpb_prop
                  {
                    \prop_put:Nen \l_kulemt_master_prop
                      { ##1 \tl_if_empty:nF {####1} { . ####1  } } { ####2 }
                  }
              }
            \prop_put:Nnn \l_kulemt_master_prop {abbreviation} {#1}
            \prop_set_eq:cN { l__kulemt_cfg_master_ #1 _prop }
              \l_kulemt_master_prop
          }
      }
  }
\cs_generate_variant:Nn \kulemt_set_master:n { V }
```

(*End of definition for* `\kulemt_set_master:n`. *This function is documented on page 12.*)

`\kulemt_master_get_item:nN`

Return the value of ⟨*key*⟩ ('`#1.`⟨*current language*⟩' or '`#1`') of the current master into the ⟨*var*⟩ (`#2`). If ⟨*current language*⟩ is `british`, `english` is also tried. The ⟨*var*⟩ is

set to \q_no_value if the item is not found for the current master.
The property list \l_kulemt_master_prop holds the information of the current master.

```
\cs_new_protected:Nn \kulemt_master_get_item:nN
  {
    \prop_if_empty:NT \l_kulemt_master_prop
      { \msg_fatal:nnn {kulemt} {cfg/master-not-set} {#1} }
    \exp_args:NNe \prop_get:NnNF \l_kulemt_master_prop {#1 . \languagename} #2
      {
        \str_if_eq:VnTF \languagename {british}
          { \prop_get:NnNF \l_kulemt_master_prop {#1 .english} #2 }
          { \use:n }
            { \prop_get:NnN \l_kulemt_master_prop {#1} #2 }
      }
    \quark_if_no_value:NF #2 { \__kulemt_cfg_str_with_utf:nN {#1} #2 }
  }
\msg_new:nnn {kulemt} {cfg/master-not-set}
  { A~ master~ must~ be~ selected~ before~ getting~ information~ about~ '#1'.}
```

(*End of definition for* \kulemt_master_get_item:nN. *This function is documented on page* 12.)

\kulemt_master_get_item_or_fallback:nnN

If the item #1 is not found for the current master, #2 is used as fallback. If #2 is empty, the function __kulemt_cfg_master_get_item_fallback:nN provides a fallback value.

```
\cs_new_protected:Nn \kulemt_master_get_item_or_fallback:nnN
  {
    \kulemt_master_get_item:nN {#1} #3
    \quark_if_no_value:NT #3
      {
        \tl_if_empty:nTF {#2}
          { \__kulemt_cfg_master_get_item_fallback:nN {#1} #3 }
          { \tl_set:Nn #3 {#2} }
      }
  }
```

(*End of definition for* \kulemt_master_get_item_or_fallback:nnN. *This function is documented on page* 12.)

\kulemt_master_get_required_item:nN

If the item is not found for the current master, an error is issued. If you continue after the error, the function __kulemt_cfg_master_get_item_fallback:nN provides a fallback value.

```
\cs_new_protected:Nn \kulemt_master_get_required_item:nN
  {
    \kulemt_master_get_item:nN {#1} #2
    \quark_if_no_value:NT #2
      {
        \__kulemt_cfg_master_get_item_fallback:nN {#1} \l_tmpa_tl
        \prop_get:NnN \l_kulemt_master_prop {abbreviation} \l_tmpb_tl
        \msg_error:nneee {kulemt} {cfg/missing-key}
          { \l_tmpb_tl } {#1} { \l_tmpa_tl }
        \prop_put:NnV \l_kulemt_master_prop {#1} \l_tmpa_tl
        \prop_set_eq:cN { l__kulemt_cfg_master_ \l_tmpb_tl _prop }
          \l_kulemt_master_prop
      }
  }
\msg_new:nnnn {kulemt} {cfg/missing-key}
```

```
  { The~ required~ key~ '#2'~ is~ missing~ for~ master~ '#1'.}
  { If~ you~ continue,~ the~ value~ '#3'~ will~ be~ used. }
```

(*End of definition for* `\kulemt_master_get_required_item:nN`. *This function is documented on page* 12.)

`\__kulemt_cfg_master_get_item_fallback:nN` This function returns the fallback value for ⟨*key*⟩ (`#1`) in ⟨*tl var*⟩ `#2`.

```
\cs_new_protected:Nn \__kulemt_cfg_master_get_item_fallback:nN
  {
    \seq_set_eq:Nc \l_tmpa_seq {c__kulemt_cfg_ #1 _values_seq}
    \tl_set:Ne #2 { \seq_if_exist:NTF \l_tmpa_seq
                      { \seq_item:Nn \l_tmpa_seq {1} }
                      { ???? } }
  }
```

(*End of definition for* `\__kulemt_cfg_master_get_item_fallback:nN`.)

`\kulemt_master_get_faculty_name:N` Returns the faculty name in `#1` or clear it.

```
\cs_new_protected:Nn \kulemt_master_get_faculty_name:N
  {
    \kulemt_master_get_item:nN {faculty} #1
    \bool_lazy_or:nnT
      { \quark_if_no_value_p:N #1 }
      { \str_if_eq_p:Vn #1 {multi} }
      { \tl_clear:N #1 }
    }
```

(*End of definition for* `\kulemt_master_get_faculty_name:N`. *This function is documented on page* 13.)

`\kulemt_master_obsolete_item:nTF` An item is considered obsolete if it contains a dot.

```
\prg_new_protected_conditional:Nnn \kulemt_master_obsolete_item:n { T, F, TF }
  { \tl_if_in:nnTF {#1} {.} { \prg_return_true: } { \prg_return_false: } }
```

(*End of definition for* `\kulemt_master_obsolete_item:nTF`. *This function is documented on page* 13.)

## 3.5 Typesetting configuration data

`\kulemt_titlecase_first:n`
`\kulemt_titlecase_first:V`

This helper function converts the first character of `#1` to uppercase, taking into account the current language, as stored in `\languagename`.
Unfortunately `\text_titlecase_first:nn` exists only since 2020. So, for older kernels we approximate it with `\MakeUppercase`, but this is unaware of the current language.

```
\cs_if_exist:NTF \text_titlecase_first:nn
  {
    \cs_new_protected:Nn \kulemt_titlecase_first:n
      { \exp_args:NV \text_titlecase_first:nn \languagename {#1} }
  }
  {
    \cs_new_protected:Nn \kulemt_titlecase_first:n
      { \MakeUppercase #1 }
  }
\cs_generate_variant:Nn \kulemt_titlecase_first:n { V }
```

(*End of definition for* `\kulemt_titlecase_first:n`. *This function is documented on page* 13.)

\kulemt_cfg_print_text:n
\kulemt_cfg_print_text_ucfirst:n
__kulemt_cfg_get_text:n

The function `\kulemt_cfg_print_text:n` prints the configuration data '`text.#1`' for the current language, which is stored in `\languagename` by babel. If it is not defined the language `english` is tried after issuing an error.
The function `\kulemt_cfg_print_text_ucfirst:n` also makes the first character uppercase.
Both functions use the function `\__kulemt_cfg_get_text:n` to get the configuration data.

```
\cs_new_protected:Nn \kulemt_cfg_print_text:n
  {
    \__kulemt_cfg_get_text:n {#1}
    \l__kulemt_cfg_tmp_tl
  }
\cs_new_protected:Nn \kulemt_cfg_print_text_ucfirst:n
  {
    \__kulemt_cfg_get_text:n {#1}
    \kulemt_titlecase_first:V \l__kulemt_cfg_tmp_tl
  }
\cs_new_protected:Nn \__kulemt_cfg_get_text:n
  {
     \exp_args:NNe
      \prop_get:NnNF \l_kulemt_cfg_prop { text. #1 .\languagename}
        \l__kulemt_cfg_tmp_tl
        {
          \str_if_eq:VnT \languagename {british}
            {
              \prop_get:NnN \l_kulemt_cfg_prop {text.#1.english}
                \l__kulemt_cfg_tmp_tl
            }
        }
    \quark_if_no_value:NT \l__kulemt_cfg_tmp_tl
      {
        \msg_error:nnee {kulemt} {cfg/text-not-set} {#1} { \languagename }
        \prop_get:NnN \l_kulemt_cfg_prop { text.#1.english}
          \l__kulemt_cfg_tmp_tl
      }
    \quark_if_no_value:NTF \l__kulemt_cfg_tmp_tl
      { \tl_set:Ne \l__kulemt_cfg_tmp_tl {???} }
      { \__kulemt_cfg_str_with_utf:N \l__kulemt_cfg_tmp_tl }
  }
\msg_new:nnnn {kulemt} {cfg/text-not-set}
  { Configuration~ data~ 'text.#1'~ is~ undefined~ for~ language~ '#2'. }
  { Correct~ the~ configuration~ file.\\
    I~ shall~ use~ the~ English~ text~ for~ now~ if~ available. }
```

(*End of definition for* `\kulemt_cfg_print_text:n`, `\kulemt_cfg_print_text_ucfirst:n`, *and* `\__kulemt_cfg_get_text:n`. *These functions are documented on page 13.*)

\kulemt_cfg_print_text_from_opt:n
\kulemt_cfg_print_text_from_opt_ucfirst:n
__kulemt_cfg_get_text_from_opt:n

These functions add `.plural` to the argument `#1` if the option holds multiple items. Then they call the corresponding function without `_from_opt` in its name.

```
\cs_new_protected:Nn \kulemt_cfg_print_text_from_opt:n
  {
    \__kulemt_cfg_get_text_from_opt:n {#1}
    \l__kulemt_cfg_tmp_tl
  }
```

```
\cs_new_protected:Nn \kulemt_cfg_print_text_from_opt_ucfirst:n
  {
    \__kulemt_cfg_get_text_from_opt:n {#1}
    \kulemt_titlecase_first:V \l__kulemt_cfg_tmp_tl
  }
\cs_new_protected:Nn \__kulemt_cfg_get_text_from_opt:n
  {
    \exp_args:Ne \__kulemt_cfg_get_text:n
      {
        #1
        \int_compare:nNnF { \seq_count:c { l_kulemt_opt_ #1 _seq } } = {1}
          { .plural }
      }
  }
```

(*End of definition for* `\kulemt_cfg_print_text_from_opt:n`, `\kulemt_cfg_print_text_from_opt_ucfirst:n`, *and* `\__kulemt_cfg_get_text_from_opt:n`. *These functions are documented on page* 13.)

`\kulemt_master_print_required_item:n`

Works like `\kulemt_master_get_required_item:nN` but prints the value instead of storing it in a variable.

