Class dictionarySCR*

Claudio Beccari Heinrich Fleck

Indice

Sommario

This class has been prepared on an initial nucleus written by Heinrich Fleck (pseudonym); he intended to use it for typesetting his Astronomy Dictionary. This initial part has been documented and extended by Claudio Beccari and here its usage is described.

This class is useful not only for typesetting a specialized dictionary as the one mentioned above, but can be used for typesetting any generic or thematic dictionary.

*Version number v.1.8d; last revision 2013/10/06.

1 Introduction

It was a long time since Heinrich Fleck wanted to write a dictionary on astronomy; as a member of an amateur astronomers group, he wanted to write down a dictionary for the members benefit, but also for the pleasure of collecting in a single document the wealth of information he had collected in this field.

He wanted to typeset a book with the front matter, consisting of the title page, a few pages of instructions and dedications, and the main matter, consisting of the dictionary proper, typeset in two column mode, with bold face entries aligned with their description, as is customary in most dictionaries; he wished the headers with the information of the entries in force at the beginning and at the end of the page, irrespective of their recto or verso position; he wanted a thumbnail in a fixed position on both left and right pages, sticking in the outer margin, containing the initial letter of the entries contained in the page; he wanted to use figures both in column and page width mode without the need of using specific tricks; he did not want to number the figures and desired to be free to put their captions either above or below the illustrative material, even if this implied some degree of manual fine tuning; for tables, besides the usual full width and column width ones, he desired to have also wider tables sticking in the outer margin up to occupy the space normally reserved to marginal notes.

Some entries would contain numbered equations; their counter should be reset at each new entry; in this case there would not be any problem to refer the the single equations contained in their entry, but it might pose problems for referencing other entries equations; therefore the equation reference command should be able to recover also the entry name so as to refer to them in the form "... equation 2 of entry Abbe...".

In order to build this class over an existing framework, we examined the classes book, memoir, and scrbook; eventually we decided that the latter was the most versatile with respect to the various changes, modifications, additions, customizations that were necessary for our goal, which in any case required also a considerable amount of external packages.

We believe the class we obtained is functional also for typesetting any kind of scientific dictionary thanks to the extension packages, but it is also suitable for any other dictionary. In this respect we think we filled a gap that was present in the vast collection of LaTeX extension packages, since we could not find any ready made class with the LaTeX mark up suitable for typesetting a dictionary of this kind.

Just to make an example, the environment `catalogues` and its command `\insertcatalog` allows to insert some astronomical catalogues, but with the same ease it can be used for inserting long tabular material requiring special emphasis on the data.

2 What it is possible to do with this class

The overall set of features that this class exhibits are summarized as follows; all these features will be discussed in more detail in the subsequent sections.

1. First of all, evidently the class is used to typeset a dictionary; the entries are in two column mode. Pictures relating to the entries can be accommodated both within each column and/or on top of the page.
2. Each initial letter is treated as a separate unnumbered chapter; each entry is treated a an unnumbered section.
3. Each initial letter is inserted as a thumbnail on both recto and verso pages.
4. The class allows also to set tables that hang outside the external margin.
5. Tables mai contain figures.
6. Table and figure caption may be inserted either over or beneath the object they refer to by means of specific instructions.
7. A miniature table of contents my be typeset at the beginning of lengthy entries and the same titles and subtitles amy structure the entry development.
8. It is possible to refer to other entries with specific commands so as to have a full reference of the cited entry.
9. If present, the equations are numbered by entries; reference to such equations is a plain one if the equation belongs to the same entry, otherwise the referred equation is completed with its parent entry reference. Footnotes are also numbered by entries and their counter is reset at each new entry; since footnotes are so rare in a dictionary, it should not be a problems to confuse an entry footnote with another entry one.
10. Long tables running on several pages may be typeset; non only star catalogs, as it was done when the class was first designed, but also tables concerning any other tabular material.
11. The main dictionary language may be chosen among English, Italian, French, German; instruction contained in this documentation allow to add the necessary elements for using a another main language different from the previous ones.
12. The decimal separator is being treated in the proper way irrespective of the main language in use.
13. Other features, as, for example, the inclusion of special files, boxed material with colored background running on several pages, the bibliography, the index are standard with any TEX system typesetting engine that uses the LATEX 2_ε mark-up.

3 New commands

It goes without saying that the class mus be called with the command:

`\documentclass[`⟨*options*⟩`]{dictionarySCR}`

This class on turn calls the class `scrbook.cls` so that the latter's commands can be fully used as well as any customization.

All the new and existing options are defined or redefined in this class `dictionarySCR.cls`. The options that do not receive a specific definition are passed on to the inner class `scrbook`; if the latter can use them, that's good; if it can pass them forward to other packages, that's good; if they can't be used by both the class or the called packages, the unused options are silently ignored.

There are very few really new options; possibly the most important one is the `9pt` one, because it is used to fix the default normal font size; the inner class `scrbook` does not have this option, while, had we used the `memoir` class, this option would have been already available. Another new option is `GreekTimes` that is used to substitute the default CBreek fonts with the ones defined by the package txfontsb that probably are a better match with the default Latin fonts provided by the txfonts package. Notwithstanding this feature they are sort of incomplete for what concerns the families, series and shapes available with the CBgreek fonts and for what concerns the specific glyphs that are generally available only with the CBgreek fonts. Most probably the txfontsb are more suited to typeset plain Greek text (with monotoniko spelling) so that unexpected results might be obtained if one uses the advanced commands available with the packages that assume to use the CBgreek fonts. The option `debug` allows the authors to use the trace package for debugging purposes. Even the end user might specify this option if s/he fully knows how to use the trace package and knows what s/he is doing. Last but not least the option `noinputencoding` allows to override the default specification of `latin1` for the input file character definitions. Of course if this option is specified, no input encoding is declared and no national characters can be input; if the dictionary is typeset fully in English, there would not be any need for an input encoding specification, otherwise the end user should specify as usual:

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

and `{`⟨*encoding*⟩`}` should be replaced with the LaTeX name of the desired encoding: `latin1` for the ISO-8859-1 encoding, `utf8` for the UNICODE encoding, `ansinew` when using oldish Windows systems, and `applemac` when using oldish Macintosh systems or when the end user does not want to customize his/her Mac preferred editor to use a more widely used encoding.

There are several language options: `italian`, `english`, `french`, `german`, `latin`, and `greek`; the `latin` option is unlikely to be used; the `greek` option refers to the multi-accented orthography of ancient Greek (below, this issue will be discussed in more detail) but as far as typesetting by means of the typesetting engine *pdftex*, the end user might use both kinds of orthography, single-accented or multi-accented one; it's up to her/him to use the proper diacritics. This unusual set is the one preloaded by default with package babel, but as every LaTeX

user knows, the last option specified to babel is treated as the document main language; in order to overcome this rigidity the above class options will set the specified language as the main one. This mechanism does not work with languages not included in the above list; if for example one desires to typeset a dictionary in, say, Spanish, s/he should insert the `spanish` option in the class call, but should also provide an explicit `\selectlanguage{spanish}` after the `\begin{document}` statement (see more below).

This said, the options that are being used by default are the following: `10pt`, `captions=tableheading`, `captions=nooneline`, `headings=small`, `twoside`, `a4paper`; the `9pt` font size must be explicitly specified if this is the desired normal size. If option `9pt` is specified, this gets executed after the inner class has already set up the `10pt` normal size; afterwards after many alias assignments by means of `\let`, the various size commands get shifted one position towards the smaller sizes; the result is that all size command such as `\large`, `\small`, etc., assume another definition, but the advantage is that a true `\HUGE` command comes into existence.

The packages that are being loaded are quite a few; they are listed in table 1 with their possible options.
The purpose of each package is explained hereafter.

inputenc This package gets loaded *only* if the option `noinputencoding` is *not* specified. Its purpose is to select an encoding option for the input characters if national characters are being used. The default option `latin1` is suitable for all UNIX systems, Mac OS X included. By specifying the class option `noinputencodig` you can change to your preferred encoding. For a dictionary typeset in one of the Western European languages the default encoding is suitable for most machines and operating systems, even with the Windows ones, although some *rare* characters might not be coincident in encoding `latin1` compared with encoding `ansinew`, but they are so rare that the chance of getting one of them is almost naught. Nowadays the `utf8` input encoding is gaining popularity and many LaTeX oriented editors offer this possibility, that is often the default setting just out of the box.

fontenc is used to specify the encoding for the fonts used for the output typeset document; by default this class uses the Times eXtended fonts as specified by the txfonts package with encoding `T1`; this class uses also the Greek fonts, either the collection of the CBgreek ones, or, by specifying the option `GreekTimes` the font collection specified by the package texfontsb, but the end user must not concern him/herself with the Greek font encoding, since this is taken care of by the `greek` option to the babel package. For languages that use the Latin script, the `T1` encoding is a must in order to allow correct and full hyphenations also in those words that contain national characters. We suggest not to change this setting for any reason.

etoolbox is a recent package that, among other features, adds several commands for delaying code execution.

babel is used to typeset one's document according to the customs and typographical traditions of the selected language, be it the main one, or a quotation

Package	Options
inputenc	latin1
fontenc	T1
etoolbox	
babel	german, french, english, greek, latin, italian
fixltx2e	
amsmath	
amssymb	
delarray	
graphicx	
captcont	
xcolor	
afterpage	
eso-pic	
longtable	
paralist	
textcomp	
wasymb	
txfonts	varg
txfontsb	
SIunits	squaren, cdot, binary, noams, derivedinbase, derived, textstyle
trace	
microtype	
framed	
colortabl	
booktabs	
multicol	
marginnote	
marvosym	
imakeidx	
hyperref	colorlinks, linkcolor=red, citecolor=verdeguit, hyperindex

Tabella 1: Loaded packages with their options

in a different language. babel also selects the language hyphenation rules; these on turn must already be embedded in the LaTeX format; it is not a babel duty to code the hyphenation rules into the format file. Control the beginning of any `.log` file typeset with your LaTeX program; if the languages you use are listed in those first few lines, the right patterns are already embedded in the format file. If one or more of your languages do not appear in that list, then examine your TeX distribution documentation and install the language patterns you need.

teubner This package is not actually loaded. Its functionality would have been very useful for typesetting many classical Greek citations or to write down the numbers in the ancient Greeks style; several dictionary entries required this functionality. It would have been useful also for using a particular inclined font that Heinrich would have used for long quotations in classical Greek. He would have used the upright default Greek font for listing a single Greek word o short phrase. For the latter purpose a `\GRD` command was devised, while for the former one `\GRL` was made available. This was the situation until the end of 2010, when a clash appeared because a new collection of Greek fonts was made available by Antonis Tsolomitis and his package txfontsb, that he considered a Greek extension to the package txfonts, in order to use matching Greek fonts with the standard Latin script Times eXtended fonts. This situation required a complete revision of the use of Greek fonts for classical Greek quotations. The solution was the following: (*a*) the teubner package was not loaded any more and its functionality was embedded into this class; (*b*) a class option `GreekTimes` was introduced in order to let the end user chose what Greek fonts s/he would prefer to use; (*c*) several test were implemented in order to supplement the necessary code in either situation so as to use the CBgreek font macros for supplementing the possibly missing glyphs with Tsolomitis' fonts (this happens mostly with ancient Greek numerals); (*d*) if available the teubner accent macros contained in file `lgraccent-glpyphs.def` are loaded in order to allow the use of these macros that repair some glitches with Tsolomitis' fonts.

fixltx2e This small patch is used to "correct" some features of the LaTeX output routine, that is not adequate when floating objects appear in two column typeset documents. This negative feature was observed long ago, and this fix has been available for several years now.

amsmath allows the use of the many extensions programmed by the American Mathematical Society for typesetting advanced mathematics.

amssymb gives access to the multitude of mathematical symbols created by the American Mathematical Society; this package on turn, loads the other package amsfonts.

delarray allows the use of a uniform syntax for specifying various kind of matrices delimited by any delimiter one might wish to use; moreover it extends the functionality of package array, that it loads in case it was not already loaded, so that the end user can access also the various facilities of the latter package, as, for example, the column descriptor `m{...}` and `b{...}`; the defining commands for any new type of column; the left and right column modifiers `>{...}` and `<{...}`.

graphicx is the main package for handling any type of external insertions, in particular picture files; it allows also rotation, scaling and resizing of such pictures.

captcont is particularly useful when a long table is split over several pages; it might be used also for figures, but in this class figures are not numbered.

xcolor handles color definitions in a more advanced way than the usual color package.

afterpage defines the command `\afterpage` the argument of which is delayed to the first available end of page; very often it is used to issue a `\clearpage` command that, if used correctly in two column mode, cleanly empties the queues of floating objects waiting for a positioning onto the output document; by delaying the action of the `clearpage` command, short pages and/or columns are avoided.

eso-pic This package allows to set pictures or other objects as background images on any page or a series of pages.

longtable This important package allows to typeset long tables split across several pages; it is compatible with all other table handling packages, such as array, caption, and the like.

paralist defines a small environment to typeset in line lists, without displaying them as the normal list environments do. It is useful when the list items are very short.

textcomp This package defines more than one hundred commands for typesetting para-alphabetic symbols' it can be used, while in text mode, to typeset, for example, 90°, or №, or µ (the decimal prefix for "micro"), and many other symbols.

wasysym extends the list of para-alphabetic symbols made available by textcomp, among which the symbols for planets and Zodiac constellations.

txfonts As already mentioned in the Introduction, the default fonts for the dictionary are the Times eXtended fonts, made available by loading the txfonts package. This package defines also the mathematical fonts so that math expressions turn out in the same style as the textual parts; the package accesses the eXtended Helvetica fonts for the sans serif family and a variant of Courier for typewriter fonts. The only "drawback" of these fonts is that they come in one optical size that gets enlarged or shrunk according to the necessity: title page fonts, running titles, first and second order math indices, and so on. This technique is used in a variety of technical documents and books; as one may easily verify, this produces thinner then expected shrunk symbols and blacker than expected enlarged symbols; by maintaining the scale factor close to one, this drawback is hardly noticeable. In spite of this feature we chose these fonts because they are narrower than the default TEX system ones, so that typesetting in two column mode yields a better uniform layout of the lines.

txfontsb is the package that gets loaded in order to typesett Greek text with Antonis Tsolomitis' fonts; it requires the option `GreekTimes`. In this case proper actions are performed in order to supplement the missing glyphs so as to use this class even with these Times compatible Greek fonts.