```
\cs_new_protected:Nn \kulemt_master_print_required_item:n
  {
    \kulemt_master_get_required_item:nN {#1} \l__kulemt_cfg_tmp_tl
    \l__kulemt_cfg_tmp_tl
  }
```

(*End of definition for* `\kulemt_master_print_required_item:n`. *This function is documented on page* 13.)

⟨/class⟩

File IV

# kulemt-opt.dtx
# Option processing

## 1 Setting options

Options can be set as document class options or with the preamble command `\setup`. Some options can only be used as document class options because they are used directly in the kulemt class file. On the other hand, some key-value pairs cannot be used as class options because their value contains commands, braces, brackets, commas or spaces. In practice these are the options which provide information for the title page ('`assessor`', '`assistant`', '`author`', '`promoter`', '`subtitle`', and '`title`').

Most of the options are optional, except for '`master`' (which determines the master) and most of the information for the title page ('`assessor`', '`assistant`', '`author`', '`promoter`', and '`title`'). However, when the option '`article`' is used, all of these options are optional too.

`\setup` `{`⟨*key-value list*⟩`}`

Contrary to document class options, the values in `\setup` have no restrictions if they are grouped (i.e., enclosed in braces). The `\setup` command can be used multiple times but only in the document preamble.

Since no language settings are active in the preamble, language specific active characters (e.g. '`"`' as defined by babel for Dutch) cannot be used.

`\kulemt_process_class_options:`

The function `\kulemt_process_class_options:` can be used to process the document class options.

### 1.1 Using the option information

`\l_kulemt_opt_article_bool`

The document class option '`article`' switches from the thesis layout to an article layout, as provided by the memoir class. The variable `\l_kulemt_opt_article_bool` remembers the switch.

In this layout, no front pages are generated, all other options are optional and an additional option '`twocolumn`' is available. The additional option can only be used after the '`article`' option.

`\l_kulemt_memoir_options_seq`

Many document class options are actually memoir options. They are remembered in the sequence `\l_kulemt_memoir_options_seq`. These are all options without values: '`10pt`', '`11pt`', '`draft`', '`fleqn`', '`oldfontcommands`', '`openany`', '`openleft`', and '`openright`'.

`\l_kulemt_opt_lrequal_bool`

The options '`oneside`' and '`twoside`' are also memoir options. The new option '`twosidelrequal`' corresponds to the `twoside` option but with equal inner and outer margins. They all set the variable `\l_kulemt_opt_lrequal_bool`.

`\l_kulemt_opt_ptsize_int`

The type size is not only passed to memoir, but the number is also stored in the integer `\l_kulemt_opt_ptsize_int`.

`\l_kulemt_language_tl`
`\l_kulemt_master_language_tl`
`\l_kulemt_babel_seq`

Three document class options are available to set the language options. The main text language is stored in `\l_kulemt_language_tl` and set either by the option '`dutch`' or '`english`'. Additionally the language '`british`' is available, since this is the English variant recommended by the KU Leuven. Initially `\l_kulemt_language_tl` is set to the language of the master program, which is stored in `\l_kulemt_master_language_tl`. If no '`master`' option is given, `\l_kulemt_language_tl` is initialized to `english`, which is the default babel language.

Extra babel languages can be added with option '`extralanguage`' but no dialects of '`english`' (namely '`american`', '`australian`', '`canadian`', '`newzealand`') or '`dutch`' (namely '`afrikaans`') are allowed. This option can be used multiple times.

All babel options are stored in the sequence `\l_kulemt_babel_seq`.

`\l_kulemt_opt_bind_dim`
`\l_kulemt_opt_cfgfile_tl`
`\l_kulemt_opt_master_tl`

The remaining document class options store their information in an appropriate variable. The dimension `\l_kulemt_opt_bind_dim` holds the binding loss length (option '`bind`'). The variable `\l_kulemt_opt_cfgfile_tl` holds the name of the configuration file (option '`cfgfile`') and `\l_kulemt_opt_master_tl` holds the master option abbreviation (option '`master`').

`\l_kulemt_opt_masteroption_seq`

The option '`masteroption`' selects the master option. If students of different options work on one thesis, the option can be used multiple times. The value of an option can either be an abbreviation or a full option name. Only if it is an abbreviation, this option can be used as a document class option. The sequence `\l_kulemt_opt_masteroption_seq` holds the master options with all abbreviations replaced by their full name.

\l_kulemt_include_coverpage_bool
\l_kulemt_include_frontpages_bool
\l_kulemt_include_text_bool

These boolean variables determine which pages are generated: the cover page, the front pages and the text. By default, the cover page is not generated, which is compatible with version 1. In the article layout, the front pages and the cover page are not generated.

The options (without values) '`coverpageonly`' and '`frontpagesonly`' can be used to generate only the cover page or only the front pages (the title page and the copyright page). They set the boolean variables mentioned above. These options can be used as document class options as well as within the `\setup` argument.
In the article layout these options have no effect.

\l_kulemt_opt_acyear_int
\l_kulemt_opt_subtitle_tl
\l_kulemt_opt_title_tl

The options '`acyear`', '`subtitle`', and '`title`' store their value in the corresponding variables.

`\title` For compatibility with other LaTeX packages, the title is also stored with the standard LaTeX command `\title`.

\l_kulemt_opt_assessor_seq
\l_kulemt_opt_assistant_seq
\l_kulemt_opt_author_seq
\l_kulemt_opt_promoter_seq

The options '`assessor`', '`assistant`', '`author`', and '`promoter`' store their value in a sequence. These options can be used multiple times or values can be combined separated with '`\and`'. The options '`assessor`' and '`assistant`' also allow an empty value to suppress the printing of their information.

`\author` For compatibility with other LaTeX packages, the authors are also stored with the standard LaTeX command `\author`.

## 1.2 Removed version 1 options

### 1.2.1 Filing card options

Since a filing card is no longer provided, all options providing additional information for the filing card are removed: '`translatedtitle`', '`shortabstract`', '`udc`', '`keywords`', and '`articletitle`'. Additionally the option '`filingcard`' is removed.

### 1.2.2 Typeblock layout options

The obsolete option '`bindcover`' has been removed.

### 1.2.3 Text encoding option

Since 2018 the default LaTeX text encoding is UTF-8, which supports all characters. Furthermore Unicode engines only support UTF-8. So, the option '`inputenc`' becomes much less relevant. If people need it, they can use

```
\usepackage[⟨encoding name⟩]{inputenc}
```

at the beginning of the preamble.

### 1.2.4 Font selection

Depending on the use of traditional fonts or OpenType fonts and depending on the engine, font selection varies a lot. So, it doesn't seem to be a good idea to make too much assumptions. It seems better to let users put their own font packages in the preamble.

Latin Modern is used as default font family since this is a pretty complete family which works well with most popular font encodings. The default font encoding is either `TU` for LuaTeX and XeTeX, `T1` (instead of `OT1`) otherwise.

# 2 Implementation

```
⟨*class⟩
⟨@@=kulemt_opt⟩
```

Some x-variants are since October 2023 version no longer available. We generate here the e-type variants for functions which don't exist yet and are used in this file.

```
\cs_generate_variant:Nn \msg_error:nnn { nne }
\cs_generate_variant:Nn \msg_fatal:nnn { nne }
\cs_generate_variant:Nn \msg_fatal:nnnn { nnee }
```

## 2.1 Setting the font defaults

By default, LaTeX sets the font encoding to `TU` for Unicode engines and to `OT1` otherwise. In the latter case we change it to `T1`. The LaTeX core uses `\renewcommand`, so we use `\cs_set:Npn` instead of `\tl_set:Nn` to have the same effect.

```
\str_if_eq:VnT \encodingdefault {OT1}
  {
    \cs_set:Npn \encodingdefault {T1}
    \fontencoding { \encodingdefault }
  }
```

The Latin Modern family is the default font family for the `TU` font encoding, so we prefer it also for other encodings over the default font family Computer Modern.

```
\str_if_eq:VnT \rmdefault {cmr}
  {
    \cs_set:Npn \rmdefault {lmr}
    \cs_set:Npn \sfdefault {lmss}
    \cs_set:Npn \ttdefault {lmtt}
    \fontfamily { \rmdefault }
  }
```

## 2.2 Keys which can only be used as class options

The following keys can only be used as class options because they are either used directly in the kulemt class file or they must be passed as class options to memoir.

`\l__kulemt_opt_allow_class_option_bool` This boolean is set true when we are processing document class options. We also set it true while loading this class to make initialization work.

```
\bool_new:N \l__kulemt_opt_allow_class_option_bool
\bool_set_true:N \l__kulemt_opt_allow_class_option_bool
```

(*End of definition for* `\l__kulemt_opt_allow_class_option_bool`.)

`\__kulemt_opt_class_option:n` Execute `#1` only when processing a document class option. Otherwise raise an error.

```
\cs_new_protected:Nn \__kulemt_opt_class_option:n
  {
    \bool_if:NTF \l__kulemt_opt_allow_class_option_bool
      { #1 }
      { \msg_error:nne {kulemt} {opt/not-class} { \kulemt_keys_key: } }
  }
\msg_new:nnnn {kulemt} {opt/not-class}
  { Option~ '#1'~ can~ only~ be~ used~ as~ a~ class~ option. }
  { This~ setup~ option~ will~ be~ ignored. }
```

(*End of definition for* `\__kulemt_opt_class_option:n`.)

`\l_kulemt_memoir_options_seq` This variable stores the list of options to pass to the memoir class.

```
\seq_new:N \l_kulemt_memoir_options_seq
```

(*End of definition for* `\l_kulemt_memoir_options_seq`. *This variable is documented on page 29.*)

`\__kulemt_opt_memoir_option:n`
`\__kulemt_opt_memoir_option:` The function `\__kulemt_opt_memoir_option:n` stores `#1` as a memoir option. For the more common case of storing the current option key (not the value), `\__kulemt_opt_memoir_option:` can be used.
Since the memoir options are used in the kulemt class, these functions can only be used in the definition of document class options or before loading the memoir class.

```
\cs_new_protected:Nn \__kulemt_opt_memoir_option:n
  {
    \__kulemt_opt_class_option:n
      { \seq_put_right:Nn \l_kulemt_memoir_options_seq {#1} }
  }
\cs_new_protected:Nn \__kulemt_opt_memoir_option:
  { \exp_args:Ne \__kulemt_opt_memoir_option:n { \kulemt_keys_key: } }
```

(*End of definition for* `\__kulemt_opt_memoir_option:n` *and* `\__kulemt_opt_memoir_option:`.)

### 2.2.1 Selecting an article layout

`article` (*option*) The option '`article`' select an article layout instead of the thesis layout.

`\l_kulemt_opt_article_bool` The variable `\l_kulemt_article_bool` remembers the request for an article layout.