SIunits allows the correct use of SI units of measurement; may be the list of options with which it has been loaded is a little overdone, but it does not hurt. Remember, though, that, since the babel package has already been loaded when SIunits gets loaded, and since one of the default languages is `italian` and this option may get propagated to other packages, two commands are available while typesetting Italian text: `\unit`, defined by the `italian` option to the babel package, and `\unita`, defined by SIunits while in Italian mode. These two commands have a different syntax, so the end user should conform to the specific command required syntax: with `\unit` the information is very compact: you can type 35 km/h by writing in the source file `35\unit{km/h}`; with `\unita` you have to write `\unita{35}{\kilo\metre\per\hour}` in the source file (notice the British spelling for "metre"). It's worth noting that if you like to use the `SIunits` package so as to have available the more descriptive macros for the SI units, but you want to maintain the compactness of the `babel-italian` command, you can do so by inserting at the end of your preamble in your document the following lines:

```
\makeatletter
\DeclareRobustCommand{\unit}[1]{\@inunitcommandtrue
\ensuremath{\SI@fstyle{\@qsk\period@active{#1}}}}
\makeatother
```

By so doing, you actually redefine the `SIunits` command `\unit` so as to use its functionality, but with the `babel-italian` syntax. This means that you may input `35\unit{\kilo\metre\per\second}` that would not work with the `babel-italian` command, but it is not any more the `SIunits` original command. If you want to use here and there the `SIunits` syntax, (while you are writing in Italian, not in other languages) you still have available the `\unita` command that has not been affected by the above redefinition of `unit`.

trace is a very useful tool for debugging and to decipher possible programming errors; this package is loaded only if the `debug` option has been specified. Reading the `.log` output file when trace is in action requires "strong hearted" users. The very use of this package requires the user to be aware of what s/he is doing. If the tracing commands are used in an improper way, the compilation time may increase from a few dozen seconds to several minutes and the `.log` file might became so huge that it may clog the hard disk. So, please, don't use the tracing commands unless you know what you are doing!

microtype is very useful when typesetting with *pdfLaTeX* and while using suitable vector fonts. The default Times eXtended fonts are such suitable fonts

so that microtype develops its full power, that is to protrude certain small punctuation marks or parts if the font serifs into the side margins; to slightly extend or shrink horizontally each whole line, spaces included, in order to properly justify every line, without stretching the inter word space too much. The result is the almost complete elimination of hyphenated lines and a better appearance of each paragraph, where any possible white rivulet is eliminated. This is very useful in a two column typeset dictionary.

framed extends the possibility of framing and coloring some text compared to what can be done with the standard `\fbox` and `\framebox` commands.

colortbl allows the construction of colored tables.

booktabs gives the end user the facilities necessary for typesetting professional tables, by means of certain horizontal lines of different thickness and suitable distance from the table cell contents, so that formulas and capital letters do not butt any more against the standard horizontal lines provided by LaTeX standard `tabular` facilities.

multicol differently from the `twocolumn` mode of LaTeX typesetting, allows to typeset one's test in two or more columns without automatically starting a new page and by dividing its columns in such a way as they have the same height.

marginnote allows a simpler way to specify marginal notes compared with what can be done with the standard command `\marginpar`.

marvosym is another package for typesetting a certain number of analphabetic signs, such as identifying symbols for the planets, the Sun, the Moon. Some of these signs are similar to the ones provided by the wasysym package, but there is no conflict with their commands.

imakeidx is loaded so as to compile the index synchronously with the main text; it also allows several customizations and the production of several indices.

hyperref must be the last loaded package because it redefines some commands that were previously defined by the classes, especially the inner class `scrbook`, and or the previous packages. Its options specify that their links are colored ones, not boxed links. Links to internal destinations are red; citation links are dark green, and that the index entry pages must be linked to the pages where the entry was used.

It's important to take into account the fact that such packages are preloaded before deciding to load other packages. Since the preloaded ones are preloaded by means of the system command `\RequirePackage` the typesetting engine knows exactly which packages are already in memory and it will refuse to reload any of them; it will silently ignore a second loading request. This also avoids to reload the same packages with different options. In spite of this it is correct to use the `\documentclass` option list to pass options both to the inner class and the called

packages that use the facility of using the class global list option. Such options arrive to the called packages in addition to the options that are already specified. For example, if one wished to typeset a dictionary, say, in Spanish, s/he should not reload babel with the `spanish` option, but s/he should specify this option to the class; babel will retrieve this information from the class global option list; the user shall just issue the `\selectlanguage{spanish}` just after the `\begin{document}` statement. Since graphicx does not retrieve its options from the class global option list, provisions are made to pass the `draft` option also to the this package.

The end user who wants to exploit the loaded packages is invited to read their documentation in order to use them at their best.

4 Default fonts txfonts and compatibility issues between Tsolomitis' fonts and the *CBgreek* ones

We repeat here some information that was previously given; this in order to emphasize the difficult compatibility issues that arose when Antonis Tsolomitis uploaded his Greek fonts; he intended his fonts to match in a better way the Latin ones provided by the txfonts package; in effects they are a little blacker than the default CBfonts ones, that were designed to match the standard TEX system default lighter fonts Computer Modern, European Computer Modern, CM-super Latin and Cyrillic fonts, Latin Modern. The Times eXtended are more compact that the above mentioned fonts and should be accompanied by another compact Greek font; Tsolomitis's fonts meet this goal and a text with mixed Latin and Greek script does not exhibit any change of "text color" in conjunction with the change of alphabet.

There are two or three problems with the presence of the font collection created by Tsolomitis:

1. the CBgreek fonts were produced for philological works and contain a number of glyphs that were important for typesetting some symbols in ancient Greek; those concerning the numerical notation, and some symbols present in ancient calendars. Such symbols lack from the fonts by Tsolomitis.

2. The use of the tilde to insert an unbreakable space in Latin scripts conflicts with the need to use it as a circumflex accent in Greek; this problem was solved with the CBfonts working in tandem with any Latin font collection, but is not solved with Tsolomitis' fonts.

3. The font description files have strict naming rules; they must be formed by the agglutination of the lowercase encoding name and the lowercase family name; Greek typesetting with the default Latin and CBgreek fonts implies font description files named `lgrcmr.fd`, `lgrcmss.fd`, `lgrcmtt.fd`. If the Latin TX fonts are used, the font description files are named `lgrtxr.fd`, `lgrtxss.fd`, and `lgrtxtt.fd`. In both cases changing alphabet simply means changing the encoding name from `T1` to `LGR`, while the family names remain the same. Tsolomitis *had to use the same names* with his fonts,

therefore with the Times eXtended Latin collection the font description files for the CBgreek fonts became mutually exclusive.

4. Tsolomitis was very attentive to map the vector fonts to the LGR encoding in order to correctly and routinely write in Greek with the monotonic spelling; he uses his fonts routinely for writing his textbooks for his university students; but these books are typeset in modern monotonic Greek spelling and this point is the one that was most important for Tsolomitis. In facts, with polytonic ancient Greek his fonts do no behave properly when some diacritic sign combinations are used; moreover the various font tables (upright, oblique, etc.) do not contain the same collection of glyphs.

5. Last but not least, Tsolomits vector fonts were available in the TEX Live distribution, but initially they were not available as preinstalled vector fonts to be used by *dvips*, *pdflatex*, etc.. To let these programs use the vector fonts, these must be listed in the respective map files `dvips.map`, `pdftex.map`, etc. With no previous error or warning message the typesetting engine would look for the source `.mf` files, but could not find them; therefore the program crashed miserably. Now it is not too difficult to find out what must be listed in a specific map file, nor how to add this information to the system wide map files, but it is very delicate and beyond the possibilities of most LATEX beginners. This problem was eventually solved, but the above incompatibilities remained. Fortunately enough, after informing Tsolomitis, this final glitch has been removed and now Tsolomitis fonts are usable by simply loading the txfontsb package, and the end user needs not do anything special; if s/he typesets in monotonic Greek s/he might usefully specify the option `iso-8859-7` to the inputenc package; this encoding option directly maps the Greek keyboard Greek glyphs to the `LGR` encoded Tsolomitis' fonts. But of course this has nothing to do with typesetting dictionaries containing ancient Greek quotations.

Before the availability of Tsolomitis' fonts, this class used the teubner package; in particular this teubner package was completely revised at the beginning of 2010, some six months before Tsolomitis' contributed his txfontsb package.

Due to the incompatibilities described above it was necessary to find a solution. This was found by avoiding to load the teubner package, and by copying the relevant macros directly in this class; moreover this class is programmed to make the necessary test to check if the `GreekTimes` option was specified, if Tsolomitis fonts are available, and in each case to perform the necessary actions so as to avoid every possible clash, while, in case, supplementing the missing glyphs if Tsolomitis fonts are being used; also the teubner package definition file `lgraccents-glyphs.def` is exploited in order to correct the missing diacritic combinations in Tsolomitis' fonts. We think we have succeeded in patching this uncomfortable situation for which nobody is to blame; it is simply hard coded into the very guts of LATEX new font selection scheme NFSS.

Nevertheless, in spite of these efforts, the recent changes (September 2013) that have taken place in the Greek language handling, raise new incompatibilities

between the CBfonts and those by Tsolomitis; the code remains, but Tsolomitis fonts *should not be used if Greek numerals are being used in the dictionary.*

5 New and modified commands

It must be noted that this English version of our initial `dizionarioSCR` class has been slightly extended; most new commands in the original class had Italian names; this "international" version has an English alias for every Italian command, so that we even avoid listing the Italian commands. The internal definition code might retain Italian temporary scratch names, but this is of no concern for the end user.

Also some infix Italian words are changed into English ones; corresponding infix words are provided for the other preloaded languages, so they are the right ones (we hope) for every dictionary main language option; instructions are given to localize this class in other non predefined languages.

`\chapter` `\section`

Macros `\chapter` and `\section` certainly are not new commands; but have been redefined as follows. `\chapter` is used to start a new "chapter" when starting a new alphabetic section; it is also used to set the thumbnail in the outer margin. But, although it starts a numbered chapter, its number never appears in print. Similarly the `\section` command has been redefined as to typeset its running title in line with the following text, and it never prints its number. The end user must never use it directly but s/he should use the next command `\entry` that uses it in the background. These redefinitions were made necessary in order to comply with the inner workings of the hyperref package that needs unique references in order to perform correctly.

To this purpose it's worth noting that, if after editing the source files by modifying or adding other entries, it may happen that the compilation issues menacing messages; they arise with the contrast between the new information collected by hyperref and the information collected in the previous run; just press the Q or q keys at the program prompt and keep going; the next compilation will be free of errors. Alternatively delete the main file `.aux`, `.log`, `.out` files and start again with at least two runs.

`\abovesectionskip` `\belowsectionskip`

In a while we specify why the entries are entered with a new section-like command `\entry` Actually this section-like command is a renamed and enriched version of the original `section` command. this command requires a fixed skip above the section heading of 3.5 ex and another skip after the section heading of 2.3 ex. We redefined the `\section` command so as to allow parametrized values that can be set by the typesetter; these values remain the default ones, but we think it is very interesting to change them to the new values:

```
\abovesectionskip=0pt plus 0.5es
\belowsectionskip= -\belowsectionskip
```

The dictionary appears very compact, although the entries are distinctively emerging from the remaining text with their bold sans serif shape. Even in two column mode the dictionary columns appear well balanced on every page, and for what

concerns the column balancing at the end of every chapter it seems to be much easier to achieve decent balancing by means of the `\balance` macro (see below).

`\entry` Every entry is introduced with command `\entry`; the syntax is:

`\entry[`⟨*entry-s*⟩`]{`⟨*entry*⟩`}`

where `{`⟨*entry*⟩`}` is the real entry name complete with all "ornaments" (diacritical marks and other) necessary for its correct spelling and, possibly, information in other alphabets, that in astronomy take place quite often, for example "Carinae, η"; on the opposite `[`⟨*enty-s*⟩`]` is the simplified entry name from which all ornaments have been stripped off and where parts in other alphabets are transliterated with Latin letters, for example "Carinae, eta". If the optional argument `[`⟨*entry-s*⟩`]` is not specified, it is assumed coincident with the real entry name, and this is acceptable only if the latter does not contain accented letters and/or parts in other alphabets. This is because the plain entry stripped form any non ASCII character become the arguments of the internal hyperref macros used by the package to construct the hyperlink anchors and targets.

`\seeentry` It is possible to make a reference and a link to another entry by means of this macro: the reference label must coincide with the referred entry optional argument. If you enter into the source file something such as `\seeentry{Abbe}` you not only produce a reference to the other entry name "Abbe", but also create an hyperlink to to the other entry so that, when reading the dictionary on a computer, you can directly point and click the mouse in order to be moved to the other entry location.

`\equref` You use this macro as the corresponding macro `\eqref` defined by the amsmath package or the ordinary LaTeX `\ref` command. Its argument is the label tat was assigned to the referred equation; obviously every equation must have a unique label along the whole dictionary. The new command `\equref` accesses the 5-argument label produced by hyperref into the `.aux` file, and extracts the entry name and the kind of hyperlink reference. It writes out something such as "... equation 2 of lemma Abbe...". You can click over the string "equation 2" in order to move the focus on equation 2; or you can click on the string "Abbe" in order to be moved to the start of the description of the Abbe entry.

`\autoref` Although it is not a new command, but a command defined by hyperref, this one defines an anchor for a link (that is the place where to click for moving the focus to the link target) as a string formed by the name of the object and its number; at this stage we defined only "equation" and "entry" (actually *equation* is already defined by default, but we needed another definition in order to localise this class for Italian documents). This implies a certain attention in writing the text that precedes the anchor, since the name is already contained in the generated string; when citing an equation it may appear strange to write "`... see \equref{myequation} ...`", without using the word "equation", but such English usage is less striking compared to the wording on languages that use articles; for example in Italian: "`... vedi l'\equref{myequation} ...`", where the elided article `l'` implies a following noun starting with a vowel; actually this noun gets typeset by the `\equref` command.

\drop This macro is used to set a versal; its syntax is:

`\drop{`⟨*letter*⟩`}{`⟨*other text*⟩`}`

where `{`⟨*letter*⟩`}` is the versal and `{`⟨*other text*⟩`}` is formed by the test that follows the initial versal; this text is typeset in small caps; it may consist even of a single letter, that is taken as a reference for scaling and lowering the initial versal. The macro is an adaptation of another one written by David G. Cantor for use with plain TEX, and adapted to LATEX by Dominik Wujastyk.

\balance By default the two column typesetting mode does not balance the columns in the last page of a chapter. The new command `\balance` can achieve this goal, without using the balance package, that has restrictions on the contents of the page to be balanced: it shouldn't contain any floats. This command `\balance` is much less sophisticated; it needs to be used by hand by the typesetter with a cut-and-try approach if and when s/he wants to balance the columns in the last chapter page. Notice that there should not be any floating object declaration *after* the `\balance` command, otherwise columns do not get correctly balanced. But we assume, in any case that slightly unbalanced columns are acceptable and the `\balance` command gets used only when the second column is very short.