```
\bool_new:N \l_kulemt_opt_article_bool
\keys_define:nn {kulemt}
  {
    article .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \bool_set_true:N \l_kulemt_opt_article_bool
            \__kulemt_opt_memoir_option:
          }
      } ,
    article .value_forbidden:n = true
  }
```

(*End of definition for* `\l_kulemt_opt_article_bool`. *This variable is documented on page 28.*)

`twocolumn` (*option*) The option '`twocolumn`' is only available in the article layout. Furthermore it can only be used after the '`article`' option. Otherwise it generates an error.

```
\keys_define:nn {kulemt}
  {
    twocolumn .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \bool_if:NTF \l_kulemt_opt_article_bool
              { \__kulemt_opt_memoir_option: }
              { \msg_error:nne {kulemt} {opt/not-art} { \kulemt_keys_key: } }
          }
      } ,
    twocolumn .value_forbidden:n = true
  }
\msg_new:nnnn {kulemt} {opt/not-art}
  { Option~ '#1'~ can~ only~ be~ used~ after~ the~ class~ option~ 'article'. }
  { This~ setup~ option~ will~ be~ ignored. }
```

### 2.2.2 Selecting the master's program

`cfgfile` (*option*) The option '`cfgfile`' defines the name of the configuration file. By default it is set to "`kulemt.ini`".

`\l_kulemt_opt_cfgfile_tl` This variable holds the name of the configuration file.

```
\tl_new:N \l_kulemt_opt_cfgfile_tl
\keys_define:nn {kulemt}
  {
    cfgfile .code:n =
      {
        \__kulemt_opt_class_option:n
          { \tl_set:Nn \l_kulemt_opt_cfgfile_tl {#1} }

      } ,
    cfgfile .value_required:n = true,
    cfgfile .initial:n = { kulemt.ini }
  }
```

(*End of definition for* `\l_kulemt_opt_cfgfile_tl`. *This variable is documented on page 29.*)

`master` (*option*) The value of the option '`master`' is the abbreviation of the master.

`\l_kulemt_opt_master_tl` It is stored in the variable `\l_kulemt_opt_master_tl`.

```
\tl_new:N \l_kulemt_opt_master_tl
\keys_define:nn {kulemt}
  {
    master .code:n =
      {
        \__kulemt_opt_class_option:n
          { \tl_set:Nn \l_kulemt_opt_master_tl {#1} }
      } ,
    master .value_required:n = true
  }
```

(*End of definition for* `\l_kulemt_opt_master_tl`. *This variable is documented on page 29.*)

### 2.2.3 Type size

10pt (*option*)
11pt (*option*)

The type size option is either '10pt' or '11pt' (the default value). It will be passed to memoir but it will also be used to determine the page layout. These options are mutually exclusive.

\l_kulemt_opt_ptsize_int

This variable stores the number of the type size.

```
\int_new:N \l_kulemt_opt_ptsize_int
\int_set:Nn \l_kulemt_opt_ptsize_int {11}
\keys_define:nn {kulemt}
  {
    10pt .code:n =
      {
        \__kulemt_opt_class_option:n
          { \int_set:Nn \l_kulemt_opt_ptsize_int {10} }
      } ,
    10pt .value_forbidden:n = true,
    11pt .code:n =
      {
        \__kulemt_opt_class_option:n
          { \int_set:Nn \l_kulemt_opt_ptsize_int {11} }
      } ,
    11pt .value_forbidden:n = true
  }
```

(*End of definition for* `\l_kulemt_opt_ptsize_int`. *This variable is documented on page* 29.)

### 2.2.4 Printing options

A4 paper is used. No other options to change the paper size are available.

```
\__kulemt_opt_memoir_option:n { a4paper }
```

draft (*option*)
openany (*option*)
openleft (*option*)
openright (*option*)

The following options are passed directly to memoir.

```
\keys_define:nn {kulemt} {
    draft .code:n = { \__kulemt_opt_memoir_option: } ,
    draft .value_forbidden:n = true ,
    openany .code:n = { \__kulemt_opt_memoir_option: } ,
    openany .value_forbidden:n = true,
    openleft .code:n = { \__kulemt_opt_memoir_option: } ,
    openleft .value_forbidden:n = true,
    openright .code:n = { \__kulemt_opt_memoir_option:n {open} } ,
    openright .value_forbidden:n = true
  }
```

\l_kulemt_opt_lrequal_bool

When this boolean is set true, the inner and outer margins must be made equal. By default it is true since version 2.

```
\bool_new:N \l_kulemt_opt_lrequal_bool
\bool_set_true:N \l_kulemt_opt_lrequal_bool
```

(*End of definition for* `\l_kulemt_opt_lrequal_bool`. *This variable is documented on page* 29.)

oneside (*option*) These options set the memoir options `oneside` or `twoside`. They also set the variable
twoside (*option*) `\l_kulemt_opt_lrequal_bool`, false for `twoside`, true otherwise.
twosidelrequal (*option*)

```
\keys_define:nn {kulemt} {
    oneside .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \seq_put_right:Nn \l_kulemt_memoir_options_seq {oneside}
            \bool_set_true:N \l_kulemt_opt_lrequal_bool
          }
      } ,
    oneside .value_forbidden:n = true,
    twoside .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \seq_put_right:Nn \l_kulemt_memoir_options_seq {twoside}
            \bool_set_false:N \l_kulemt_opt_lrequal_bool
          }
      } ,
    twoside .value_forbidden:n = true,
    twosidelrequal .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \seq_put_right:Nn \l_kulemt_memoir_options_seq {twoside}
            \bool_set_true:N \l_kulemt_opt_lrequal_bool
          }
      } ,
    twosidelrequal .value_forbidden:n = true
  }
```

bind (*option*) The option '`bind`' specifies the loss of visible paper due to binding the book.

\l_kulemt_opt_bind_dim This dimension is stored in the variable `\l_kulemt_opt_bind_dim`.

```
\dim_new:N \l_kulemt_opt_bind_dim
\keys_define:nn {kulemt}
  {
    bind .code:n =
      {
        \__kulemt_opt_class_option:n
          { \dim_set:Nn \l_kulemt_opt_bind_dim {#1} }
      } ,
    bind .value_required:n = true
  }
```

(*End of definition for* `\l_kulemt_opt_bind_dim`. *This variable is documented on page* 29.)

### 2.2.5 Language options

\l_kulemt_babel_seq The options of the babel package are collected in the variable `\l_kulemt_babel_seq`.

```
\seq_new:N \l_kulemt_babel_seq
```

(*End of definition for* `\l_kulemt_babel_seq`. *This variable is documented on page* 29.)

`dutch` (*option*)
`english` (*option*)
`british` (*option*)

The options '`dutch`', '`english`', or '`british`' allow you to select the main text language. Since you can have only one main text language, these three options are mutually exclusive.

`\l_kulemt_language_tl`

The main text language is stored in the variable `\l_kulemt_language_tl`. We use a token list instead of a string because babel assumes document category codes when comparing language names: it uses `\ifx` for the comparison.

```
\tl_new:N \l_kulemt_language_tl
```

(*End of definition for* `\l_kulemt_language_tl`. *This variable is documented on page* 29.)

`\__kulemt_opt_store_language:n`

Store the main text language `#1` in `\l_kulemt_language_tl`. Since you can have only one main text language, an error is issued if you try to do this more than once.

```
\cs_new_protected:Nn \__kulemt_opt_store_language:n
  {
    \__kulemt_opt_class_option:n
      {
        \tl_if_empty:NTF \l_kulemt_language_tl
          { \tl_set:Nn \l_kulemt_language_tl {#1} }
          {
            \msg_fatal:nnee {kulemt} {opt/multiple-languages}
              { \l_kulemt_language_tl } {#1}
          }
      }
  }
\msg_new:nnnn {kulemt} {opt/multiple-languages}
  { You~ can~ set~ the~ main~ text~ language~ only~ once.\\
    You~ used~ the~ language~ options~ '#1'~ and~ '#2'. }
  { Remove~ one~ of~ the~ language~ options. }
```

(*End of definition for* `\__kulemt_opt_store_language:n`.)

```
\keys_define:nn {kulemt}
  {
    dutch .code:n =   { \__kulemt_opt_store_language:n {dutch} } ,
    dutch .value_forbidden:n = true,
    english .code:n = { \__kulemt_opt_store_language:n {english} } ,
    english .value_forbidden:n = true,
    british .code:n = { \__kulemt_opt_store_language:n {british} } ,
    british .value_forbidden:n = true
  }
```

`extralanguage` (*option*)

The option '`extralanguage`' adds a language to babel. Its value is added to the list `\l_kulemt_babel_seq`. We assume that the extra language is not one of the languages in `\l_kulemt_babel_seq` or a dialect of one of them, since babel cannot handle this.

```
\keys_define:nn {kulemt}
  {
    extralanguage .code:n =
      {
        \__kulemt_opt_class_option:n
          {
            \tl_if_empty:nF {#1}
              { \seq_put_right:Nn \l_kulemt_babel_seq {#1} }
          }
```

```
      } ,
    extralanguage .value_required:n = true
  }
```

### 2.2.6 Other options

`fleqn` (*option*)
`oldfontcommands` (*option*)

The following options are also passed directly to the memoir class.

```
\keys_define:nn {kulemt}
  {
    fleqn .code:n = { \__kulemt_opt_memoir_option: } ,
    fleqn .value_forbidden:n = true,
    oldfontcommands .code:n = { \__kulemt_opt_memoir_option: } ,
    oldfontcommands .value_forbidden:n = true
  }
```

## 2.3 Keys which can also be used in `\setup`

The following keys can be used multiple times in the preamble, as a document option and in every `\setup`.

`\__kulemt_opt_check_required:nn`

If the required option `#2` of type `#1` (e.g. `tl`) was not used, i.e. its variable is empty, raise a fatal error at the end of the document preamble, but only in the thesis layout.