In order to use `\balance` the user must keep in mind that this command just inserts (through a "whatsit") the equivalent of a `\newpage` command in the very point where the command is used; in two column mode `\newpage` means "new column": a negative penalty is inserted in the very point where the `\balance` macro is used, and when the output routine composes the columns and the page, it finds a good place to break the column just after the line that contained the "whatsit", so that the line justification is not altered, but the column gets split at that point. In order to work correctly non floating object should follow the `\balance` macro, otherwise the output routine gets confused; and in any case it is not that important that the facing columns have exactly the same height, but it is important that the facing columns are not too unbalanced. This is where the cut-and-try approach to find the best position for the `\balance` macro come into play and is the user's responsibility to find the best place to break the columns by allowing a little bit of unbalance, but avoiding, for example, widow lines.

\versal On the opposite the single argument command `\versal`, with the syntax:

`\versal{`⟨*letter*⟩`}`

typesets `{`⟨*letter*⟩`}` as a versal, but it does not need any further text string. Long ago this command was defined in package versal which is not available any more; the code defined in this class is a special application of the above command `\drop`.

\scaption This macro is e generic one for typesetting a caption without any reference to the object name and its number, although it steps the referable counters and anchor and target links are uniquely made so as to refer to the object with the usual `\label`, (`\ref`) and `\pageref` commands. The next commands are built on top of this one, but are enriched with other information, so that the end user should rather use the latter ones.

\scaptiona \scaptionb Both macros are used to typeset a non numbered caption just as `\scaption`

does, but the first one should be used to typeset its caption above the object, while the second one does it below the object (the ending 'a' and 'b' remind *above* and *below*); moreover a visible arrow tip gets printed before the caption in order to point to the object the caption refers to.

It must be remarked that all three commands accept the regular LaTeX sectioning command syntax, as well as the standard `\caption` command: a first optional argument and a second obligatory one; the short caption optional argument is not used to typeset the list of figures or the list of tables, but is processed by hyperref to create a target for the hyperlinks. As it was recommended for the `\entry` arguments, this short caption should be used for writing a text string without special characters or changes of alphabet, while both special characters and alphabet changes can be freely used in the long caption argument. The end user is strongly invited to make a frquent use of this optional argument.

`\upperline` `\lowerline` `\mttile` `\mtsubtitle` `\mtssubtitle`

The first four macros are used to typeset a sort of mini table of contents (minitoc; hence the first two letters of the last three macro names); actually the first two are internally used by the environment `minitoc` that will be described shortly. The other macros are used for entering the necessary information by the end user. The `minitoc` is used to announce the contents of some complicated and long entries, so the reader can find the relevant point of the entry description in a more comfortable way. The `minitoc` is separated from the entry title and, possibly, first paragraph, by a horizontal line, as well with another horizontal line after it is finished. `\mttitle` and `mtsubtitle` are used to enter the entry section and subsection titles, while `\mtssubtitle` is used only in the body of the entry description in order to further section a long subsection in the entry description. Such commands will be repeated by the end user in the body of the description. Nothing is typeset in an automatic way; its up to the end user to make a proper use of these commands.

`minitoc`

In spite of the fact that the end user must insert the necessary commands in the `minitoc` and must repeat them in the body of the entry description, at least the environment `minitoc` takes care of the separating horizontal lines and of the necessary spacing.

`\degr`

This macro is substantially an alias to both the `\textdegree` macro defined with the textcomp package, and the `\degree` macro defined by the SIunits package. The advantage is that you can use it both in text and math modes and it behaves differently from the original package macros: in text mode the `\textdegree` macro is used, while in math mode it is represented by a tiny circle, one of the math symbols, in the position of an exponent. See below the comments regarding the the symbol for the Celsius temperature.

`\diff`

This macro is used to typeset the differential symbol according to the ISO regulations and with proper asymmetrical spacings on either side.

`\possiblysortindexfiles`

This macro "possibly" exploits the modern TeX system feature to open a shell and execute system commands. This feature is enabled if the typesetting program is launched with the option `shell-escape` or `enable-write18`, depending on the TeX system distribution. It is not important whether the launching command is given in the command line window, or if it is embedded in the string used by the shell editor to run the typesetting program; the end user might prefer to do it

by hand instead of having the shell editor always set up for opening a shell. If this feature is enabled, then this class will close the index output stream, open a shell and launch the *makeindex* program and, when this is finished, it resumes and inputs the sorted and formatted index file. If the shelling feature is not enabled this macro does nothing, and the sorting program must be run by hand as it used to be since the introduction of LaTeX in 1984.

`\printindex` This is the macro that actually imports the sorted index file and typesets it. If the previous macro was allowed to operate, then the typeset index is always in sync with the typeset text. Otherwise `\printindex` will just typeset the `.ind` file, if there exists one, or just issues a message that no sorted index file is available. Of course in case the shelling feature was not enabled, it's the end user responsibility to run *makeindex* as it was necessary with older versions of the TeX system.

`\cleardoublepage` This command has been redefined in order to accept an optional argument that is used to specify the page style of the blank page that possibly might be printed in order to restart typesetting on an odd numbered page. The default style is `plain` so that the blank page actually reports its page number; the user might specify the optional argument `empty` and the blank page is output completely white; should s/he have defined other page styles in addition to `headings` and `myheadings` but more suited for a blank page, s/he can specify any other custom style it is fit to this situation.

`\pointsto` `\longpointsto` Commands `\pointsto` and `\longpointsto` are used to point to other entries; thesecommands are added to the functionality of `\seeentry` in that they emphasize the necessity of reading another entry in order to fully understand the current entry. If the dictionary is typeset "on paper" and without colored links, the relevance of the link might be lost if these pointers were not used.

`\insertremark` The macro `\insertremark` is used to insert an incidental remark surrounded by en dashes that are separated from the remark contents with unbreakable spaces; the usage is therefore:

`\insertremark{`⟨*parenthetical remark*⟩`}`

`\SetTableIntoExternalColumn` This long named macro tries to typeset tabular material so that it may hang into the external margin; it must be used with great care. Its syntax is the following:

`\SetTableIntoExternalColumn{`⟨*didascalia e ambiente tabular*⟩`}`

As it can be seen the only obligatory argument is rather complex, since it includes both the table caption and the tabular material. This command can replace the `table` environment in this sense: You first typeset the tabular material with the preceding caption within a regular `table` environment. If the tabular material turns out to be wider than a column but not so large as to require a `table*` setting, then the opening `table` statement may be replaced by this macro name with its opening brace; while the closing `table` statement is replaced by the matching closing brace.

Be aware that such table is not a floating object any more and it's typeset at the very point where it is declared; but it must hang into the external margin,

so that the macro controls the parity of the page number where it falls, so as to set it hanging to the left on left pages and on the right on right pages; it guesses the parity at the first run, but it has the correct parity at the second and further runs. Unfortunately the asynchronous way used by the typesetting engine to split a page into columns is such that it's impossible to know, while the source code is keyed into the source file, if that table will fall in an internal vs. an external column of a given page; therefore if such a table is in the left column of a right page it will hang to the right and will overlap the right column. Non problem, but it cannot be handled in an automatic way and it's necessary to edit the source file so as to move it some paragraphs forward in a right page and some paragraphs backwards in a left page. We repeat: no problem, but it requires some cut and try approach that is generally to be avoided with the LaTeX mark-up.

Remember that such a table is not floating; if the caption is numbered it may happen that its final position is before another floating numbered object that is floated after or before this fixed table. This is an intrinsic drawback of mixing floating and fixed figures or tables.

`Wtabular` If the method of typesetting a hanging table does not give good results, the typesettr might usa the next environment that produces a similar typeset table while keeping it a floating object. Some manual tune up is still necessary, but some inconveniences are avoided. This new environment has a syntax very similar to the `tabular` one, except the first optional argument is not used at all. This helps in patching a slightly oversized table, in the sense that it suffices to change the `tabular` environment by adding a capital `W` in front of the `tabular` name. The syntax is the following:

```
\begin{Wtabular}[⟨not used⟩]{⟨column descriptors⟩}\\
⟨rows⟩\\
\end{Wtabular}
```

The caption, if needed, must be set before the `Wtabular` environment as usual. Remember: it's the `table` environment that floats this `Wtabular` one and that specifies which caption prefix to use.

`shadedtabular` `shadedWtabular` Two other environments similar to the `tabular` and `Wtabular` ones are available for typesetting tables with shaded rows or columns o single cells. Actutally they do not do anything by themselves, they are simply enclosed in a color frame environment and the user must specify the actual colors and the objects to be colored. The syntax for these two environments are the following

```
\begin{shadedtabular}[⟨background⟩]{⟨column descriptors⟩}
⟨table row⟩\\
⟨table row⟩\\
...
\end{shadedtabular}
```

```
\begin{shadedWtabular}[⟨background⟩]{⟨column descriptors⟩}
⟨riga della tabella⟩\\
⟨riga della tabella⟩\\
...
\end{shadedWtabular}
```

Both environments accept a first optional argument intended to set the background color. By default this color is "straw". The user can use such option to specify a different color defined by him/herself or chose among the predefined colors of this class – see below.

`\textLipsias` `\GRL` `\textDidot` `\GRD`

These four commands are used to typeset Greek text; `\GRL` is a shortened version of `\textLipsias`, while `\GRD` performs in a similar way for `\textDidot`. The shortened commands limit themselves to selecting the `LGR` encoding and the correct font shape; therefore they are suited to typeset a single word or at most a short phrase; hyphenation remains the main language one, therefore it might not be the desired outcome; one might use the standard discretionary hyphen command`\-`. In general the availability of the shortened commands is justified by the ease of their usage; the long commands do their work in the proper way but suffer of other idiosyncrasies, such as producing strange errors if used in the entry title; in spite of being robust commands sometimes they break up in the entry long title; of course the optional entry short title is there on purpose in order to avoid Greek text in the hyperlink anchors and targets, but sometimes the long commands surprise the user with an unexpected break up.

Thanks to the `polutoniko` attribute specified to the `greek` option of the `babel` package, it is possible to use the full functionality of the `LGR` encoded CBgreek fonts. Nevertheless, should there be any clash with active characters, especially the tilde, it's possible to use the extended accent macros that are defined in the `lgraccents-glyphs.def` file; this file accompanies every recent complete distribution of the TEX system (since 2010) and the typesetting program can fetch it without problems; should it be absent, please either update your system or, at least, update the `teubner` package, since that file is an accessory of that package. Such extended accent macros form the same accent sequences used in "normal" usage of the `LGR` encoded fonts, but such sequence is prefixed with a backslash; for example, should `~>a` produce strange results, change it to `\~>a` and everything will be fixed; should `>'eureka` turn out without the ligature with the acute accent, change it into `\>'eureka` and everything will work as expected; this is (was?) an actual possibility when slanted Tsolomitis fonts are used.

The two commands `\textDidot` and `\textLipsias` behave as they should with correct fonts and Greek hyphenation rules; the former uses the upright font inspired to the Didot shape, and the latter uses the slanting font inspired to the one designed in Lipsia and used by the Teubner Printing Company. The commands' syntax is the following:

```
\textDidot{⟨Greek text⟩}
```

```
\textLipsias{⟨Greek text⟩}
```

Let us underline that the {⟨*Greek text*⟩} may consist of several paragraphs.

`catalogs` Eventually here we have some macros and environments in order to typeset some material in one column mode with different headers, and general set up. The name "catalogs" comes from Henirch's need to insert Stellar Catalogs, that is, long and large tables full of information concerning every listed star. In many other thematic dictionaries similar material might need to be included. Hereafter we'll refer to this material as a "catalog".

We assume a catalog is fundamentally a long an large table that requires several pages to be typeset. There are a multitude of long rows of cells; without using horizontal rules, that are considered a poor practice in typography, we nevertheless need something to lead the reader's eye along each line; we decided to use background alternating shading for each row.

Therefore these catalogues are typeset by means of the environment `longtable` on one column mode; since the dictionary is typeset in two column mode, we need to start the `catalogs` environment by memorizing the previous typesetting mode in order to be able to reset it upon exiting the environment, because this appendix-like environment may be followed by the bibliography and/or an index of, say, authors, persons, devices named in the dictionary without a specific entry, it's important to reset the typesetting mode, so that the back matter component are not confused by assuming a non existing mode.

One or more catalogs shall therefore be inserted in the input stream within a `catalogs` environment, intended to set and reset the typesetting mode, specifically to set one column mode and saving the necessary information in case the outer typesetting mode has to be reset. This environment resets the headers style so as to avoid horizontal rules that might be mixed up with tabular rules; in addition the headers contents is not predefined so that each catalog can produce any header contents the author considers it fit.

`\insertcatalog` It is convenient that each catalog is entered by means of an ad hoc source file that, besides specifying a title for the main section, contains some information on the contents of the catalog itself and, perhaps, some information to the reader on how to consult the catalog. Command `\insertcaltalog` performs this insertion; it requires two compulsory arguments and an optional one. The optional argument is blank/empty by default; any non empty entry (its arbitrariness may be solved by specifying `color`) will specify what to do to shade the background, and the single lines. No optional argument, implies a white background and no shaded alternate rows. This "colorless" is more suited for a short table and few columns, where the reader's eye does not require any help for reading the table.

The compulsory arguments are the name of the file to be input and the header contents for this particular catalog pages. Any inserted catalog starts on a new page.

IMPORTANT NOTICE! the main sectioning takes place through the command `\chapter*` but must be followed by the `\thispagestyle{plain}` *compulsory* statement because this class defines a particular page style that is functional for a dictionary; this compulsory specification is important in order to avoid complicated page style redefinitions.

The control file shall take the following form:

```
\begin{catalogs}
\chapter*{Stellar catalogs}
\thispagestyle{plain}
Hereafter there are a few some catalogs that ...

\insertcatalog{smith}{Smith's catalog}
\insertcatalog[color]{Brown}{Brown's catalog}
...
\end{catalogs}
```

Each catalog file is loaded within a group; therefore the imported file can contain local definitions that change for that particular catalog some typesetting parameters; this grouping allows to compose a number of catalogs that are stylistically different from one another.

Notice that for shaded catalogs there are already available the definitions of certain shading colors and a number of new column types, so as to allow to specify the row or column colors; see table 2.

Commands to set table row colors are as such:

```
\rowcolor{lightyellow}
```

where the color name has already been defined toghether with the other colors with the color specification:

`\definecolor{`⟨*color name*⟩`}{`⟨*colr model*⟩`}{`⟨*numerical parameters*⟩`}`

Similarly the cell color is defined by commands such as:

```
\cellcolor{straw}
```

These colors may be redefined; in the sense that the corresponding commands may be redefined. Even better, it is possible to create new definitions and to `\let` `\RCeven` and `\RCodd` to be aliases to the new definitions. For example, this class defines:

```
\def\RCiv{\rowcolor{paglierino}}% "straw"
\let\RCeven\RCiv
```

(don't bother about the Italian name for the "straw" color; any other name would do as well). This trick might be very useful for customizing shaded tables and catalogs.

Struts must be placed by hand inside any cell that is preceded or followed by `\toprule`, `\midrulle` and/or `\bottomrule` in order to avoid white strips at the rule sides.

When a long table gets split on several pages it would be desirable to set the necessary struts in the last row of every fraction of the table, as well, probably

Comando	Note
`\tabcoolsep`	redefined to 5,6 pt
`\RCi`	row color cyan
`\RCii`	row color light yellow
`\RCiii`	row color light orange
`\RCiv`	row color straw
`\CCyellow`	cell color yellow
`\CCYellow`	cell color saturated yellow
`\RCodd`	odd rows color (default: `\relax`)
`\RCeven`	even row color (default: `\RCiv`)
`\CC`	cell color (default: `\CCYellow`)
`L`	column descriptor with light yellow background and left aligned contents
`R`	column descriptor with light yellow background and right aligned contents
`C`	column descriptor with light yellow background and centered contents
`M`	column descriptor with light yellow background and left aligned math contents
`A`	dcolumn descriptor with light yellow background and right aligned right ascension
`B`	column descriptor with light yellow background and right aligned math contents
`\Tstrut`	strut that overhangs 2 pt on top of the default strut
`\Bstrut`	strut that overhangs 2 pt below the default strut
`\TBstrut`	strut that overhangs 2 pt on top of and below the default strut

Tabella 2: Default definitions for typesetting color shaded tables

in the first line of every fraction. To our best knowledge this is not feasible in an automatic way, not even within the longtable package and its environment, so the user must do it manually. Moreover the rules defined with `\toprule`, `\midrule` and `\bottomrule` defined by package booktabs are built up with the necessary spaces that in colored tables remain uncolored and the table colors remain interrupted.

This iw why it's necessary to manually insert command `\CC` at the beginning of every cell built up with a `\multicolumn` command; and why it's necessary to manually insert commands `\RCodd` or `\RCeven` at the beginning of every first row-cell of every colored table with alternate color rows.

The new column descriptors have been defined by means of the package array command `\newcolumntype`. It is possible to define other column descriptors; they must consist of only one letter but it must be different from those already defined, that is `l`, `r`, `c`, `p`, `b`, `m`, already defined in that package, `d` defined by package dcolumn, and `L`, `C`, `R`, `M`, `A`, `B`, defined in this class.

In order to color a table with alternating rows it's possible to use the commands provided by the already loaded package xcolor; we did not exploit these commands dedicated to tables, but the user can act independently.

\RA The Stellar Catalogs report for every star its right ascension; this angle is traditionally given in (sidereal) hours, minutes and seconds, instead of degrees, primes and angular seconds; expressing the tabular contents in the proper ISO way is time consuming for the end user; therefore this class provides a macro that requires just to specify the numerical values: the hour without units, a compulsory space, the minutes with an optional fractional part followed by another compulsory space. The syntax is the following:

`\RA`␣⟨*hours*⟩␣⟨*minutes,tenths*⟩␣

where the symbols ␣ display the compulsory spaces, therefore it's important not to forget any of them, especially the last one. The `\RA` macro typesets the right ascension with both hours and minutes with a leading zero if their value is lower then 10; the fractional minutes part is always typeset, even if it was not specified nor was specified the decimal separator; should the fractional part be absent, then a decimal separator and a single zero is appended. The decimal separator in input can be either a comma or a point; on output is the proper sign according to the language chosen.

Furthermore remember that the ⟨*hour*⟩ and ⟨*minutes,tenths*⟩ values (with or without the optional `,tenths` part) must not be completed with the units 'h' and 'min'; it is the macro's duty to insert such information according the ISO regulations.

This macro may be used as a "declaration" within the column modifiers, but it must be the last element in that list; if one wants the right ascension properly typeset right justified in its column, it can use the new special column descriptor `A` (that correspond the the definition `>{\RA}r`), but attention must be paid to the fact that the input data must consist of two numerical strings separated *and followed* by one space before the `&` end of cell sign, and the `\\` end of row command.

This macro is a little peculiar because it must parse the data strings and this analysis requires what Knuth in his TEXbook calls "dirty tricks".

One might ask why a similar command was not introduced for typesetting in the proper way also the stardeclination; may be, if the need arises, it will be done in the future. By now we skipped this definition because the information concerning angular degrees, primes and seconds do not require as much information as that needed for hours, minutes and minute tenths for the right ascension. The declination is always made up with three pieces of information, but those concerning angles may be input without leaving the *pdftex* math mode and they are formed with few letters: compare a declination string such as `37\degr25'48''`, that gets typeset as $37°25'48''$, with a right ascension input string such as `15\unit{h} 24.5\unit{min}` that turns out as 15 h 24.5 min, with unbreakable thin spaces between numerical values and the respective units of time.

6 Typesetting suggestions

6.1 End of line hyphenation

Typesetting in two column mode sets forth some little problems that don't manifest themselves in one column mode, or if they do, they are so small that remain unnoticed.

One of these problems is line breaking with hyphenation. If a word cannot be hyphenated for any reason, the typesetting engine *pdftex* tries to typeset the paragraph as best as it can, by adjusting the inter word space so as to minimize the overall paragraph *badness*. This badness is a numerical internal parameter that *pdftex* computes according to a sophisticated algorithm that takes into account the ratio of the stretch or shrink of the available "rubber" spaces, the penalties expressly or implicitly inserted by the class and style macros, the presence of consecutive hyphenated lines, and other elements.

When a document is typeset at full text width, lines of text are sufficiently long so as to contain enough inter word stretchable and shrinkable space that allows the paragraph to be fully justified at both sides, without producing inter word spaces too wide or too tight; in any case, among all possible line breaking points in a paragraph with or without hyphenation, *pdftex* chooses the one that yields the least amount of badness.

It might happen that a long word falls by the end of a line, but that the word can't be hyphenated for a number of reasons; the most common reason is that it gets typeset with a font where hyphenation is inhibited. By default LaTeX uses just one such font, the typewriter type font, and this feature is imposed on purpose, because this font is mostly used to typeset stretches of various programming languages, where it is absolutely forbidden to hyphenate the language reserved words and the program variables.

An insidious case takes place when a phrase is emphasized by means of the `\emph` command that produces the desired emphasis by changing the phrase font: *the first or the only word after a font change cannot be hyphenated.*

If this non hyphenated word hangs by one or two points outside the paragraph margin, the typesetting in not perfect, but the deficiency is noticeable only to a trained eye. When typesetting in one column this deficiency is hardly noticeable by anyone, but when the text is being typeset in two columns this deficiency remains hardly noticeable if it involves the right margin, but if the word hangs into the (narrow) inter column space the deficiency appears in all its evidence. If the word hangs outside the line margin by a larger amount the deficiency is disturbing in any case.

What to do? To this purpose the "discretionary breaks" are available; by default LaTeX offers only the command `\-` that may be used to mark a valid hyphen point, but inhibits any further hyphen point in the remaining two word fragments. If a word such as "interstellar" has to be divided, by writing `inter\-stellar` in the source file, LaTeX would break the word only in that marked break point, but it would not divide in any other point.

While typesetting in Italian with the `italian` option to the babel package, there is another possibility, that is to use the special active character `"` for marking a possible hyphen point without inhibiting most of the other possible hyphen points; if we continue the preceding example, and we introduced `inter"stellar` in the source LaTeX file, and it could not be hyphenated because it was the first one after a font change, then the word second fragment could be hyphenated any way; the right strategy would be to mark with `"` the first possible hyphen point and to leave the longest second fragment, so that after a font change `in"terstellar` would display the whole set of hyphen points: "in-ter-stel-lar".

This solution might be used also to mark a possible hyphen point in an emphasized foreign word after the change of the emphasis font, but while within an Italian context; for example, if `\emph{American}` falls at the end of a line end hanging visibly outside the column, sometimes even overlapping the next column, then several double quote `"` signs might be used in order to mark the American syllabification: `\emph{A"mer"i"can}`, so that two goals are reached: the word is actually hyphenated and in a correct way.

But if the document is being typeset in a language different from Italian, or a quotation is being typeset with another language setting, you might look up on the babel documentation; it is possible that also in that languages there are shortcuts for inserting discretionary hyphen points as in Italian; the shortcut might be different, for example `"|`, but almost all languages but English, have available similar facilities. If no shortcuts exist in that language, then it's necessary to use the default command `\-` and mark all possible hyphen points in a given word.

6.2 Figures and tables

It may happen that a figure should have a legend, or a caption that contains a table. Within the `figure` environment a table may be typeset within the caption; but if the List of Figures should be typeset (unusual in a dictionary that may contain a large number of pictures), it is recommended to use the optional argument of the various caption commands, for example:

```
\begin{figure}
\centering
\includegraphics[width=\linewidt]{figure}
\caption[Special image]%  or \scaptionb
{Special image.\newline
  {\begin{tabular}{...}
  ... \\
  \end{tabuar}%
  }%
}
\label{fig:...}
\end{figure}
```

It is convenient to wrap the `tabular` environment within a group so that is clear to the typesetting engine that the various commands `\\` mark the ends of the tabular rows, and have nothing to do with new lines in the caption.

Alternatively a picture can be inserted within the table itself, for example:

```
\begin{table}
\captionabove{Immagine particolare}\label{...}% or \scaptiona
\centering
  \begin{tabular}{...}
    \multicolumn {...}{c}{\includegraphics[...]{...}}\\[1ex]
    \hline
    ...\\
    ...\\
    \hline
  \end{tabular}
\end{table}
```

The image width must be carefully specified so that the whole table does not hang outside the column.

Which approach is preferable depends on the image and on the tabular material. But we wanted to recall some simple LaTeX techniques and to remind that tabular material can be typeset within a `figure` environment, as well as an image can be inserted within a `tabular` one.

7 Backwards compatibility

As this class was originally written for a restricted use by a few Italians, there must be a definite backwards compatibility when this class is installed or is simply installed with an update of the TeX system distribution.

This new version in English, but adapted to work with French, Italian, German main languages (besides the improbable Latin and ancient Greek), must be backwards compatible with the initial `dizionarioSCR` class. It is; and the Italian commands and color definitions are still all there; there is just a small difference: the default main language is English; therefore Italian users who want to keep going with the old setup have two choices:

1. They just change only the `\documnetclass` statement into this new one:

   `\documentclass[`⟨`...,italian,...`⟩`]{dictionarySCR}`

2. They prepare a simple interface as this sample file:

   ```
   \NeedsTeXFormat{LaTeX2e}
   \ProvidesClass{dizionarioSCR}%
      [AAAA/MM/GG Interfaccia a dictionarySCR.cls]
   \DeclareOption*{\PassOptionToClass{\CurrentOption}{dictionarySCR}
   \ProcessOptions*\relax
   ```

```
\LoadClass[italian]{dictionarySCR}
\endinput
```

and save it as `dizionarioSCR.cls` in their personal `texmf` tree or in their working directory. They have to change `AAAA/MM/GG` with the actual date (year/month/day) of the day they create this small interface.

With this interface they don't need to change anything in their existing files.

NOTICE: solutions 1 and 2 above are mutually exclusive!

8 I comandi italiani

Questo paragrafo è dedicato agli utenti italiani. Come detto più volte nella parte in inglese, questa classe è nata originariamente per comporre un dizionario in italiano. I comandi erano tutti in italiano e lo sono ancora; ci sono molte equivalenze che fanno convivere i comandi originali in italiano, con i comandi "internazionali" in inglese e viceversa.

Già alcune parti del codice permettono di vedere queste equivalenze o queste doppie definizioni; per esempio tutti o quasi i colori definiti mediante `\definecolor` sono prima definiti in italiano e poi ogni definizione è ripetuta in inglese.

Non parliamo qui dei comandi di servizio; per esempio, `\virgoladecimale` è chiaramente in italiano, ma l'utente non userà mai questo comando direttamente. Nello stesso tempo si è preferito usare il nome del comando in italiano per evitare conflitti con altri pacchetti: per esempio la definizione delle regole per la lingua spagnola prevede l'uso di `\decimalcomma`, ovviamente definito in modo diverso...

Gli ambienti e i comandi in italiano sono mostrati nella tabella 3.

Va da sé che usare i nomi inglesi o quelli italiani è del tutto equivalente. Per quel che riguarda le parole "infisse", pochissime e relative solo ai comandi di riferimento incrociato relativi ai lemmi e ai loro contenuti, queste sono automaticamente gestite dalle definizioni interne relative alla lingua principale del documento. Nella documentazione in inglese è indicato come fare per aggiungere altre parole "infisse" per lingue diverse dall'inglese, italiano, francese e tedesco (oltre che latino e greco) che sono già gestite da questa classe.

9 The code

The preliminary statements that identify the class and the format file to use have been specified at the top of this file.

The next thing to do is to specify the option declarations for this class; any undeclared option given in the `\documentclass` statement will be passed on to the underlying `scrbook` class. Nevertheless we need a number of new options, specifically `9pt`, `debug`, `GreekTimes`, `noinputencoding`, and the various options necessary to specify the document main language, the default one being English. At the same time the underlying class `scrbook` is called with a specific set of

Comandi italiani	Comandi inglesi
`\lemma`	`\entry`
`\vedilemma`	`\seeentry`
`\rigaalta`	`\upperline`
`\rigabassa`	`\lowerline`
`\titoletto`	`\mtssubtitle`
`\titolino`	`\mtsubtitle`
`\titolo`	`\mttitle`
`\pareggia`	`\balance`
`\gradi`	`\degr`
`\freccia`	`\pointsto`
`\freccialunga`	`\longpointsto`
`\inciso`	`\insertremark`
`\AR`	`\RA`
Ambienti italiani	Ambienti inglesi
`miniindice`	`minitoc`
`cataloghi`	`catalogs`
Colori italiani	Colori inglesi
`rosellino`	`lightrose`
`verzolino`	`lightgreen`
`celeste`	`lightcyan`
`rossovivo`	`strongred`
`verdeguit`	`GuITgreen`
`giallo`	`yellow`
`azzurrino`	`lightblue`
`giallochiarissimo`	`verylightyellow`
`aranciochiarissimo`	`verylightorange`
`paglierino`	`straw`
`giallochiaro`	`lightyellow`
`giallocarico`	`strongyellow`

Tabella 3: Comandi, ambienti e colori in italiano e loro equivalenti in inglese

options. Notice that if another language is desired that is not included in this language option list, it must be specified as an option to the class; this will pass it silently to the called class and packages, `babel` included, but the main language usage must be explicitly specified at the beginning of the document, because this action is not done behind the scene as the declared language options do. Beware that there might be some clashes between language options different from `english`, `italian` and `french`; it depends on the text that is being typeset.