```
\cs_new_protected:Nn \__kulemt_opt_check_required:nn
  {
    \exp_args:Ne \kulemt_at_end_preamble:n
      {
        \exp_not:N \bool_if:NF \exp_not:N \l_kulemt_opt_article_bool
          {
            \exp_not:c { #1 _if_empty:NT } \exp_not:c { l_kulemt_opt_ #2 _ #1 }
              { \exp_not:N \msg_fatal:nnn {kulemt} {opt/missing} {#2} }
          }
      }
  }
\msg_new:nnn {kulemt} {opt/missing} { A~ required~ option~ '#1'~ is~ missing. }
```

(*End of definition for* `\__kulemt_opt_check_required:nn`.)

### 2.3.1 Setting the master's program option

`masteroption` (*option*)

The value of the option '`masteroption`' defines the master's program option or specialization (aka "optie" or "afstudeerrichting"). The value is a text starting with "`option` ..." (or its Dutch counterpart) or something similar. If the master's program defines options, you can use the option abbreviation as a `masteroption` value. In that case, '`masteroption`' can also be used as a document class option.

If students of different master's program options work on one common master's thesis, the option '`masteroption`' can be used multiple times.

`\l__kulemt_opt_masteroption_seq`

A list of all `masteroption` values is stored in `\l__kulemt_opt_masteroption_seq`. It contains no duplicates.

```
\seq_new:N \l__kulemt_opt_masteroption_seq
\keys_define:nn {kulemt}
  {
    masteroption .code:n =
```

```
      {
        \tl_if_empty:nF {#1}
          {
            \seq_if_in:NnF \l__kulemt_opt_masteroption_seq {#1}
              { \seq_put_right:Nn \l__kulemt_opt_masteroption_seq {#1} }
          }
      } ,
    masteroption .value_required:n = true
  }
```

(*End of definition for* `\l__kulemt_opt_masteroption_seq`.)

`\l_kulemt_opt_masteroption_seq` This variable holds the list of master options with the abbreviations expanded to their full text.

```
\seq_new:N \l_kulemt_opt_masteroption_seq
```

(*End of definition for* `\l_kulemt_opt_masteroption_seq`. *This variable is documented on page* 29.)

`\__kulemt_opt_check_masteroption:` Once we know the master's program, we can check the master option(s) and eventually expand the abbreviations. If no master's program is known, we simply keep the values from '`masteroption`'. Since a master option can be set with `\setup`, we have to check this at the end of the document preamble.

```
\cs_new_protected:Nn \__kulemt_opt_check_masteroption:
  {
    \tl_if_empty:NTF \l_kulemt_opt_master_tl
      {
        \seq_set_eq:NN \l_kulemt_opt_masteroption_seq
          \l__kulemt_opt_masteroption_seq
      }
      {
        \seq_clear:N \l_kulemt_opt_masteroption_seq
        \kulemt_master_get_item_or_fallback:nnN {option} {} \l_tmpa_tl
        \str_if_eq:VnTF \l_tmpa_tl {forbidden}
          {
            \seq_if_empty:NF \l__kulemt_opt_masteroption_seq
              {
                \msg_warning:nn {kulemt} {opt/masteroption-forbidden}
                \seq_clear:N \l__kulemt_opt_masteroption_seq
              }
          }
          {
            \str_if_eq:VnT \l_tmpa_tl {required}
              {
                \seq_if_empty:NT \l__kulemt_opt_masteroption_seq
                  {
                    \msg_fatal:nne {kulemt} {opt/masteroption-missing}
                      { \l_kulemt_opt_master_tl }
                  }
              }
            \seq_map_inline:Nn \l__kulemt_opt_masteroption_seq
              {
                \kulemt_master_get_item:nN { option. ##1  } \l_tmpa_tl
                \quark_if_no_value:NT \l_tmpa_tl
                  {
```

```
                    \msg_info:nnn {kulemt} {opt/masteroption-no-abbrev} {##1}
                    \tl_set:Nn \l_tmpa_tl {##1}
                  }
                \seq_put_right:NV \l_kulemt_opt_masteroption_seq \l_tmpa_tl
              }
          }
      }
  }
\kulemt_at_end_preamble:n { \__kulemt_opt_check_masteroption: }
\msg_new:nnn {kulemt} {opt/masteroption-forbidden}
  {
    The~ option~ 'masteroption'~ is~ ignored~ because~ your~ program~
    disallows~ a~ master's~ program~ option~ on~ front~ pages.
  }
\msg_new:nnn {kulemt} {opt/masteroption-missing}
  {
    For~ master~ '#1'~ you~ must~ specify~ at~ least~ one~
    master's~ program~ option.
  }
\msg_new:nnn {kulemt} {opt/masteroption-no-abbrev}
  {
    The~ master~ option~ '#1'~ is~ not~ a~ known~ abbreviation.\\
    It~ is~ used~ directly~ as~ the~ master~ option~ text.
   }
```

(*End of definition for* `\__kulemt_opt_check_masteroption:`.)

### 2.3.2 Information for the title page

`\__kulemt_opt_seq_add_split:NnnN`

Some of the options can store multiple values, separated by `#3`. This function converts its argument `#2` to a sequence and adds it to the sequence `#1`, ignoring the empty first element of it. If `#4` is ⟨*true*⟩, an empty option value is allowed otherwise not. So if `#4` is ⟨*true*⟩ and `#2` is empty after removing any `#3`, the sequence `#1` is set to a sequence with one empty item. (This is the only way to get an empty item in the sequence.)

```
\cs_new_protected:Nn \__kulemt_opt_seq_add_split:NnnN
  {
    \seq_set_split:Nnn \l_tmpa_seq {#3} {#2}
    \seq_remove_all:Nn \l_tmpa_seq {}
    \seq_if_empty:NTF \l_tmpa_seq
      {
        \bool_if:NT #4
          {
            \seq_clear:N #1
            \seq_put_right:Nn #1 {}
          }
      }
      {
        \seq_if_empty:NTF #1
          { \seq_set_eq:NN #1 \l_tmpa_seq }
          {
            \seq_get:NN #1 \l_tmpa_tl
            \tl_if_empty:NTF \l_tmpa_tl
              { \seq_set_eq:NN #1 \l_tmpa_seq }
              { \seq_concat:NNN #1 #1 \l_tmpa_seq }
```

```
            }
        }
    }
```

(*End of definition for* `\__kulemt_opt_seq_add_split:NnnN`.)

`\c__kulemt_opt_disallow_empty_bool` `\c__kulemt_opt_allow_empty_bool`

To make the code more readable, these constant booleans are defined to be used as `#4` of `\__kulemt_opt_seq_add_split:NnnN`.

```
\bool_const:Nn \c__kulemt_opt_disallow_empty_bool { \c_false_bool }
\bool_const:Nn \c__kulemt_opt_allow_empty_bool    { \c_true_bool  }
```

(*End of definition for* `\c__kulemt_opt_disallow_empty_bool` *and* `\c__kulemt_opt_allow_empty_bool`.)

`title` (*option*)

The option '`title`' defines the title of the thesis.

Since '`title`' is a required option in the thesis layout, a fatal error is issued if the option is missing at the beginning of the document.

`\l_kulemt_opt_title_tl`

This variable holds the title. For compatibility with other LaTeX packages, e.g. hyperref, the title is also stored with the standard LaTeX command `\title`. Since packages may redefine this command, its use is postponed until the end of the preamble.

```
\tl_new:N \l_kulemt_opt_title_tl
\keys_define:nn {kulemt}
  {
    title .code:n =
      {
        \tl_set:Nn \l_kulemt_opt_title_tl {#1}
        \kulemt_at_end_preamble:n { \title{#1} }
      } ,
    title .value_required:n = true
  }
\__kulemt_opt_check_required:nn {tl} {title}
```

(*End of definition for* `\l_kulemt_opt_title_tl`. *This variable is documented on page 30.*)

`subtitle` (*option*)

The option '`subtitle`' defines the subtitle.

`\l_kulemt_opt_subtitle_tl`

The subtitle is stored in the variable `\l_kulemt_opt_subtitle_tl`.

```
\tl_new:N \l_kulemt_opt_subtitle_tl
\keys_define:nn {kulemt}
  {
    subtitle .tl_set:N = \l_kulemt_opt_subtitle_tl ,
    subtitle .value_required:n = true
  }
```

(*End of definition for* `\l_kulemt_opt_subtitle_tl`. *This variable is documented on page 30.*)

`author` (*option*)

The option '`author`' defines one author or multiple authors separated by '`\and`'. This option can also be used multiple times.

`\l_kulemt_opt_author_seq`

The variable `\l_kulemt_opt_author_seq` holds the list of authors.

Since '`author`' is a required option, a fatal error is issued if the option is missing at the beginning of the document. The `\__kulemt_opt_check_required:nn` also guarantees that an empty `\l_kulemt_opt_author_seq` is not possible in the thesis layout.

```
\seq_new:N \l_kulemt_opt_author_seq
\keys_define:nn {kulemt}
```

```
  {
    author .code:n =
      {
        \__kulemt_opt_seq_add_split:NnnN \l_kulemt_opt_author_seq
          {#1} { \and } \c__kulemt_opt_disallow_empty_bool
      } ,
    author .value_required:n = true
  }
\__kulemt_opt_check_required:nn {seq} {author}
```

For compatibility with other LaTeX packages, the authors are also stored with the standard LaTeX command `\author` using '`\and`' to separate the authors. Since packages may redefine the command, its use is postponed until the end of the preamble.

```
\kulemt_at_end_preamble:n
  {
    \seq_if_empty:NF \l_kulemt_opt_author_seq
      {
        \exp_args:Ne \author
          { \seq_use:Nn \l_kulemt_opt_author_seq { \and } }
      }
  }
```

(*End of definition for* `\l_kulemt_opt_author_seq`. *This variable is documented on page* 30.)

`promoter` (*option*)
`promotor` (*option*)

The option '`promoter`' or its alias '`promotor`' defines one promoter or multiple promoters separated by '`\and`'. This option can also be used multiple times. Since this is a required option, a fatal error is issued if the option is missing.

`\l_kulemt_opt_promoter_seq`

The variable `\l_kulemt_opt_promoter_seq` holds the list of promoters.

```
\seq_new:N \l_kulemt_opt_promoter_seq
\keys_define:nn {kulemt}
  {
    promoter .code:n =
      {
        \__kulemt_opt_seq_add_split:NnnN \l_kulemt_opt_promoter_seq
          {#1} { \and } \c__kulemt_opt_disallow_empty_bool
      } ,
    promoter .value_required:n = true ,
    promotor .meta:n = { promoter = {#1} } ,
    promotor .value_required:n = true
  }
\__kulemt_opt_check_required:nn {seq} {promoter}
```

(*End of definition for* `\l_kulemt_opt_promoter_seq`. *This variable is documented on page* 30.)

`assessor` (*option*)

The option '`assessor`' defines one assessor or multiple assessors separated by '`\and`'. This option can also be used multiple times. Since this is a required option, a fatal error is issued if the option is missing. However, an empty value is allowed and suppresses its printing.

`\l_kulemt_opt_assessor_seq`

The variable `\l_kulemt_opt_assessor_seq` holds the list of assessors.