Any normal font size accepted by the underlying class `scrbook` may be specified among the `\documentclass` options; the accepted sises include 10pt, 11pt, and 12pt; no size declaration implies the 11pt font size. The 9pt font size is not forseeb

by the underlying class, so it is necessary to emulate it; for this purpse the `9pt` option sets to `true` the `shiftsizes` switch that will associate the usual LaTeX size commands to a shifted set of sizes.

```
\newif\ifshiftsizes      \shiftsizesfalse
\newif\ifGreekTimes      \GreekTimesfalse
\newif\ifDebug           \Debugfalse
\newif\ifNoInputEncoding \NoInputEncodingfalse
\DeclareOption{9pt}{\shiftsizestrue}
\DeclareOption{italian}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{italian}}}}
\DeclareOption{english}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{english}}}}
\DeclareOption{french}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{french}}}}
\DeclareOption{german}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{german}}}}
\DeclareOption{greek}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{greek}}}}
\DeclareOption{latin}{\AtEndOfClass{\AfterEndPreamble{\selectlanguage{latin}}}}
\DeclareOption{GreekTimes}{\GreekTimestrue}
\DeclareOption{debug}{\Debugtrue}
\DeclareOption{noinputencoding}{\NoInputEncodingtrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{scrbook}}
\ProcessOptions\relax
\LoadClass[twoside,a4paper,captions=tableheading,%
headings=small,captions=nooneline]{scrbook}
```

If option `9pt` is being expressed, it is necessary to shift the meanings of all size selecting commands starting from the set specified for the `10pt` option. But with the Koma-Script classes the default size is 11pt; therefore afre reading and loading the options and the underlyng class, if the `9pt}` option was specified, it is necessary to load the specific class option size file `scrsize10pt.clo` and perform the above mentioned shifting only after loading this size file.

This shifting consists in assigning the `\normalsize` at 10 pt to `\large` for the normal font at 9 pt, and so on with all other standard sizes; just `\tiny` remains the same and does not need to be shifted; but at the other end of the range `\Huge` at 10 pt becomes `\HUGE` at 9 pt. We do not exclude that more suitable definitions, relying on the fact that we are using uniformly scalable fonts to any size, might yield even better results; nevertheless the tests we performed with these shifted commands are perfectly acceptable. The first `\HUGE` assignment with `\let` is simply to provide a default definition in case no size option is specified, but actually it does not behave differently from `\Huge`.

```
\let\HUGE\Huge
\ifshiftsizes
   \input{scrsize10pt.clo}%
   \let\HUGE\Huge             %25pt
   \let\Huge\huge             %20pt
   \let\huge\LARGE            %17pt
   \let\LARGE\Large           %14pt
   \let\Large\large           %12pt
   \let\large\normalsize      %10pt
   \let\normalsize\small      % 9pt
   \let\small\footnotesize    % 8pt
```

```
30    \let\footnosize\scriptsize % 7pt
31    \let\scriptsize\tiny       % 6pt
32 \fi
```

It is worth noting the the base commands `\fontsize` followed by its two arguments, may always be used in order to specify font sizes not directly available with the standard LaTeX size commands; for example with a particular table it might be suitable to use a 8.5 pt font size; therefore a font size command such as `\fontsize{8.5}{10.2)\selectfont` would be a convenient way to make the table a little smaller; similarly a command such as `\fontsize{35pt}{35pt}\selectfont` would be a convenient solution to typeset a title in all caps, but avoiding the usual 20% interline additional space, since capital letters are virtually all without descenders. But these are just suggestions for emergency solutions, not for the main text. They are usable only if continuously scalable fonts are being used throughout the document.

It is worthwhile so remark that the possibility of selecting various normal sizes for typesetting a dictionary should not distract the user from the main oint: the font normal sise must be decided in advance; afterwords any table or picture or any other floating body must be sized according to the chosen default size. If by any chance the author would like to "work" the document by using a larger size, he should be aware that equations, floating odies and other fixed size material might produce problems by protruding into the right margin or by becoming a difficult floating body to be output in a regular page, to the point of saturating the respective queue and blocking the addition of further floating bodies. In spite of being annoying situations, please, don't worry about; just verify that there are no problems when the default normal size is restored; this is the final definitive document font, and only with this font everything has to be suitably designed and realized.

9.1 The text body grid

With the `memoir` class the normal font size implies the settings listed in table 4.

With the inner class `scrbook`, belonging to the KOMA-Script class family, it is convenient to let the class determine the best layout by simply fixing the text body grid. The point is that it is necessary to pilot the program to do what we want, not what it is its default behavior; in facts this class determines the layout with the method of the equal strips, and it assigns two strips to the outer/lower margins end one strip to the inner/upper margin. It is possible to compensate this effect by using the *binding correction* to be passed as an optional parameter to the call of`\areaset`; therefore here we avoid some of these complications and make explicit calculations by exploiting the extended TeX features available with *pdftex* since version 1.40; we also adjust to our "taste" the footnote distance and the upper margin:

```
33 \areaset[10mm]{\dimexpr448pt+5mm\relax}{\dimexpr639pt+6\baselineskip\relax}
34 \footskip2\baselineskip
35 \advance\topmargin1.5\baselineskip
```

The missing dimensions are determined by the `\areaset` command for an A4 paper size, but although they are similar to those computed by the `memoir` class,

Parametro	Valore
\textheight	639 pt
\textwidth	448 pt
\evensidemargin	89 pt
\oddsidemargin	60 pt
\topmargin	114 pt
\headheight	10 pt
\headsep	20 pt
\footskip	20 pt
\columnsep	10 pt
\columnseprule	0 pt
\marginparsep	17 pt
\marginparwidth	51 pt

Tabella 4: Layout design with the `memoir` settings with the normal size of 9 pt.

they are not identical. This is obviously OK since it is not necessary that similar classes with different algorithms produce identical layout parameters; moreover it's a matter of personal taste to decide which is the best layout when the differences are so small.

At the same time, after `\areaset` has determined its layout parameters, it is convenient to write into the `.log` file the results of these computations; it must be clear that the `.log` file will contain both layout values, the ones the the `scrbook` class determines by default and the ones that it determines with our text body requirements; the first message is the one output by the command `\areaset`, the second is the list typeset by this class with the following commands.

```
\ClassInfo{dictionarySCR}{%
  These are the values describing the layout:\MessageBreak
  DIV\space\space=\space\number\ta@div\MessageBreak
  BCOR\space=\space\the\ta@bcor\MessageBreak
  \string\paperwidth\space\space\space\space\space\space=%
                        \space\the\paperwidth\MessageBreak
  \space\string\textwidth\space\space\space\space\space\space=%
                        \space\the\textwidth\MessageBreak
  \space DIV-departure\space\space\space=%
                        \space\the\@tempcnta/100\MessageBreak
  \space\string\evensidemargin\space=%
                        \space\the\evensidemargin\MessageBreak
  \space\string\oddsidemargin\space\space=%
                        \space\the\oddsidemargin\MessageBreak
  \string\paperheight\space\space\space\space\space=%
                        \space\the\paperheight\MessageBreak
  \space\string\textheight\space\space\space\space\space=%
                        \space\the\textheight\MessageBreak
  \space\string\topmargin\space\space\space\space\space\space=%
                        \space\the\topmargin\MessageBreak
  \space\string\headheight\space\space\space\space\space=%
```

Parameter	Value
DIV	8
BCOR	0.0pt
\paperwidth	597.50793pt
\textwidth	448.0pt
DIV-departure	-4/100
\evensidemargin	27.40195pt
\oddsidemargin	-22.43402pt
\paperheight	845.04694pt
\textheight	639.0pt
\topmargin	-36.58768pt
\headheight	15.0pt
\headsep	18.0pt
\topskip	10.0pt
\footskip	42.0pt
\baselineskip	12.0pt

Tabella 5: typearea layout parameters when the text grid is specified

```
                          \space\the\headheight\MessageBreak
    \space\string\headsep\space\space\space\space\space\space\space\space=%
                          \space\the\headsep\MessageBreak
    \space\string\topskip\space\space\space\space\space\space\space\space=%
                          \space\the\topskip\MessageBreak
    \space\string\footskip\space\space\space\space\space\space\space=%
                          \space\the\footskip\MessageBreak
    \space\string\baselineskip\space\space\space=%
                          \space\the\baselineskip\MessageBreak
}%
```

Matter of fact the layout parameters computed by `\areaset` with the values we passed in the command arguments are listed in table 5.

9.2 Input/output codes and languages

If the `noinputencoding` option is specified, it means that the end user wants to specify his/her preferred input code in the preamble; otherwise we load by default the inputend package with the option `latin1`; in future releases we might chose a different default input encoding; but at the moment we prefer the `latin1` input encoding that with minor negligible incompatibilities is valid for all the main platforms with most operating systems.

Then languages are specified as options to the babel package; we must pay attention to the Greek language because it uses a specific encoding in order to typeset that language with its specific alphabet; nevertheless it uses the same family names as the corresponding Latin fonts of the CM and/or LM collections; series and shapes are substantially the same, although the CBgreek fonts have more families, series and shapes. Since here the Latin fonts are provided by the

Times eXtended collection, through the txfonts package, but the family names have different names from the ones used with the CM and/or LM collections, some patches need to be made, otherwise the typesetting program will use only the "error font", that is the medium series upright font taken from the `grmn*` fonts. It is therefore necessary to define the commands `\textLipsias` and `textDidot` in a compatible way with the TX fonts. We define them as "long" commands so as to make them capable of absorbing several paragraphs. Since Heinrich finds them to long to input, we defined also two almost equivalent shorter robust commands.

The whole Greek font management, when the package txfontsb by Antonis Tsolomitis is available, becomes even more complicated, because Tsolimitis' font description files, `.fd`, have "compulsory" names that equal to those of the CBgreek ones `lgrtx**.fd`, therefore one either uses the CBgreek fonts or the TXb ones; *they are mutually exclusive.*

For this reason, if the `GreekTimes` option is specified, this class loads the TXb fonts through txfontsb, if the latter is available, otherwise it resets the `GreekTimes` as if it was never specified. If the latter package is available Tsolomitis' fonts are loaded, otherwise the necessary adjustments are made in order to load the CBgreek fonts by directly creating the necessary declarations and definitions,

The option `GreekTimes` sets a boolean variable, initially `false`, to `true`. If this variable is `true` this class uses Tsolomitis' fonts if the TEX system is sufficiently recent and if those fonts are correctly installed.; otherwise, the boolean variable is set to `false` and the CBgreek fonts used as if the option was never specified.

When Tsolomitis' fonts are used, since they are incomplete, that is they do not contain all the glyphs that are contained in the CBgreek fonts, a new family is defined `txbfamily` that allows a local font change, although characterized by the same encoding series, shape and size, but belonging to another family. Of course in order to avoid messing up everything it's necessary that this local change is performed within a group.

```
\RequirePackage{etoolbox}
\unless\ifNoInputEncoding\RequirePackage[latin1]{inputenc}\fi
\RequirePackage[LGR,T1]{fontenc}
\RequirePackage[german,french,greek,latin,italian,english]{babel}
\languageattribute{greek}{polutoniko}
```

Some of the special accent macros, such as, for example, `\U`, are needed by Heinrich, not only to philologists; therefore it's necessary to replicate some macros contained in the teubner package, specifically `\lift@accent`, to redefine `\U`, to add `\lift@tie` so as to assure the macro works correctly with both the Latin fonts and the Greek ones, Tsolomitis' fonts included. Since the latter are not complete it's necessary to replace the wide breve (character 151 in `LGR` encoding) with the mathematical "smiley" that is included in the math italic `OML` encoded font (character 94 with encoding `OML`); since it was not created to be used as an accent, its necessary to fiddle a little bit in order to set up things in the proper way.

```
\newcommand*\lift@accent[2]{\leavevmode
{\edef\slant@{\strip@pt\fontdimen1\font}%
\dimen@=\z@\setbox\z@\hbox{\char#1}\advance\dimen@-.5\wd\z@
```

```
\setbox\tw@\hbox{i}\setbox\z@\hbox{#2}%
\ifdim\wd\z@>\wd\tw@\advance\dimen@ .5\wd\z@
    \else\advance\dimen@ .3\wd\z@\fi
\ifx#2a\advance\dimen@-.1\wd\z@\fi
\ifx#2h\advance\dimen@.05\wd\z@\fi
\@tempdima\ht\z@\advance\@tempdima-1ex\relax
\advance\dimen@\slant@\@tempdima
\raise\@tempdima\hbox to\z@{\kern\dimen@\char#1\relax\hss}\box\z@}}

\newcommand*\lift@tie[1]{\leavevmode
{\edef\slant@{\strip@pt\fontdimen1\font}%
\dimen@=\z@
\setbox4\hbox{\raise1.07ex%
    \hbox{\usefont{OML}{cmmi}{m}{it}\selectfont\char94}}\relax
\advance\dimen@-.5\wd4
\setbox\tw@\hbox{i}\setbox\z@\hbox{#1}%
\ifdim\wd\z@>\wd\tw@\advance\dimen@ .5\wd\z@
    \else\advance\dimen@ .3\wd\z@\fi
\ifx#1a\advance\dimen@-.1\wd\z@\fi
\ifx#1h\advance\dimen@.05\wd\z@\fi
\@tempdima\ht\z@\advance\@tempdima-1ex\relax
\advance\dimen@\slant@\@tempdima
\raise\@tempdima\hbox to\z@{\kern\dimen@\box4\hss}\box\z@}}


\DeclareTextCommand{\U}{LGR}[1]{%
\iffontchar\font151\relax
   \lift@accent{151}{#1}%
\else
   \lift@tie{#1}%
\fi}

```

The definitions of the new commands `\textLipsias` and `\textDidot` allow switching between typesetting in Latin script and Greek script, but they also change language and hyphenation rules

```
\def\lishape{\fontshape{li}\selectfont}
\def\spazia{\ifx\let@token\sp@token\ \fi}
\providecommand\textLipsias{}\providecommand\textDidot{}
\renewcommand\textLipsias[1]{\leavevmode\iflanguage{greek}{{\lishape#1}}
   {\begin{otherlanguage}{greek}{\lishape#1}\end{otherlanguage}}%
   \futurelet\let@token\spazia}
\renewcommand\textDidot[1]{\leavevmode\iflanguage{greek}{{\upshape#1}}
   {\begin{otherlanguage}{greek}{\upshape#1}\end{otherlanguage}}%
   \futurelet\let@token\spazia}
```

The `\GRD` and `\GRL` commands are now defined by means of the properties of the `\scantokens` primitive command; this necessity arises in order to avoid problems with definitions of certain character codes, especially the ambivalence of the tilde one; by using `\scantokens` the string to be typeset again is treated as it was just

read from an input file and default cathodes are assigned to its characters. The drawback is that `\scantokens` appends a space to its argument, and this space must be eliminated; we do this task by simply appending the `\relax` command at the end of the argument string.

```
\DeclareRobustCommand\GRD[1]{%
{\catcode`\"=12\relax\catcode`\~=12\relax
\usefont{LGR}{\f@family}{\f@series}{n}\selectfont\scantokens{#1\relax}}}
\DeclareRobustCommand\GRL[1]{%
    {\catcode`\"=12\relax\catcode`\~=12\relax
 \ifGreekTimes\usefont{LGR}{\f@family}{\f@series}{it}\else
 \usefont{LGR}{\f@family}{\f@series}{li}\fi\selectfont\scantokens{#1\relax}}}
```

We continue setting up the different actions that are required to properly use Tsolomitis' fonts or the other TEX system default fonts. We define the actual command to select certain greek CBfonts families to be used even when the Tsolomits Greek fonts are used. In any case it is necessary to define what is being meant when the `\lishape` is specified, since the Tsoloitis font don't have a Lipsian shape; they are also incomplete and so we use the CBfonts Greek numeral signs even when the Tsolomits fonts get used. But we delay their definitions until it has been decided what fonts to use. Meanwhile we load the default TXfonts, since they are a little more compact than the standard Latin Modern ones. For the Greek fonts usage we define also le font description macros to match the CBfonts to the TX ones, unless the `GreekTimes` option has been specified

```
\AtBeginDocument{\RequirePackage{txfonts}}

\ifGreekTimes
\IfFileExists{txfontsb.sty}{%
\AtBeginDocument{\RequirePackage{txfontsb}%
\let\lishape\itshape
}%
}{\GreekTimesfalse
\ClassWarning{dictionarySCR}{%
Greek fonts matching the TX ones are not available.
\MessageBreak
Option GreekTimes is ignored
}%
}
\else
\DeclareFontFamily{LGR}{txr}{}
  \DeclareFontShape{LGR}{txr}{m}{n}{<->ssub * cmr/m/n}{}
  \DeclareFontShape{LGR}{txr}{m}{it}{<->ssub * cmr/m/it}{}
  \DeclareFontShape{LGR}{txr}{m}{sl}{<->ssub * cmr/m/sl}{}
  \DeclareFontShape{LGR}{txr}{m}{sc}{<->ssub * cmr/m/sc}{}
  \DeclareFontShape{LGR}{txr}{b}{n}{<->ssub * cmr/bx/n}{}
  \DeclareFontShape{LGR}{txr}{b}{it}{<->ssub * cmr/bx/it}{}
  \DeclareFontShape{LGR}{txr}{b}{sl}{<->ssub * cmr/bx/sl}{}
  \DeclareFontShape{LGR}{txr}{b}{sc}{<->ssub * cmr/bx/sc}{}
  \DeclareFontShape{LGR}{txr}{bx}{n}{<->ssub * cmr/bx/n}{}
  \DeclareFontShape{LGR}{txr}{bx}{it}{<->ssub * cmr/bx/it}{}
```

```
  \DeclareFontShape{LGR}{txr}{bx}{sl}{<->ssub * cmr/bx/sl}{}
  \DeclareFontShape{LGR}{txr}{bx}{sc}{<->ssub * cmr/bx/sc}{}

  \DeclareFontShape{LGR}{txr}{m}{li}{<->ssub * cmr/m/li}{}
  \DeclareFontShape{LGR}{txr}{b}{li}{<->ssub * cmr/b/li}{}
  \DeclareFontShape{LGR}{txr}{bx}{li}{<->ssub * cmr/bx/li}{}
  \DeclareFontShape{LGR}{txr}{m}{ui}{<->ssub * cmr/m/ui}{}
  \DeclareFontShape{LGR}{txr}{b}{ui}{<->ssub * cmr/m/ui}{}
  \DeclareFontShape{LGR}{txr}{bx}{ui}{<->ssub * cmr/bx/ui}{}
  \DeclareFontShape{LGR}{txr}{m}{rs}{<->ssub * cmr/m/rs}{}
  \DeclareFontShape{LGR}{txr}{b}{rs}{<->ssub * cmr/m/rs}{}
  \DeclareFontShape{LGR}{txr}{bx}{rs}{<->ssub * cmr/bx/rs}{}

  \DeclareFontFamily{LGR}{txss}{}
  \DeclareFontShape{LGR}{txss}{m}{n}{<->ssub * cmss/m/n}{}
  \DeclareFontShape{LGR}{txss}{m}{it}{<->ssub * cmss/m/it}{}
  \DeclareFontShape{LGR}{txss}{m}{sl}{<->ssub * cmss/m/sl}{}
  \DeclareFontShape{LGR}{txss}{m}{sc}{<->ssub * cmss/m/sc}{}
  \DeclareFontShape{LGR}{txss}{b}{n}{<->ssub * cmss/bx/n}{}
  \DeclareFontShape{LGR}{txss}{b}{it}{<->ssub * cmss/bx/it}{}
  \DeclareFontShape{LGR}{txss}{b}{sl}{<->ssub * cmss/bx/sl}{}
  \DeclareFontShape{LGR}{txss}{b}{sc}{<->ssub * cmss/bx/sc}{}
  \DeclareFontShape{LGR}{txss}{bx}{n}{<->ssub * cmss/bx/n}{}
  \DeclareFontShape{LGR}{txss}{bx}{it}{<->ssub * cmss/bx/it}{}
  \DeclareFontShape{LGR}{txss}{bx}{sl}{<->ssub * cmss/bx/sl}{}
  \DeclareFontShape{LGR}{txss}{bx}{sc}{<->ssub * cmss/bx/sc}{}

  \DeclareFontFamily{LGR}{txtt}{\hyphenchar\font=-1}
  \DeclareFontShape{LGR}{txtt}{m}{n}{<->ssub * cmtt/m/n}{}
  \DeclareFontShape{LGR}{txtt}{m}{it}{<->ssub * cmtt/m/it}{}
  \DeclareFontShape{LGR}{txtt}{m}{sl}{<->ssub * cmtt/m/sl}{}
  \DeclareFontShape{LGR}{txtt}{m}{sc}{<->ssub * cmtt/m/sc}{}
  \DeclareFontShape{LGR}{txtt}{b}{n}{<->ssub * cmtt/bx/n}{}
  \DeclareFontShape{LGR}{txtt}{b}{it}{<->ssub * cmtt/bx/it}{}
  \DeclareFontShape{LGR}{txtt}{b}{sl}{<->ssub * cmtt/bx/sl}{}
  \DeclareFontShape{LGR}{txtt}{b}{sc}{<->ssub * cmtt/bx/sc}{}
  \DeclareFontShape{LGR}{txtt}{bx}{n}{<->ssub * cmtt/bx/n}{}
  \DeclareFontShape{LGR}{txtt}{bx}{it}{<->ssub * cmtt/bx/it}{}
  \DeclareFontShape{LGR}{txtt}{bx}{sl}{<->ssub * cmtt/bx/sl}{}
  \DeclareFontShape{LGR}{txtt}{bx}{sc}{<->ssub * cmtt/bx/sc}{}
\fi
```

it's necessary to provide the required commands to use this shape and the commands for using the Greek numbers, both the Milesian and the Attic ones, with the proper symbols that are missing from Tsolomitis' fonts. some of them are defined but the variant `\qoppa` symbols are missing and all the symbols required for the Attic numerals.

```
\def\txbfamily{\usefont{LGR}{lmr}{\f@series}{\f@shape}}

\DeclareTextCommand{\stigma}{LGR}{{\txbfamily\textstigma}}
```

```
\DeclareTextCommand{\varstigma}{LGR}{{\txbfamily\textvarstigma}}
\DeclareTextCommand{\koppa}{LGR}{{\txbfamily\textkoppa}}
\DeclareTextCommand{\varkoppa}{LGR}{{\txbfamily\textqoppa}}
\let\coppa\varkoppa
\DeclareTextCommand{\sampi}{LGR}{{\txbfamily\textsampi}}
\DeclareTextCommand{\Coppa}{LGR}{{\txbfamily\textQoppa}}
\let\Koppa\Coppa
\DeclareTextCommand{\Stigma}{LGR}{{\txbfamily\textStigma}}
\DeclareTextCommand{\Sampi}{LGR}{{\txbfamily\textSampi}}
\DeclareTextCommand{\Euro}{LGR}{{\txbfamily\texteuro}}
\DeclareTextCommand{\permill}{LGR}{{\txbfamily\textperthousand}}

```

Since one of the authors of this class is also author of the teubner package, some definitions are directly copied from the latter one together with the algorithms required to transform the Arabic numerals into the Greek ones.

```
\DeclareTextCommand{\f}{LGR}{{\txbfamily\textdigamma}}
\AtBeginDocument{
   \@ifpackageloaded{amssymb}%
     {% amssymb loaded
        \let\AMSdigamma\digamma
        \def\digamma{\textormath{\f}{\AMSdigamma}}}
     {% amssymb not loaded
        \let\digamma\f}
}
   \DeclareTextCommand{\F}{LGR}{{\txbfamily\textDigamma}}
   \DeclareTextCommand{\Digamma}{LGR}{{\txbfamily\textDigamma}}

\DeclareRobustCommand{\fLow}%
    {{\setbox\z@\hbox{\f}\dimen@\ht\z@
    \advance\dimen@-1ex\raise-\dimen@\hbox{\box\z@}}}
\DeclareRobustCommand{\fHigh}%
    {{\setbox\z@\hbox{\f}\dimen@\dp\z@\raise\dimen@\hbox{\box\z@}}}

\def\@ifStar#1#2{\def\@tempA{#1}\def\@tempB{#2}\futurelet\@tempC\@testStar}
\def\@testStar{\ifx\@tempC*\bbl@afterelse\expandafter\@tempA\@gobble\else
      \bbl@afterfi\@tempB\fi}
\DeclareRobustCommand*{\Greeknumeral}{%
      \let\n@vanta\Coppa\let\n@vecento\Sampi
      \@ifStar{\Gr@@kn@meral}{\Gr@@knum@ral}}
\DeclareRobustCommand*{\greeknumeral}{%
      \let\n@vanta\varkoppa\let\n@vecento\sampi
    \@ifStar{\let\s@i\stigma\gr@@numeral}{\let\s@i\fLow\gr@@numeral}}
\def\Gr@@kn@meral#1{\let\s@i\Stigma
    \expandafter\MakeUppercase\expandafter{\gr@@numeral{#1}}}
\def\Gr@@knum@ral#1{\let\s@i\Digamma
    \expandafter\MakeUppercase\expandafter{\gr@@numeral{#1}}}
\def\grtoday{{\expandafter\greeknumeral\expandafter{\the\day}}\space
\gr@c@month\space{\expandafter\greeknumeral\expandafter{\the\year}}}

```

```
\def\gr@@numeral#1{\mbox{%
  \ifnum#1<\@ne\space\gr@ill@value{#1}%
  \else
    \ifnum#1<10\relax\expandafter\gr@num@i\number#1%
    \else
      \ifnum#1<100\relax\expandafter\gr@num@ii\number#1%
      \else
        \ifnum#1<\@m\relax\expandafter\gr@num@iii\number#1%
        \else
          \ifnum#1<\@M\relax\expandafter\gr@num@iv\number#1%
          \else
            \ifnum#1<100000\relax\expandafter\gr@num@v\number#1%
            \else
              \ifnum#1<1000000\relax\expandafter\gr@num@vi\number#1%
              \else
                \space\gr@ill@value{#1}%
              \fi
            \fi
          \fi
        \fi
      \fi
    \fi
  \fi
}}
\def\gr@num@i#1{%
  \ifcase#1\or a\or b\or g\or d\or e%
  \or \s@i\or z\or h\or j\fi
  \ifnum#1=\z@\else\anw@true\fi\anw@print}
\def\gr@num@ii#1{%
  \ifcase#1\or i\or k\or l\or m\or n%
  \or x\or o\or p\or \n@vanta\fi
  \ifnum#1=\z@\else\anw@true\fi\gr@num@i}
\def\gr@num@iii#1{%
  \ifcase#1\or r\or s\or t\or u\or f%
  \or q\or y\or w\or \n@vecento\fi
  \ifnum#1=\z@\anw@false\else\anw@true\fi\gr@num@ii}
\def\gr@num@iv#1{%
  \ifnum#1=\z@\else\katwtonos\fi
  \ifcase#1\or a\or b\or g\or d\or e%
  \or \s@i\or z\or h\or j\fi
  \gr@num@iii}
\def\gr@num@v#1{%
  \ifnum#1=\z@\else\katwtonos\fi
  \ifcase#1\or i\or k\or l\or m\or n%
  \or x\or o\or p\or \n@vanta\fi
  \gr@num@iv}
\def\gr@num@vi#1{%
  \katwtonos
  \ifcase#1\or r\or s\or t\or u\or f%
  \or q\or y\or w\or \n@vecento\fi
```

```
  \gr@num@v}
\ifGreekTimes
\DeclareTextCommand{\Vmiria}{LGR}{{\txbfamily\char5}}
\DeclareTextCommand{\Vkilo}{LGR}{{\txbfamily\char4}}
\DeclareTextCommand{\Vetto}{LGR}{{\txbfamily\char3}}
\DeclareTextCommand{\Vdeka}{LGR}{{\txbfamily\char2}}
\else
\DeclareTextSymbol{\Vmiria}{LGR}{5}
\DeclareTextSymbol{\Vkilo}{LGR}{4}
\DeclareTextSymbol{\Vetto}{LGR}{3}
\DeclareTextSymbol{\Vdeka}{LGR}{2}
\fi
\newcommand*\attic@ill@value[1]{\PackageWarning{dictionarySCR}{%
Illegal value (\number#1) for \string\ACticNumeral\space}}
\DeclareRobustCommand*\AtticNumeral[1]{%
\ifnum#1<\@ne \attic@ill@value{#1}\else
  \ifnum#1>99999\relax \attic@ill@value{#1}\else
    \AtticCycl@{#1}
  \fi
\fi}

\def\AtticCycl@#1{%
    \bgroup
    \countdef\valore=252\countdef\cifra=250\relax
    \valore=#1\relax
    \cifra=\valore \divide\cifra10000\relax
    \valore=\numexpr\valore-\cifra*10000\relax
    \ifnum\cifra>4\relax\Vmiria \advance\cifra-5\fi
       \@whilenum\cifra>\z@\do{M\advance\cifra\m@ne}%
    \cifra=\valore \divide\cifra1000\relax
    \valore=\numexpr\valore-\cifra*1000\relax
    \ifnum\cifra>4\relax\Vkilo \advance\cifra-5\fi
       \@whilenum\cifra>\z@\do{Q\advance\cifra\m@ne}%
    \cifra=\valore \divide\cifra100\relax
    \valore=\numexpr\valore-\cifra*100\relax
    \ifnum\cifra>4\relax\Vetto \advance\cifra-5\fi
       \@whilenum\cifra>\z@\do{H\advance\cifra\m@ne}%
    \cifra=\valore \divide\cifra10\relax
    \valore=\numexpr\valore-\cifra*10\relax
    \ifnum\cifra>4\relax\Vdeka \advance\cifra-5\fi
       \@whilenum\cifra>\z@\do{D\advance\cifra\m@ne}%
    \cifra=\valore
    \ifnum\cifra>4\relax P\advance\cifra-5\relax\fi
       \@whilenum\cifra>\z@\do{I\advance\cifra\m@ne}%
    \egroup}

```

9.3 Typesetting style and facilities

We set by default the french typesetting style without any larger space after major punctuation, in particular after a full stop.

```
\frenchspacing
```

We load the fixltx2e package that fixes some glitches remained since the transition form LaTeX 209 to LaTeX 2_ε; in our case we are mostly concerned with the column vs. full width floating objects and we want to avoid that a smaller numbered floating object is output after a larger numbered one. Actually it might seem a useless concern since we do not number figures; no, it's only apparent: numbers are generated and are used by the hyperlink management mechanism.

```
\RequirePackage{fixltx2e}
```

9.4 The comma sign in mathematics

While writing mathematics in a context different from English, the comma sign plays two rôles: that of a punctuation mark and that of a decimal separator. It is necessary to distinguish these two rôles for a correct professional composition. In facts, notice the difference in the expression $f(x,y) \neq f(x, y)$; the difference is small but in the second case, where the comma is a punctuation mark, the comma is followed by a small space, while in the first case, where the comma is treated as a decimal separator (although improperly, since x and y are not decimal digits) there is no space at all.