```
\seq_new:N \l_kulemt_opt_assessor_seq
\keys_define:nn {kulemt}
  {
    assessor .code:n =
```

```
      {
        \__kulemt_opt_seq_add_split:NnnN \l_kulemt_opt_assessor_seq
          {#1} { \and } \c__kulemt_opt_allow_empty_bool
      } ,
      assessor .value_required:n = true
  }
\__kulemt_opt_check_required:nn {seq} {assessor}
```

(*End of definition for* `\l_kulemt_opt_assessor_seq`. *This variable is documented on page 30.*)

`assistant` (*option*) The option '`assistant`' defines one assessor or multiple assessors separated by '`\and`'. This option can also be used multiple times. Since this is a required option, a fatal error is issued if the option is missing. However, an empty value is allowed and suppresses its printing.

`\l_kulemt_opt_assistant_seq` The variable `\l_kulemt_opt_assistant_seq` holds the list of assistants.

```
\seq_new:N \l_kulemt_opt_assistant_seq
\keys_define:nn {kulemt}
  {
    assistant .code:n =
      {
        \__kulemt_opt_seq_add_split:NnnN \l_kulemt_opt_assistant_seq
          {#1} { \and } \c__kulemt_opt_allow_empty_bool
      } ,
    assistant .value_required:n = true
  }
\__kulemt_opt_check_required:nn {seq} {assistant}
```

(*End of definition for* `\l_kulemt_opt_assistant_seq`. *This variable is documented on page 30.*)

`acyear` (*option*) The option '`acyear`' sets the starting year of the academic year of the thesis. The value starts with a 4-digit number. For compatibility with a previous version, the other tokens are ignored. This option should probably not be used because the default works quite well. To allow for the thesis to be printed in September, we start the default academic year on October 1.

`\l_kulemt_opt_acyear_int` This variable holds the starting year.

```
\int_new:N \l_kulemt_opt_acyear_int
\int_set:Nn \l_kulemt_opt_acyear_int
  { \c_sys_year_int \int_compare:nNnT { \c_sys_month_int } < {10} { - 1 } }
\keys_define:nn {kulemt}
  {
    acyear .code:n =
      {
        \tl_if_empty:nF {#1}
          {
            \regex_extract_once:nnN {\A\d{4}} {#1} \l_tmpa_seq
            \seq_if_empty:NTF \l_tmpa_seq
              { \msg_error:nnn {kulemt} {opt/invalid-year} {#1} }
              {
                \int_set:Nn \l_kulemt_opt_acyear_int
                  { \seq_item:Nn \l_tmpa_seq {1} }
              }
          }
      } ,
```

```
    acyear .value_required:n = true
  }
\msg_new:nnn {kulemt} {opt/invalid-year}
  { Value~ `\tl_to_str:n{#1}'~ of~ option~ 'acyear'~ does~ not~ start~ with~
    a~ 4-digit~ number. }
```

(*End of definition for* `\l_kulemt_opt_acyear_int`. *This variable is documented on page 30.*)

### 2.3.3 Conditionally generating pages

\l_kulemt_include_coverpage_bool
\l_kulemt_include_frontpages_bool
\l_kulemt_include_text_bool

These variables determine which pages are generated. By default, the cover page is not generated.

```
\bool_new:N \l_kulemt_include_coverpage_bool
\bool_new:N \l_kulemt_include_frontpages_bool
\bool_new:N \l_kulemt_include_text_bool
\bool_set_true:N \l_kulemt_include_frontpages_bool
\bool_set_true:N \l_kulemt_include_text_bool
```

coverpageonly (*option*)
frontpagesonly (*option*)

The options 'coverpageonly' and 'frontpagesonly' set these variables.

```
\keys_define:nn {kulemt}
  {
    coverpageonly .code:n =
      {
        \bool_set_true:N  \l_kulemt_include_coverpage_bool
        \bool_set_false:N \l_kulemt_include_frontpages_bool
        \bool_set_false:N \l_kulemt_include_text_bool
      } ,
    coverpageonly .value_forbidden:n = true ,
    frontpagesonly .code:n =
      {
        \bool_set_false:N \l_kulemt_include_coverpage_bool
        \bool_set_true:N  \l_kulemt_include_frontpages_bool
        \bool_set_false:N \l_kulemt_include_text_bool
      } ,
    frontpagesonly .value_forbidden:n = true
  }
```

(*End of definition for* `\l_kulemt_include_coverpage_bool`, `\l_kulemt_include_frontpages_bool`, *and* `\l_kulemt_include_text_bool`. *These variables are documented on page 30.*)

## 2.4 Option handling commands

### 2.4.1 The setup command

\setup

This command lets you set additional options. Currently we assume their is no need for category code changes.

```
\NewDocumentCommand \setup { m }
  {
    \bool_set_false:N \l__kulemt_opt_allow_class_option_bool
    \keys_set:nn {kulemt} {#1}
  }
\@onlypreamble \setup
```

### 2.4.2 Handling class options

\l_kulemt_master_language_tl

This variable stores the master language. It is set from the configuration key `language` when setting the master, but with document category codes. Otherwise babel gets confused.

```
\tl_new:N \l_kulemt_master_language_tl
```

(*End of definition for* `\l_kulemt_master_language_tl`. *This variable is documented on page* 29.)

\kulemt_process_class_options:

Only one document class option is required: '`master`'. So, after handling the options, we set the property list of the master and the master language. If the option is missing a fatal error is raised.
If the variable `\l_kulemt_language_tl` is still empty at the end of option processing, it is set to the master language.

```
\cs_new_protected:Nn \kulemt_process_class_options:
  {
    \bool_set_true:N \l__kulemt_opt_allow_class_option_bool
    \ProcessKeyOptions \scan_stop:
    \tl_if_empty:NTF \l_kulemt_opt_master_tl
      {
        \bool_if:NF \l_kulemt_opt_article_bool
          { \msg_fatal:nn {kulemt} {opt/master-missing} }
        \tl_if_empty:NT \l_kulemt_language_tl
          { \tl_set:Nn \l_kulemt_language_tl {english} }
        \seq_put_right:NV \l_kulemt_babel_seq \l_kulemt_language_tl
      }
      {
        \kulemt_set_master:V \l_kulemt_opt_master_tl
        \kulemt_master_get_required_item:nN {language} \l_tmpa_str
        \tl_set_rescan:NnV \l_kulemt_master_language_tl {} \l_tmpa_str
        \tl_if_empty:NT \l_kulemt_language_tl
          { \tl_set_eq:NN \l_kulemt_language_tl \l_kulemt_master_language_tl }
        \bool_lazy_and:nnF
          { \str_if_eq_p:Vn \l_kulemt_language_tl {british} }
          { \str_if_eq_p:Vn \l_kulemt_master_language_tl {english} }
          { \seq_put_right:NV \l_kulemt_babel_seq \l_kulemt_master_language_tl }
        \str_if_eq:VVF \l_kulemt_language_tl \l_kulemt_master_language_tl
          { \seq_put_right:NV \l_kulemt_babel_seq \l_kulemt_language_tl }
      }
  }
\msg_new:nnn {kulemt} {opt/master-missing}
  { The~ required~ document~ class~ option~ 'master'~ is~ missing. }
```

(*End of definition for* `\kulemt_process_class_options:`. *This function is documented on page* 28.)

⟨/class⟩

# File V
# kulemt-load.dtx
# Loading the required class and packages

`\kulemt_selectlanguage:n` `\kulemt_selectlanguage:(o|V)`

`\kulemt_selectlanguage:n {⟨language⟩}`

Since a language and one of its dialects cannot be used at the same time as an option to babel, a combination of '`english`' as a master language and '`british`' as a text language is not possible. Therefore we use '`british`' to select the language instead of '`english`' when '`british`' is one of the babel options.

Furthermore, it can handle an empty ⟨*language*⟩ argument. In that case no new language is selected.

`\theHsubfigure` `\theHsubtable`

These hyperref variables get a default value to avoid link label conflicts.

`\l_kulemt_incompatible_clist`

Comma list holding the incompatible packages. Using them raises an error unless it is an emulated package.

## 1 Implementation

```
⟨*class⟩
⟨@@=kulemt_load⟩
```

Some x-variants are since October 2023 version no longer available. We generate here the e-type variants for functions which don't exist yet and are used in this file.

```
\cs_generate_variant:Nn \msg_fatal:nnn { nne }
\cs_generate_variant:Nn \seq_put_left:Nn { Ne }
```

### 1.1 The memoir class

After processing the document class options, the memoir class is loaded.

```
\kulemt_process_class_options:
\seq_put_left:Ne \l_kulemt_memoir_options_seq
  { \int_use:N \l_kulemt_opt_ptsize_int pt }
\exp_last_unbraced:Ne \LoadClass
  { [ \seq_use:Nn \l_kulemt_memoir_options_seq {,} ] } {memoir} [2018/04/04]
```

## 1.2 The babel package

Since the main language can be set both as a gobal and a local option, we cannot use it as the last local babel option. So the `main` option must be used here.

```
\exp_last_unbraced:Ne \RequirePackage
  { [ \seq_use:Nn \l_kulemt_babel_seq {,} , main = \l_kulemt_language_tl ] }
  {babel}
```

\kulemt_selectlanguage:n
\kulemt_selectlanguage:o
\kulemt_selectlanguage:V

Automatically take care of language dialects (currently only British) and an empty argument.
Note: Using `\tl_if_empty:nF` instead of `\str_if_empty:nF` because the latter is only available since 2022-04-10.

```
\cs_new_protected:Nn \kulemt_selectlanguage:n
  {
    \tl_if_empty:nF {#1}
      {
        \str_if_eq:nnTF {#1} {english}
          {
            \seq_if_in:NnTF \l_kulemt_babel_seq {british}
              { \selectlanguage {british} }
              { \selectlanguage {#1} }
          }
          { \selectlanguage {#1} }
      }
  }
\cs_generate_variant:Nn \kulemt_selectlanguage:n { o, V }
```

(*End of definition for* `\kulemt_selectlanguage:n`. *This function is documented on page* 45.)

English/British and Dutch translations of additional memoir commands are also provided. Later on the translations for the ... `name` variables of the new commands and environments are also added to these variables.

```
\cs_if_exist:NTF \captionsbritish
  { \tl_put_right:Nn \captionsbritish }
  {
    \cs_if_exist:NTF \captionsenglish
      { \tl_put_right:Nn \captionsenglish }
      { \use_none:n }
  }
    {
      \tl_set:Nn \appendixtocname  { Appendices }
      \tl_set:Nn \appendixpagename { Appendices }
      \tl_set:Nn \appendixrefname  { Appendix \nobreakspace }
      \tl_set:Nn \chapterrefname   { Chapter \nobreakspace }
      \tl_set:Nn \figurerefname    { Figure }
      \tl_set:Nn \pagerefname      { page }
      \tl_set:Nn \partrefname      { Part \nobreakspace }
      \tl_set:Nn \tablerefname     { Table }
    }
\cs_if_exist:NTF \captionsdutch
  { \tl_put_right:Nn \captionsdutch }
  { \use_none:n }
    {
      \tl_set:Nn \appendixtocname  { B\ij lagen }
```

```
      \tl_set:Nn \appendixpagename { B\ij lagen }
      \tl_set:Nn \appendixrefname  { b\ij lage \nobreakspace }
      \tl_set:Nn \chapterrefname   { hoofdstuk \nobreakspace }
      \tl_set:Nn \figurerefname    { figuur }
      \tl_set:Nn \pagerefname      { pagina }
      \tl_set:Nn \partrefname      { Deel \nobreakspace }
      \tl_set:Nn \tablerefname     { tabel }
    }
```

## 1.3 The graphicx package

This package is needed to include graphics, e.g., the logo on the front page.

```
\RequirePackage{graphicx}
```

## 1.4 The hyperref package

The package hyperref is very useful for online PDF files, less for printed documents. Because the package interacts with many other packages, it is not loaded by default.

If you load hyperref, it is best to load it as one of the last packages of the preamble according to its documentation since it overwrites definitions. This may change in the future.

\theHsubfigure
\theHsubtable

To avoid link name conflicts, subfloats should be numbered within the parent float. The defaults are provided for the most common cases of subfigures and subtables. It is no problem (according to the hyperref documentation) to define these, even if hyperref is not used.

```
\tl_set:Nn \theHsubfigure { \theHfigure . \arabic{subfigure} }
\tl_set:Nn \theHsubtable  { \theHtable  . \arabic{subtable}  }
```

(*End of definition for* `\theHsubfigure` *and* `\theHsubtable`*. These functions are documented on page* 45.)

## 1.5 Incompatible packages

Some packages may break the assumptions of this class, e.g. about the layout. An error will be raised if they are used.

\l_kulemt_incompatible_clist

The list of incompatible packages, such as polyglossia, which is a replacement for babel. The list can be extended later on inside the definition of the kulemt class.

```
\clist_new:N  \l_kulemt_incompatible_clist
\clist_set:Nn \l_kulemt_incompatible_clist { polyglossia }
```

(*End of definition for* `\l_kulemt_incompatible_clist`*. This variable is documented on page* 45.)

⟨/class⟩

After loading the class, an error is issued if an incompatible package is used. To cope with emulated packages, `\AtBeginPackage` is used instead of testing at the end of the preamble.

```
⟨*atend⟩
\clist_map_inline:Nn \l_kulemt_incompatible_clist
  {
    \AtBeginPackage {#1}
      { \msg_fatal:nne {kulemt} {load/incompatible} {#1} }
```

```
      {}
  }
\msg_new:nnn {kulemt} {load/incompatible}
  { Package~ '#1'~ is~ incompatible~ with~ kulemt. }
⟨/atend⟩
```

# File VI
# kulemt-layout.dtx
# Document layout

\maketitle The LaTeX command \maketitle is made a no-op in the thesis layout because it otherwise conflicts with our layout.

\clearforchapter The memoir command \clearforchapter is redefined to avoid empty pages in the front and back matter. These parts of the text are always openany. If you don't like this, you can use the \openleft or \openright command in the document.

## 1 Implementation

```
⟨*class⟩
⟨@@=kulemt_layout⟩
```

First the popular packages incompatible with this layout are enumerated.

```
\clist_put_right:Nn \l_kulemt_incompatible_clist
  { a4, a4wide, anysize, fullpage, geometry, typearea }
```

\maketitle Since we have our own functions to typeset the title and authors, the user command \maketitle should do nothing in the thesis layout except stop the scanning.

```
\bool_if:NF \l_kulemt_opt_article_bool
  { \cs_gset_eq:NN \maketitle \scan_stop: }
```

### 1.1 Page layout

The default \headheight and \headsep from memoir are left as is, but the text body dimensions are redefined depending on the text point size (10pt and 11pt respectively).

```
\int_compare:nNnTF { \l_kulemt_opt_ptsize_int } = {10}
  {
    \dim_set:Nn \textwidth  { 13cm }
    \dim_set:Nn \textheight { 20cm }
  }
  {
    \dim_set:Nn \textwidth  { 14cm }
    \dim_set:Nn \textheight { 215mm }
  }
```

The inner (\spinemargin) and outer (\foremargin) margins are computed as follows:

$$\texttt{\foremargin} = 0.6\,(\texttt{\paperwidth} - \texttt{\textwidth} - \text{binding})$$
$$\texttt{\spinemargin} = 0.4\,(\texttt{\paperwidth} - \texttt{\textwidth} - \text{binding}) + \text{binding}$$

When equal margins are requested, the visible parts of both margins are made equal (use a factor 0.5 instead of 0.4 and 0.6).

```
\dim_set:Nn \foremargin
  {
```

```
    ( \paperwidth - \textwidth - \l_kulemt_opt_bind_dim )
    * \bool_if:NTF \l_kulemt_opt_lrequal_bool {5} {6} /10
  }
\dim_set:Nn \spinemargin
  {
    ( \paperwidth - \textwidth - \l_kulemt_opt_bind_dim )
    * \bool_if:NTF \l_kulemt_opt_lrequal_bool {5} {4} /10
    + \l_kulemt_opt_bind_dim
  }
```

Margin notes get a fixed width independent of one side or two side printing. This makes sure that generating a PDF version (one side) or a printed version (two side) have the same text on each page. The separation between notes is kept as small as possible, as well as the distance from the text block.

```
\dim_set:Nn \marginparwidth { 56pt }
\dim_set:Nn \marginparsep   { 1.2\onelineskip }
\dim_set_eq:NN \marginparpush \onelineskip
```

The lower margin is 1.2 times the upper margin. The header parameters are set to the default values.

```
\setulmargins{*}{*}{1.2}
\setheaderspaces{*}{\headsep}{*}
```

Finish up the layout definitions. Redo this at the end of the document preamble in case users redefine some parameters (which they shouldn't of course).

```
\checkthelayout
\fixthelayout
\kulemt_at_end_preamble:n { \checkandfixthelayout }
```

`\clearforchapter` The `open...` options only control the main matter chapters. Chapters in the front and back matter are always `openany`.

```
\RenewDocumentCommand \clearforchapter {}
  {
    \legacy_if:nTF {@mainmatter}
      {
        \legacy_if:nTF {@openleft}
          { \cleartoverso }
          {
            \legacy_if:nTF {@openright}
              { \cleartorecto }
              { \clearpage    }
          }
      }
      { \clearpage }
  }
```

## 1.2 Page styles

By default the pagestyle ruled is used. However for front matter (actually for non-main matter) the header on odd pages is the same as on even pages, because typically front matter chapters have no sections.

```
\makeoddhead {ruled} {} {}
  { \legacy_if:nTF {@mainmatter} { \rightmark } { \scshape \leftmark } }
\pagestyle {ruled}
```

The nohead pagestyle puts the page number in the footer at the outer margin.

```
\makepagestyle {nohead}
\makeevenfoot {nohead} { \thepage } {} {}
\makeoddfoot  {nohead} {} {} { \thepage }
```

The chapter pagestyle is aliased to this new pagestyle.

```
\aliaspagestyle {chapter} {nohead}
```

## 1.3 Section numbering

Sections are numbered up to the subsection level.

```
\maxsecnumdepth {subsection}
```

But numbering in the table of contents ends at the section level.

```
\maxtocdepth {section}
```

## 1.4 Content lists

The content lists are typeset ragged right without hyphenation.

```
\setrmarg { 2.55em plus 1fil }
```

In memoir, content lists don't start a new page. By default it is done here for the table of contents. It is no longer done for the list of figures or the list of tables, as was the default in version 1.

```
\tl_set:Nn \tocheadstart { \clearforchapter \chapterheadstart }
```

For these lists the space before chapter items is halved. This is only important when the lists don't start a new page.

```
\skip_set:Nn \cftbeforechapterskip { 1ex plus 1pt }
```

## 1.5 Tables and figures

The captions of tables and figures have the last line centered. The caption name is printed in small caps. Because of bugs in some versions of memoir the font settings for the caption name must be undone for the caption title.

```
\captionnamefont { \scshape }
\captiontitlefont { \upshape }
\captionstyle [ \centering ] { \centerlastline }
```

⟨/class⟩

File VII

# kulemt-front.dtx
# Front pages

The front material consists of the cover page and the front pages (the title page, and the copyright page). The cover page and the title page have exactly the same layout.

An Helvetica font must be used on the front material. The best look-alike in LaTeX is "`TeX Gyre Heros`". It is available in OpenType as well as Postscript font format. Note that math is not supported on the front pages, so try to avoid it.

`\kulemt_front_font:` The command `\kulemt_front_font:` selects the appropiate font for a front page, taking into account the TeX engine used. It sets the font family and if needed the encoding.

## 1 Implementation

1176 ⟨*class⟩
1177 ⟨@@=kulemt_front⟩

Some x-variants are since October 2023 version no longer available. We generate here the e-type variants for functions which don't exist yet and are used in this file.

1178 `\cs_generate_variant:Nn \msg_warning:nnn { nne }`

### 1.1 Front page font

If possible, the font "`TeX Gyre Heros`" is used, in OpenType or as a Postscript font format, depending on the font encoding.

If possible, we prefer the `TU` font encoding, which is the most complete one. Otherwise the `T1` font encoding is preferred over `OT1` because languages such as Dutch often use accented characters.

`\__kulemt_front_define_font:` `\kulemt_front_font:` The function `\__kulemt_front_define_font:` defines `\kulemt_front_font:`, which changes the font family and encoding to the appropriate ones for a front page. To avoid collisions with scaled Helvetica body fonts, a specific front page font is defined based on unscaled Helvetica. Since `\__kulemt_front_define_font:` declares a font and depends on possible preamble settings, it can only be used at the end of the preamble.
Even in the article layout, which prints no front pages, `\kulemt_front_font:` is defined, in case you want it.
When `T1` encoding is used, the `TS1` encoding is also provided for some special symbols,

such as \textcopyright, but only for the regular font shape. If more font shapes are needed, please submit a feature request.