The trick for properly handling commas in mathematics consists in detecting if the token that immediately follows the comma sign is a digit or another sign; in the first case the comma sign must be treated as a decimal sign, while in the other cases it must be treated as a punctuation mark, i.e.as a list item separator. The only case where this strategy might fail is when the comma sign plays the rôle of item separator and items are made up with digits, as when you write something such as $\forall n = 0, 1, 2, \ldots, N$ and you forget to insert a space in the source file between the comma sign and the following digit; you have to input, quite naturally, `$\forall␣n=0,␣1,␣2,\dots,N$`. In any case, since in 2013 an intelligent comma definition is embedded in the `babel-italian` language definition file, we test if the definition is already available, otherwise we set a definition of our own.

```
\ifdefined\@math@@comma
   \IntelligentComma
\else
   \DeclareMathSymbol{\virgola}{\mathpunct}{letters}{"3B}
   \DeclareMathSymbol{\virgoladecimale}{\mathord}{letters}{"3B}
   \DeclareMathSymbol{\plaindot}{\mathord}{letters}{"3A}
%
   \AtBeginDocument{\AfterEndPreamble{%
      \unless\ifnum\language=\l@english\mathcode`\,="8000\fi}}
   {\catcode`\,=\active \gdef,{\futurelet\let@token\m@thcomma}}
   \def\m@thcomma{\let\@tempB\virgola
   \@tfor\@tempA:=0123456789\do{%
  \expandafter\ifx\@tempA\let@token\let\@tempB\virgoladecimale
```

```
  \@break@tfor\fi}\@tempB}
\fi
```

Incidentally it is worth noting that long numbers may have their digits grouped by triplets before and/or after the decimal separator; in order to achieve this behavior they *must* be separated by a thin space `\,`, while it is *absolutely forbidden* to separate them with any other sign, be it a point or a comma, as it is done when printing bills or other commercial documents. The thin space is not necessary if the digits before or after the decimal separator are less than five; therefore it is correct to write: 12 345,678 90, or 1234,567 890, or 123 456,7890, or 1234,5678.

9.5 Loaded packages

Now several packages are loaded with their options. Some of these packages, included some of their options, derive from explicit Henrich's requests; while a dictionary's drafts were getting ready, some packages were discovered to be ignored in actual use and were eliminated in the progressive fine tuning of this class; some packages were requested by Claudio, the most important of which are the microtype package for using all possible means to increase the typesetting quality of this dictionary two column layout; and the trace package for tracing the typesetting engine activity when errors had to be discovered; of course this package is loaded only when the class option `debug` is specified.

```

\RequirePackage{amsmath,amssymb,delarray}
\ifdim\overfullrule>\z@
\RequirePackage[draft]{graphicx}
\else
\RequirePackage{graphicx}
\fi
\RequirePackage{captcont}
\RequirePackage{xcolor}
\RequirePackage{afterpage}
\RequirePackage{eso-pic}

\RequirePackage{varioref}
\RequirePackage{longtable}
\RequirePackage{paralist}
\RequirePackage{textcomp}
\RequirePackage{wasysym}
\RequirePackage[varg]{txfonts}
\RequirePackage[squaren,cdot,binary,noams,derivedinbase,%
derived]{SIunits}
\ifDebug\RequirePackage{trace}\fi
\RequirePackage{microtype}
\RequirePackage{framed}
\RequirePackage{colortbl}
\RequirePackage{booktabs}
\RequirePackage{marvosym}
\RequirePackage{imakeidx}
```

```
\let\possiblysortindexfiles\relax
\let\diz@printindex\printindex
\renewcommand\printindex{\onecolumn\diz@printindex}

```

For the graphicx package correct working we specify a folder where figures are placed; of course the end user can add to the preamble the specifications for other folders and subfolders.

```
\graphicspath{{Figure/}}
```

In facts the composer should rather specify another picture folder at the beginning of each "chapter", corresponding to a different initial entry letter so as to keep the picture better divided while letting the composer specify only once the folder preamble for each picture; remember that it theoretically possible to specify an absolute address with a path that starts from the hard disk name, but this is strongly discouraged not only in this class, but for any responsible use of the TEX system programs. Reading from a wrong folder might produce errors and no damages to the disk contents, but writing to a wrong absolute address folder might be disastrous; and the end user might be unaware of the TEX system background workings.

For typesetting the index we prefer to balance the columns in the last page; this goal is achieved by using the multicol package rather than using the balance one, whose effectiveness is questionable, because it inhibits some elementary actions that should be done in the last page the columns of which should be balanced.

```
\RequirePackage{multicol}
```

For setting some figures or tables sticking in the outer margin it appears convenient to use the marginnote package and its command `\marginnote` rather than the default command `\marginpar`; apparently `\marginnote` is more robust in handling the operations needed to set hanging figures and tables.

```
\RequirePackage{marginnote}
\renewcommand*{\raggedleftmarginnote}{\raggedright}
\renewcommand*{\raggedrightmarginnote}{\raggedleft}
```

For customizing the footnote typesetting style the specific commands of the KOMA-Script class collection are used; with the following settings the space reserved for the note symbol is 1 em wide, and the note symbol has 1 en (0.5 em) on its right. May be that 1 em is too little, but in a dictionary's drafts is appears to be the right amount of space. In any case is is trivial to reset it in the preamble wit, say, 1.5 em. The typesetting style is without any hanging indentation and if a note is made up of several paragraphs, these are set without indentation. Since a dictionary uses no or very few notes, we decide to number them by "entries", as we do for equations. nevertheless the master counter setting will be set further on.

```
\deffootnote[1em]{\z@}{\z@}{\hfill\thefootnotemark.\enspace}
```

9.6 Redefining chapters and sections

Here we redefine the style for typesetting chapter heads and sections; in facts we have to number all these sectional commands, but we do not want to show their numbers; such numbers are internally used for building the hyperlinks, but they are non sense in the dictionary proper.

We start redefining the `\section` command; its title will be vertically separated from the end of the previous paragraph, but it will be in line with its text. We want also that the fonts to be of the same size as the normal text; to this end, since we called the inner class `scrbook` with the option `headings=small`, it's the subsection font size that is the same as `\normalsize`.

```
\renewcommand\section{\@startsection{section}{1}{\z@}%
  {-2.25ex\@plus -1ex \@minus -.2ex}%
  {-1em}%
  {\ifnum \scr@compatibility>\@nameuse{scr@v@2.96}\relax
          \setlength{\parfillskip}{\z@ plus 1fil}\fi
    \raggedsection\normalfont\sectfont\nobreak\size@subsection
  }%
}
\let\size@section\normalsize
```

The first commands that follow this paragraph are used to customize the tile and the headers when a new chapter is composed. In this dictionary class a "chapter" is a main section where the entries initial letter is new compared to the one in the previous "chapter". This chapter must be numbered for internal hyperlink necessities but the chapter number mus not appear anywhere in the typeset dictionary; such a chapter already contains a label of the form `chap;`⟨*letter*⟩, (for example `chap:A`), that may be referred to in any part of the dictionary. The internal command `\@chapter` is being redefined in an absolute way by means of `\def` because one of its arguments is delimited by square brackets; this `\@chapter` command is called by a kernel one, so it must have exactly that structure.

Furthermore in order to correct a little glitch due to the asynchronous page make up it's necessary to have a control sequence the stores the current entry when a new one is being entered; but at the "chapter" start no entries have been entered yet and the `\chapter` command has cleared all marks. Therefore it's necessary to modify `\chapter` in order to clear also the `\currentlemma` contents, otherwise the chapter first entry does not know how to set the marks in order to be consistent with the chapter initial letter – in other words there must not be a left mark that is last entry of the previous chapter.

```
\@at@twocolumntrue
\renewcommand*\chapterformat{}
\renewcommand*\chaptermarkformat{}
\def\size@chapter{\fontsize{40}{40}\selectfont}
\renewcommand*\chapterheadstartvskip{\vspace*{-2.2\baselineskip}}
\renewcommand*{\chaptermark}[1]{}%
\providecommand*\currentlemma{}%
\renewcommand\chapter{\if@openright\cleardoublepage\else\clearpage\fi
  \thispagestyle{\chapterpagestyle}%
```

```
  \global\@topnum\z@
  \let\currentlemma\empty
  \@afterindentfalse
  \secdef\@chapter\@schapter
}

\newcommand*\unghia{}

\def\@chapter[#1]#2{%
  \ifnum \c@secnumdepth >\m@ne
    \if@mainmatter
      \refstepcounter{chapter}%
      \typeout{\@chapapp\space\thechapter.}%
      \addcontentsline{toc}{chapter}{#1}%
    \fi
  \fi
  \chaptermark{}%
  \if@twocolumn
    \if@at@twocolumn
      \@makechapterhead{#2}%
    \else
      \@topnewpage[\@makechapterhead{#2}]%
    \fi
  \else
    \@makechapterhead{#2}%
    \@afterheading
  \fi
  \label{chap:#1}%
  \def\unghia{#1}%
}
```

9.7 Entries and their hyperlinks

Here for backwards compatibility we let `\oldparagraph` be an alias to `\section`; it is not very useful but gives the possibility of changing the equivalent command without changing the definition of the internal command `\lemma` the real command that the alias `\entry` uses for entering a new entry in the dictionary; always for backwards compatibility also `\paragraph` is made an alias to `\lemma`.

This command uses the entry title also for making up a hyperlink.
PLEASE TAKE NOTICE: if the entry title is empty or if it contains expandable macros the generated hyperlink might target the first page of the document (page 1 or page i). These situations must be taken care by the end user; it is possible to use a different label than the entry title that does not contain expandable parts by using the `\entry` optional argument; for example in order to hyperlink the entry "ï¿½ngstrï¿½m Ander Jonas" the `.tex` source file might contain `\entry[Angstrom Ander Jonas]{ï¿½ngstrï¿½m Ander Jonas}`; a reference to this entry might be done with `\seeentry[Angstrom Ander`

Jonas]{i¿½ngstri¿½m Ander Jonas}. The source file readability and the precision of the references are not reduced while the hyperlinks are correct.

Another delicate question concerns the setting of the contents of \currentlemma at the end of the definition; but at the beginning it's necessary to check if this macro is empty; in this case the first entry of a chapter is being entered, therefore it is correct to load it with the current lemma title. The there is some magics taken from the TEXbook, chapter 23: marks are set both before and after the execution of the \section command (through the alias \oldparagraph). This is done so as to reassign the preceding entry marks, so that if \section produces a page break, the first mark assignment becomes the \botmark and the second mark assignment becomes the \firstmark; the \topmark is internally assigned in a correct way so that the left mark on every page continues to point to the entry title that was in force at the beginning of the page, unless a new entry starts the page and the entry title is in the first line of the left column. Before doing this, though, we define two new skip registers and redefine the \section macro so as to have a code that is easier to customize The negative sign before the \abovesectionskip implies that the absolute value of the skip is to be dealt with before the section heading; the implied positive sign before \belowsectionskip implies that the skip is to be used below the section heading; should this skip be negative, it is dealt with as an horizontal skip to the right of the section heading. This is why if a statement such as

```
\belowsectionskip=-\belowsectionskip
```

issued after the begin{document} statement turns this value into a negative one and the entry description starts on the same line as the section heading. This customization facility is very convenient; we suggest to take into account the possibility of setting \abovesectionskip to 0 pt plus a small amount of stretch. No space will separate the various entries and the dictionary turns out to be much more compact, without any loss of functionality.

```
437
438 \newlength\abovesectionskip \abovesectionskip=3.5ex \@plus 1ex \@minus .2ex
439 \newlength\belowsectionskip \belowsectionskip=2.3ex \@plus.2ex
440
441 \renewcommand\section{\@startsection{section}{1}{\z@}%
442   {-\abovesectionskip}%
443   {\belowsectionskip}%
444   {\ifnum \scr@compatibility>\@nameuse{scr@v@2.96}\relax
445     \setlength{\parfillskip}{\z@ plus 1fil}\fi
446     \raggedsection\normalfont\sectfont\nobreak\size@section}%
447 }
448
449
450
451 \newcommand\lemma[2][]{%
452 \ifx\currentlemma\empty\def\currentlemma{#2}\fi
453 \markboth{\currentlemma}{\currentlemma}% before "section" is called
454 \ifblank{#1}{%
```

```
    \section{#2}\label{lm@#2}%
}{%
    \section{\texorpdfstring{#2}{#1}}\label{lm@#1}%
}
\markboth{#2}{#2}% after the entry title is printed
\def\currentlemma{#2}\ignorespaces}
\let\paragraph\lemma\let\entry\lemma

```

Command `\seeentry` is used to reference an entry with an hyperlink; also `\seeentry` accepts an optional argument to build up the hyperlink target without using any string containing expandable elements.

```
\newcommand*\vedilemma[2][]{%
\ifblank{#1}{\hyperref[lm@#2]{#2}}{\hyperref[lm@#1]{#2}}}
\let\seeentry\vedilemma
```

Since the equation numbers restart from 1 at each entry (they do not have a hierarchic numbering containing the chapter and the section numbers that are never printed in the dictionary) it is necessary to make all references to equations by means of their (unique) labels; but there may be hundreds of equations numbered "2" so it's necessary to cite also the entry title where that particular "equation 2" belongs to. To this end we need a new command, easy to remember, but different from `\eqref` as provided by the amsmath package, that typesets something as, for example, "... Equation 2 of entry Abbe...".