```
\cs_new_protected:Nn \__kulemt_front_define_font:
  {
    \file_if_exist:nTF {tgheros.sty}
      {
        \bool_set:Nn \l_tmpa_bool { \str_if_eq_p:Vn \encodingdefault {TU} }
        \bool_if:NT \l_tmpa_bool
          {
            \IfPackageLoadedTF {fontspec}
              {}
              {
                \file_if_exist:nTF {fontspec.sty}
                  { \RequirePackage{fontspec} }
                  { \bool_set_false:N \l_tmpa_bool }
              }
          }
        \bool_if:NTF \l_tmpa_bool
          {
```

When fontspec is used, the front page font is loaded by file name, not by font name. This way it should work in LuaTeX as well as in XeTeX.

```
            \newfontfamily \kulemt_front_font: {texgyreheros}
              [ Extension = .otf,
                UprightFont = *-regular, BoldFont = *-bold,
                ItalicFont = *-italic, BoldItalicFont = *-bolditalic ]
          }
          {
            \str_case:VnF \encodingdefault { {T1} {} {OT1} {} }
              { \msg_warning:nne {kulemt} {front/encoding} { \encodingdefault } }
            \DeclareFontFamily{TS1}{kulemtfpf}{}
            \DeclareFontShape{TS1}{kulemtfpf}{m}{n}{<-> ts1-qhvr}{}
            \DeclareFontFamily{T1}{kulemtfpf}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {n}   {<-> ec-qhvr}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {it}  {<-> ec-qhvri}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {sc}  {<-> ec-qhvr-sc}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {scit}{<-> ec-qhvri-sc}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {it}  {<-> ec-qhvbi}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {n}   {<-> ec-qhvb}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {scit}{<-> ec-qhvbi-sc}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {sc}  {<-> ec-qhvb-sc}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {sl}  {<-> ssub*kulemtfpf/m/it}{}
            \DeclareFontShape{T1}{kulemtfpf}{m} {scsl}{<-> ssub*kulemtfpf/m/scit}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {sl}  {<-> ssub*kulemtfpf/b/it}{}
            \DeclareFontShape{T1}{kulemtfpf}{b} {scsl}{<-> ssub*kulemtfpf/b/scit}{}
            \DeclareFontShape{T1}{kulemtfpf}{bx}{it}  {<-> ssub*kulemtfpf/b/it}{}
            \DeclareFontShape{T1}{kulemtfpf}{bx}{n}   {<-> ssub*kulemtfpf/b/n}{}
            \DeclareFontShape{T1}{kulemtfpf}{bx}{scit}{<-> ssub*kulemtfpf/b/scit}{}
            \DeclareFontShape{T1}{kulemtfpf}{bx}{sc}  {<-> ssub*kulemtfpf/b/sc}{}
            \cs_new:Nn \kulemt_front_font:
              { \fontencoding{T1} \fontfamily{kulemtfpf} \selectfont }
          }
      }
      {
```

```
      \str_case:VnF \encodingdefault { {T1} {} {OT1} {} }
        { \msg_warning:nne {kulemt} {front/encoding} { \encodingdefault } }
      \DeclareFontFamily{TS1}{kulemtfpf}{}
      \DeclareFontShape{TS1}{kulemtfpf}{m}{n}{<-> phvr8c}{}
      \DeclareFontFamily{T1}{kulemtfpf}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {n}   {<-> phvr8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {sc}  {<-> phvrc8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {sl}  {<-> phvro8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{n}   {<-> phvb8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{sc}  {<-> phvbc8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{sl}  {<-> phvbo8t}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {scsl}{<-> sub* kulemtfpf/m/sl}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {it}  {<-> ssub*kulemtfpf/m/sl}{}
      \DeclareFontShape{T1}{kulemtfpf}{m} {scit}{<-> ssub*kulemtfpf/m/scsl}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{scsl}{<-> sub* kulemtfpf/bx/sl}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{it}  {<-> ssub*kulemtfpf/bx/sl}{}
      \DeclareFontShape{T1}{kulemtfpf}{bx}{scit}{<-> ssub*kulemtfpf/bx/scsl}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {n}   {<-> ssub*kulemtfpf/bx/n}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {it}  {<-> ssub*kulemtfpf/bx/sl}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {sc}  {<-> ssub*kulemtfpf/bx/sc}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {scit}{<-> ssub*kulemtfpf/bx/scit}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {scsl}{<-> ssub*kulemtfpf/bx/scsl}{}
      \DeclareFontShape{T1}{kulemtfpf}{b} {sl}  {<-> ssub*kulemtfpf/bx/sl}{}
      \cs_new:Nn \kulemt_front_font:
        { \fontencoding{T1} \fontfamily{kulemtfpf} \selectfont }
    }
  }
\msg_new:nnn {kulemt} {front/encoding}
  {
    The~ front~ page~ cannot~ use~ encoding~ '#1'.\\
    Font~ encoding~ 'T1'~ is~ used~ instead.
  }
\kulemt_at_end_preamble:n { \__kulemt_front_define_font: }
```

(*End of definition for* `\__kulemt_front_define_font:` *and* `\kulemt_front_font:`*. This function is documented on page 52.*)

## 1.2 Typesetting the title page

`\__kulemt_front_print_title_page:`

The title page contains no header or footer. It starts in the front matter as page −1. Since the title page is followed by the copyright page (page 0), the first real page can start at 1. The page number of the cover page is irrelevant.

```
\cs_new_protected:Nn \__kulemt_front_print_title_page:
  {
    \clearpage
    \setcounter {page} {-1}
    \thispagestyle {empty}
```

The text on the title page starts 1 cm below the upper page edge.

```
    \hbox:n {}
    \skip_vertical:n
      { 1cm - \uppermargin - \tex_topskip:D - \tex_baselineskip:D }
```

The typeset area on the title page is different from the rest of the text. It is always centered horizontally, also with two side printing. The margins are 2 cm, resulting in a text width of 17 cm on A4 paper.

```
\hbox_to_wd:nn { \tex_hsize:D }
  {
    \skip_horizontal:n { 2cm - \spinemargin }
    \vbox_to_zero:n
      {
        \dim_set:Nn \tex_hsize:D { 17cm }
```

All elements on the title page are positioned, so avoid inserting automatic glue.

```
        \skip_zero:N \tex_lineskip:D
        \skip_zero:N \tex_parskip:D
```

Micro-typography is disabled on the title page, so we make sure that the typesetting of the title page doesn't depend on the presence of the microtype package.

```
        \cs_if_exist:NT \microtypesetup
          { \microtypesetup { activate=false } }
```

The title page text is typeset ragged right in Helvetica using the master's program language.

```
        \fontsize{12}{14} \kulemt_front_font:
        \raggedright
        \kulemt_selectlanguage:V \l_kulemt_master_language_tl
```

The first line contains the KU Leuven logo on the left, eventually combined with a faculty logo. The height of this logo line is 3 cm, which corresponds to the height of the combined KU Leuven and faculty logo. The KU Leuven logo (without attached faculty logo) has fixed dimensions (56 mm, 2 cm). The combined logo image is used at its natural dimensions, so it is up to the provider of the combined logo to make sure the KU Leuven rules are obeyed. The left margin of the KU Leuven logo is 1 cm, so it enters 1 cm into the left margin of the typeblock.

```
        \tex_noindent:D
        \skip_horizontal:n { -1cm }
        \vbox_to_ht:nn { 3cm }
          {
            \kulemt_master_get_item_or_fallback:nnN
              {faculty.logo} {logokul} \l_tmpa_tl
            \exp_args:NV \includegraphics \l_tmpa_tl
            \tex_vss:D
          }
```

The minimal space before the title is 40 pt but it stretches twice as fast as the space below the author.
The title and the subtitle are printed in the main text language.

```
        \skip_vertical:n { 40pt plus 2fill }
        \group_begin:
          \kulemt_selectlanguage:V \l_kulemt_language_tl
          \fontsize {24.88} {30} \selectfont
          \l_kulemt_opt_title_tl
          \par
```

If a subtitle is given, it is typeset at the appropriate size and at a fixed distance below the title.

```
          \tl_if_empty:NF \l_kulemt_opt_subtitle_tl
```

```
    {
      \skip_vertical:n { 1em }
      \fontsize {17.28} {22} \selectfont
      \l_kulemt_opt_subtitle_tl
      \par
    }
\group_end:
```

The minimal space before the authors is again 40 pt but with a very limited stretching. The space after it is 30 pt with the standard stretching.

```
\skip_vertical:n { 40pt plus .3fill }
\group_begin:
  \fontsize {14.4} {18} \selectfont
  \seq_use:Nn \l_kulemt_opt_author_seq { \\ }
  \par
\group_end:
\skip_vertical:n { 30pt plus 1fill }
```

The rest is ordinary text which is typeset ragged left, occupying at most half of the text body. First comes the degree, including the option or major topic. Multiple options are separated by "and" (or "en" in Dutch). It is followed by the promoter(s). On the title page, the assessors and the assistants are also listed. The space below this text is 20 pt with the same stretching as above the title.

```
\tex_noindent:D
\tex_hfill:D
\vbox:n
  {
    \dim_set:Nn \tex_hsize:D { .5\textwidth }
    \raggedleft
    \kulemt_cfg_print_text_ucfirst:n {title.pre} ~
    \kulemt_master_print_required_item:n {name}
    \seq_if_empty:NF \l_kulemt_opt_masteroption_seq
      {
        ,~
        \seq_use:Nn \l_kulemt_opt_masteroption_seq
          { ~ \kulemt_cfg_print_text:n {and} ~ }
      }
    \par
    \__kulemt_front_print_people:n {promoter}
    \__kulemt_front_print_people:n {assessor}
    \__kulemt_front_print_people:n {assistant}
  }
\skip_vertical:n { 20pt plus 2fill }
```

The academic year is printed below the text and centered on the page, with a space of 15 pt below it.

```
\centering
\fontsize{14.4}{18} \selectfont
\kulemt_cfg_print_text_ucfirst:n {acyear.pre} ~
\int_use:N \l_kulemt_opt_acyear_int
\c_space_tl \textendash \c_space_tl
\int_eval:n { \l_kulemt_opt_acyear_int + 1 }
\par
\skip_vertical:n {15pt}
```

A bottom margin of 1 cm results on A4 paper in a body height of 27.7 cm.

```
            \skip_vertical:n {-277mm}
          }
        \tex_hss:D
      }
    \clearpage
  }
```

(*End of definition for* `\__kulemt_front_print_title_page:`.)

`\__kulemt_front_print_people:n` Print the items of option `#1` on the title page, preceded by an heading.

```
\cs_new_protected:Nn \__kulemt_front_print_people:n
  {
    \seq_if_empty:cF { l_kulemt_opt_ #1 _seq }
      {
        \medskip
        \group_begin:
          \fontsize{12}{14.5} \fontshape\itdefault \selectfont
          \kulemt_cfg_print_text_from_opt_ucfirst:n {#1} \par
          \skip_vertical:n {2pt}
        \group_end:
        \seq_use:cn { l_kulemt_opt_ #1 _seq } { \\ }
        \par
      }
  }
```

(*End of definition for* `\__kulemt_front_print_people:n`.)

## 1.3 Typesetting the copyright page

The copyright page is based on the one used for a PhD thesis of the Arenberg Doctoral School of Science, Engineering & Technology.

`\__kulemt_front_print_copyright_page:` The copyright page contains no header or footer, with copyright notices for the ⟨*master language*⟩ as well as for the ⟨*document language*⟩ at the bottom of the page. The copyright notice for a ⟨*language*⟩ is printed by `\__kulemt_front_print_copyright_notice_`⟨*language*⟩`:`. Paragraphs in the copyright notice are typeset without indentation and half a line of spacing between them. If the text and the master's program language are the same, a copyright notice is printed in that language. If they differ, the English version comes first. If no copyright notice is defined for the ⟨*language*⟩, one for English is printed.
To avoid hyphenation, `\raggedright` is used and a smaller font size.

```
\cs_new_protected:Nn \__kulemt_front_print_copyright_page:
  {
    \clearpage
    \thispagestyle {empty}
    \hbox:n {}
    \tex_vfill:D
    \group_begin:
      \fontsize{10}{12} \kulemt_front_font:
      \raggedright
      \dim_zero:N \tex_parindent:D
      \skip_set:Nn \tex_parskip:D { .5\tex_baselineskip:D }
```

```
      \kulemt_selectlanguage:V \l_kulemt_master_language_tl
      \textcopyright \c_space_tl
      \int_eval:n { \l_kulemt_opt_acyear_int + 1 }
      \c_space_tl KU \nobreakspace Leuven
      \kulemt_master_get_faculty_name:N \l_tmpa_tl
      \tl_if_empty:NF \l_tmpa_tl
        { \c_space_tl \textendash \c_space_tl \l_tmpa_tl }
      \\
      \kulemt_cfg_print_text_ucfirst:n {publisher.pre} \c_space_tl
      \seq_use:Nnnn \l_kulemt_opt_author_seq
        { ~ \kulemt_cfg_print_text:n {and} ~ }
        { , ~ }
        {
          \str_if_eq:VnF \languagename {dutch} { , }
          \c_space_tl \kulemt_cfg_print_text:n {and} ~
        }
      , \\
      \kulemt_master_print_required_item:n {contact.address}
      \par
      \kulemt_cfg_print_text:n {copyright} \par
      \bool_xor:nnT
        { \str_if_eq_p:Vn \l_kulemt_master_language_tl {dutch} }
        { \str_if_eq_p:Vn \l_kulemt_language_tl {dutch} }
        {
          \kulemt_selectlanguage:V \l_kulemt_language_tl
          \kulemt_cfg_print_text:n {copyright} \par
        }
    \group_end:
    \clearpage
  }
```

(*End of definition for* `\__kulemt_front_print_copyright_page:`.)

## 1.4 Printing the required pages

At the beginning of the document in a thesis layout, the cover page is printed if needed, which depends on `\l_kulemt_include_coverpage_bool`. If this boolean is true, the title page and the copyright page are printed next. If no text must be printed, the document ends here.

In an article layout, no front pages or cover page are printed and we start immediately with the main matter (which is the default in memoir).

The hyperref package requires a unique printed page number. Since non-positive page numbers have no roman representation, the `\frontmatter` is only switched on after the copyright page.

We can't use `\AtBeginDocument` here, because some packages assume that no text is generated before the commands they add to that hook. An example is the externalization library of the package tikz.

```
\kulemt_after_begin_document:n
  {
    \bool_if:NF \l_kulemt_opt_article_bool
      {
        \bool_if:NT \l_kulemt_include_coverpage_bool
          { \__kulemt_front_print_title_page: }
```

```
      \bool_if:NT \l_kulemt_include_frontpages_bool
        {
          \__kulemt_front_print_title_page:
          \__kulemt_front_print_copyright_page:
        }
      \bool_if:NF \l_kulemt_include_text_bool
        { \end{document} }
      \frontmatter
      \AtEndDocument { \__kulemt_front_forgot_mainmatter: }
    }
}
```

__kulemt_front_forgot_mainmatter:

If text is printed, users must switch to the main matter themselves and we make sure that they don't forget it. The function `\__kulemt_front_forgot_mainmatter:` will be called at the end of the document. So it is undefined when `\mainmatter` is called. Since `\mainmatter` tests for a trailing star, we must prepend to it.

```
\cs_new_protected:Nn \__kulemt_front_forgot_mainmatter:
  { \msg_error:nnn {kulemt} {mainmatter} }
\msg_new:nnn {kulemt} {mainmatter}
  { You~ forgot~ to~ use~ ' \token_to_str:N \mainmatter '. }
\tl_put_left:Nn \mainmatter
  { \cs_gset_eq:NN  \__kulemt_front_forgot_mainmatter: \prg_do_nothing: }
```

(*End of definition for* `\__kulemt_front_forgot_mainmatter:`.)

⟨/class⟩

# File VIII
# kulemt-extra.dtx
# Extra commands and environments

`abstract` (*env.*) The `abstract` environment is typeset like a chapter, but only in the thesis layout.

`abstract*` (*env.*) The `abstract*` environment works like the `abstract` environment, but it uses the language from its optional argument. By default this is the master's program language.

`preface` (*env.*) The `preface` environment holds the preface text. It has one optional argument, which holds the preface author. The default preface author is the value of the `author` option.

`\listoffiguresandtables` `[`⟨*title*⟩`]`
This command lists figures and tables on one page. The chapter title is ⟨*title*⟩.
This command is deprecated but kept at this moment for compatibility reasons. It may disappear in a future version.

`\listfiguresandtablesname` This variable holds the default ⟨*title*⟩ of the combined list of figures and tables.

## 1 Implementation

1425 ⟨*class⟩
1426 ⟨@@=kulemt_extra⟩

Some x-variants are since October 2023 version no longer available. We generate here the e-type variants for functions which don't exist yet and are used in this file.

```
1427 \cs_generate_variant:Nn \tl_put_right:Nn { Ne }
```

### 1.1 Front matter

#### 1.1.1 Environment `preface`

`preface` (*env.*) The `preface` environment has one optional argument, namely the preface author. The default preface author is the value of the `author` option, one author per line. The preface is printed as a single page chapter. The variable `\prefacename` is defined by babel.

```
1428 \NewDocumentEnvironment {preface} { o }
1429   {
1430     \chapter { \prefacename }
1431     \IfNoValueTF {#1}
1432       {
1433         \seq_if_empty:NTF \l_kulemt_opt_author_seq
1434           { \tl_clear:N \l__kulemt_extra_preface_author_tl }
1435           {
1436             \tl_set:Nn \l__kulemt_extra_preface_author_tl
1437               { \seq_use:Nn \l_kulemt_opt_author_seq { \\ } }
1438           }
1439       }
1440       { \tl_set:Nn \l__kulemt_extra_preface_author_tl {#1} }
1441   }
1442   {
1443     \par
1444     \tl_if_blank:VF \l__kulemt_extra_preface_author_tl
```

```
      {
        \bigskip
        \raggedleft
        \itshape
        \l__kulemt_extra_preface_author_tl
      }
    \tex_vfill:D
    \clearpage
  }
```

\l__kulemt_extra_preface_author_tl This variable remembers the optional argument of the environment `preface` until the end of the environment.

```
\tl_new:N \l__kulemt_extra_preface_author_tl
```

(*End of definition for* `\l__kulemt_extra_preface_author_tl`.)

### 1.1.2 Environment `abstract` and `abstract*`

`abstract` (*env.*) In the thesis layout the `abstract` environment is redefined as an ordinary chapter. To know the layout we have to wait until the class is loaded.
The variable `\abstractname` is defined by babel.

```
\AtEndOfClass
  {
    \bool_if:NF \l_kulemt_opt_article_bool
      {
        \RenewDocumentEnvironment {abstract} {}
          { \chapter { \abstractname } }
          { \clearpage }
      }
  }
```

`abstract*` (*env.*) The `abstract*` environment sets the language from the optional argument before starting an `abstract`. The optional argument defaults to the master's program language.

```
\NewDocumentEnvironment {abstract*} { O { \l_kulemt_master_language_tl } }
  {
    \kulemt_selectlanguage:o {#1}
    \abstract
  }
  { \endabstract }
```

### 1.1.3 Content lists

`\listoffiguresandtables` The command `\listoffiguresandtables` lists first the figures and then the tables on the same page. The optional argument sets the chapter title, which is `\listfiguresandtablesname` by default.

```
\NewDocumentCommand \listoffiguresandtables { O{\listfiguresandtablesname} }
  {
    \chapter {#1}
    \tl_set:Nn \@lofmaketitle { \section* { \listfigurename } }
    \listoffigures*
    \tl_set:Nn \@lotmaketitle { \section* { \listtablename } }
    \listoftables*
  }
```

`\listfiguresandtablesname` The variable `\listfiguresandtablesname` holds the default title for a page combining the list of figures and of tables. An English (or British) as well as a Dutch translation is provided.

```
\tl_new:N \listfiguresandtablesname
\tl_set:Nn \listfiguresandtablesname { List~ of~ Figures~ and~ Tables }
\cs_if_exist:NTF \captionsbritish
  { \tl_put_right:Ne \captionsbritish }
  {
    \cs_if_exist:NTF \captionsenglish
      { \tl_put_right:Ne \captionsenglish }
      { \use_none:n }
  }
    {
      \exp_not:n { \tl_set:Nn \listfiguresandtablesname }
        { \listfiguresandtablesname }
    }
\cs_if_exist:NT \captionsdutch
  {
    \tl_put_right:Nn \captionsdutch
      {
        \tl_set:Nn \listfiguresandtablesname
          { L\ij st~ van~ figuren~ en~ tabellen }
      }
  }
```

⟨/class⟩

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

## L



## M

**N**



**O**



**T**