To this end it is necessary to recall the LaTeX cross reference data are redefined by the hyperref package so that they contain five pieces of information: the first two ones are identical to those defined by the LaTeX kernel: that is, the last counter value and the page number where the `\label` command falls in; the third piece of information is the "section" title followed by `\relax` where the `\label` falls in; the fourth piece of information is a string made up by the referenced object name (chapter, section, figure, equation,etc.) followed by a full stop, followed by the chapter number, followed by a full stop, followed by the section number, followed by a full stop, followed by the object number; for example the second equation of the three hundred and fifteenth entry of chapter C of the dictionary shall have this string as the fourth piece of information: `equation.3.315.2`; the fifth piece of information is of no interest in our case (besides the fact that generally it's empty).

The tricky part is to fetch and cleanup the data required to correctly handle our equation references; we have to fetch from the third piece of information the entry label (the entry title or the entry optional argument), clean it up from the whatever possibly follows a comma, and then to generate the text to use as the reference and the links to the original entry and specific equation.

```
\newcommand*\equref[2][]{\ifcsdef{r@#2}{%
\ifblank{#1}{%
    \edef\lemm@ref{\expandafter\LemmaEqu\csname r@#2\endcsname}%
}{%
    \edef\lemm@ref{#1}%
}%
```

```
\def\strip@ftercomma ##1,##2\@nil{##1}%
\edef\lemmaref{\expandafter\strip@ftercomma\lemm@ref,\@nil}%
\edef\argomentoperhyperref{[lm@\lemm@ref]{\lemmaref}}%
\textbf{\autoref{#2}} \ofentry\
        \textbf{\expandafter\hyperref\argomentoperhyperref}%
}{??}}
%
\def\diz@thirdoffive#1#2#3#4#5{#3}
\def\diz@secondoffive#1#2#3#4#5{#2}
\def\diz@firstoffive#1#2#3#4#5{#1}
\def\NumeroEqu#1{\ifx#1\relax ??\else\expandafter\diz@firstoffive#1\fi}
\def\PaginaEqu#1{\ifx#1\relax ??\else\expandafter\diz@secondoffive#1\fi}
\def\LemmaEqu#1{\ifx#1\relax ??\else\expandafter\diz@thirdoffive#1\fi}
```

9.8 Versals

For typesetting versals it is convenient to have available a fixed very large font; of course other packages, such as lettrine for example, are capable of adjusting the versal size according to several parameters and user options. In our case we want to create versals that occupy only two lines in the default font size and a simpler macro is sufficient.

```
\DeclareFixedFont{\largefont}{T1}{txr}{m}{n}{42}%
```

For typesetting versals Heinrich found the simple macro prepared by David C. Cantor e Dominik Wujastyk and those of the obsolete (and not available any more) versal package; he wanted to use these macros in the preliminary versions of this class; we repeat them just for backwards compatibility, while we acknowledge the original authors' copyright; although we decided to keep the `\versal` command, we tied it to the `\drop` macro by the other authors. It is a convenient way to avoid duplicating code and code that implements facilities we don't use.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start of style drop.sty
%  Macro written by David G. Cantor, and published Fri, 12 Feb 88, in
%  TeXhax, 1988 #16.  Internet:  dgc@math.ucla.edu
%  Modified for use with LaTeX by Dominik Wujastyk, February 17, 1988
%  Internet:   dow@wjh12.harvard.edu Bitnet:     dow@harvunxw.bitnet
\def\drop#1#2{{\noindent
    \setbox0\hbox{\largefont #1}\setbox1\hbox{#2}\setbox2\hbox{(}%
    \count0=\ht0\advance\count0 by\dp0\count1\baselineskip
    \advance\count0 by-\ht1\advance\count0by\ht2
    \dimen1=.5ex\advance\count0by\dimen1\divide\count0 by\count1
    \advance\count0 by1\dimen0\wd0
    \advance\dimen0 by.25em\dimen1=\ht0\advance\dimen1 by-\ht1
    \global\hangindent\dimen0\global\hangafter-\count0
    \hskip-\dimen0\setbox0\hbox to\dimen0{\raise-\dimen1\box0\hss}%
    \dp0=0in\ht0=0in\box0}#2}
% End of style drop.sty
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Sometimes it's necessary to balance the last two columns of a chapter; there exists the `balance` package, but it does not comply with the presence of floating bodies in the page where the columns have to be balanced. Here we devise a different solution; we define a `\balance` macro that is assumed to be used by hand by the typesetter if and when s/he desires to balance the columns. We admit that even this solution suffers a little bit if after the line where this command is inserted there is some floating body insertion. Nevertheless, at least this implies that there should not be floating bodies in the second column. Since anyhow this command is to be placed in the source tex by a cut-and-try approach, there is plenty of occasions to revise the floating objects positioning, so as to avoid problems. The eye of the typesetter is a better judge than an automatic approach.

```
\def\pareggia{%
  \ifvmode
    \vfill
    \penalty -\@M%
  \else
    \@bsphack
    \vadjust{\vspace{\z@\@plus1fill}\penalty -\@M}%
    \@esphack
  \fi}
\let\balance\pareggia
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcommand*\versal[1]{\drop{#1}{\relax}}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

9.9 Colors

The following color definitions have been chosen by Heinrich; Claudio added the English definitions.

```
\definecolor{rosellino}{rgb}{1,0.75,0.75}
\definecolor{lightrose}{rgb}{1,0.75,0.75}
\definecolor{verzolino}{rgb}{0.75,1,0.75}
\definecolor{lightgreen}{rgb}{0.75,1,0.75}
\definecolor{celeste}{rgb}{0.75,0.9,1}
\definecolor{lightcyan}{rgb}{0.75,0.9,1}
\definecolor{rossovivo}{rgb}{0.7,0.2,0.2}
\definecolor{strongred}{rgb}{0.7,0.2,0.2}
\definecolor{verdeguit}{rgb}{0,0.6,0}
\definecolor{GuITgreen} {rgb}{0,0.6,0}
\definecolor{gray6}{gray}{.4}
\definecolor{gray3}{gray}{.3}
\definecolor{giallo}{rgb}{0.9,0.9,0.}
\definecolor{yellow}{rgb}{0.9,0.9,0.}
\definecolor{shadecolor}{rgb}{.95,.90,.60}
\definecolor{LightRed}{rgb}{1,.70,.70}
\definecolor{LightYellow}{rgb}{.90,.85,.55}
\definecolor{LightGray}{rgb}{.90,.90,.90}
\definecolor{MediumGray}{rgb}{.70,.70,.70}
```

```
\definecolor{DarkGray}{rgb}{.50,.60,.70}
\definecolor{StrongGray}{rgb}{.10,.20,.30}
\definecolor{DarkBlue}{rgb}{0,0,0.9}
\definecolor{azzurrino}{rgb}{0.8,0.8,1}
\definecolor{lightblue}{rgb}{0.8,0.8,1}
\definecolor{giallochiarissimo}{rgb}{1,0.98,0.86}
\definecolor{verylightyellow}{rgb}{1,0.98,0.86}
\definecolor{aranciochiarissimo}{rgb}{1,0.88,0.71}
\definecolor{verylightorange}{rgb}{1,0.88,0.71}
\definecolor{paglierino}{rgb}{0.99,1,0.60}
\definecolor{straw}{rgb}{0.99,1,0.60}
\definecolor{giallochiaro}{rgb}{0.99,1,0.64}
\definecolor{lightyellow}{rgb}{0.99,1,0.64}
\definecolor{giallocarico}{rgb}{0.99,1,0.35}
\definecolor{strongyellow}{rgb}{0.99,1,0.35}
```

Some of these colors are used for link colors, others for rows, columns, cells, table colors.

9.10 Hyperlink

The last package to be loaded is hyperref; it must be the last one because it redefines some of the LaTeX kernel commands that may have been redefined by other previous packages.

```
\RequirePackage[colorlinks,linkcolor=red,citecolor=verdeguit,%
                urlcolor=DarkBlue,hyperindex]{hyperref}
```

In order to make link anchors more visible it is convenient that the object name being linked is part of the anchor; for this purpose the following commands must be defined and the commands `\autoref` and `\autopageref` should be used in place of `\ref` and `\pageref`. We provide the necessary definitions for all the languages loaded as options to the babel package. For other languages it is necessary to define similar "additions" to the specific language extras and to add them after the specific `\selectlanguage` command, for example as in:

```
\begin{document}
\selectlanguage{spanish}
\addto\extrasspanish{\def\sectionautorefname{entrada}%
\def\paragraphautorefname{entrada}%
\def\equationautorefname{ecuaci\'on}%
\def\ofentry{de la ecuaci\'on}}
```

Here are the definitions for the standard languages; actually the "equation" auto ref name is already defined in French and Italian within the hyperref package auxiliary files, but the definition is capitalized and this is not the common usage in Italian and in French. The same practice is proposed for the Spanish extras above.

```
\addto\extrasenglish{\def\sectionautorefname{entry}%
\def\paragraphautorefname{entry}%
\def\ofentry{sub entry}}
```

```
556
557 \addto\extrasitalian{\def\sectionautorefname{lemma}%
558 \def\paragraphautorefname{lemma}%
559 \def\equationautorefname{equazione}%
560 \def\ofentry{\emph{sub} lemma}}
561 %
562 \addto\extrasfrench{\def\sectionautorefname{article}%
563 \def\paragraphautorefname{article}%
564 \def\equationautorefname{\'equation}%
565 \def\ofentry{sous l'article}}
566 %
567 \addto\extrasgerman{\def\sectionautorefname{Stichwort}% Verificare genere
568 \def\paragraphautorefname{Stichwort}%
569 \def\ofentry{unten das Stichwort}}
570 %
571 \addto\extraslatin{\def\sectionautorefname{lemma}%
572 \def\paragraphautorefname{lemma}%
573 \def\equationautorefname{equationis}% "equationem"?
574 \def\ofentry{sub lemmate}}
575 %
576 \addto\extraspolutonikogreek{% "monotoniko spelling"?
577 \def\sectionautorefname{e\>'isodos}%
578 \def\paragraphautorefname{e\>'isodos}%
579 \def\equationautorefname{>exomo'iwsh}%
580 \def\ofentry{<up`o t`on e\>'isodon}}
581
```

9.11 Page styles

The dictionary pages require a particular page style; these pages must not contain chapter and section numbers or titles; they must contain a header where on its left side, irrespective o the page parity, there is the entry title of the first header on the page, possibly the header that was in force in the preceding page and is not yet finished; on the right side the title of the last entry on the page, possibly the same title as on the left if the entry description is not yet finished. In any case the mechanism we implemented works pretty well, but in this respect LaTeX has such an unintuitive and complex way to deal with marks that it's better to leave things as they are without trying to modify their operation; the tricks we introduced are already sufficiently tricky.

The page number is in the central position of the header. Sticking into the external margin end aligned with the header base there is a thumbnail that carries the initial capitalized letter of the "chapter" entries. The `\chapter` command already contains the mechanism for setting the internal command `\unghia` (thumbnail) so as to have already available the required information.

LaTeX complexity is severely stressed by the two column format of the dictionary; in spite of loading the fixing fixltx2e package, when a long entry description goes on for several pages, it may happen that "strange" numbers appear in front of the left side of a header data, instead of being identical to the previous page

last right side header data; in order to fix this "feature" it was necessary to put together a fix even more complicated and unintuitive. To be more precise the data inserted in the right part of the first page header of a long entry goes into the le left half of a token register called `\topmark`, wile the same data that appear in right side of the header of any subsequent page of the long entry is stored in the left part of another token register called `\botmark`; we therefore fetch both data from these two token registers and store them into the auxiliary macros `\@tempA` and `\@tempB` paying attention non to expand the expandable macros possibly contained in such data (they might bring the typesetting program into an infinite loop that saturates one of the internal TEX storage areas!).

This done, it is possible to compare the contents of these auxiliary macros `\@tempA` and `@tempB`; if they are equal, then the header definition must receive the `\leftmark` instead of the `\rightmark`, so we are sure that the header information is consistent; if they are different we can maintain the normal `\rightmark`.

While tackling these inconsistencies we found that the "strange" number presence was tied to some missing equivalence statements necessary to "sanitize" the commands `\chaptermark` and `\sectionmark`; by adding these missing equivalence commands the "strange" numbers disappeared. On the other side the mark management is disturbed by the floating object management; therefore occasionally the data in the right part of a header might be equal to the first entry of the next page; occasionally the data on the left side of a header points to the second entry on the page. But as the dictionary entries get added the number of these "disturbances" diminishes but it impossible to be 100% sure that they never show up again.

```
\newcommand*\getcorrectrightmark{\let\protect\noexpand
\edef\@tempA{\expandafter\@leftmark\topmark\@empty\@empty}%
\edef\@tempB{\expandafter\@leftmark\botmark\@empty\@empty}%
\let\protect\relax\ifx\@tempA\@tempB\leftmark\else\rightmark\fi}
%
\def\ps@dizionario{\let\@mkboth\@gobbletwo
  \let\chaptermark\@gobble \let\sectionmark\@gobble
  \def\@oddhead{\underline{\makebox[\textwidth]{\strut
    \bfseries\makebox[\z@][l]{\getcorrectrightmark}\hfill
    \thepage\hfill\makebox[\z@][r]{\leftmark}\begin{picture}(0,0)%
    \put(20,-2){\makebox(0,0)[lb]{\HUGE\sffamily\mdseries\unghia}}%
    \end{picture}}}}%
  \def\@evenhead{\underline{\makebox[\textwidth]{%
    \begin{picture}(0,0)\put(-20,-2){\makebox(0,0)[rb]{\HUGE\sffamily
    \mdseries\unghia}}\end{picture}\strut\bfseries\makebox[\z@][l]%
    {\getcorrectrightmark}\hfill\thepage\hfill\makebox[\z@][r]{\leftmark}}}}%
  \def\@oddfoot{}%
  \def\@evenfoot{}%
}
\let\ps@dictionary\ps@dizionario
```

At the same time it is necessary to avoid the explicit setting of the dictionary page style before starting the dictionary proper; we can exploit the `\mainmatter` and `\backmatter` commands so that the setting variation get done behind the

scenes. Therefore `\mainmatter` shall be changed in such a way as to set the dictionary page style also for the initial chapter page, and `\backmatter` to set the normal headings style and the plain style for the chapter initial page.

```
\renewcommand*\mainmatter{%
  \if@twoside\cleardoubleoddpage\else\clearpage\fi
  \@mainmattertrue\pagenumbering{arabic}%
  \pagestyle{dictionary}%
  \renewcommand*\chapterpagestyle{dictionary}%
  \twocolumn
}
\renewcommand*\backmatter{%
  \if@openright\cleardoubleoddpage\else\clearpage\fi\@mainmatterfalse
  \pagestyle{headings}
  \renewcommand*\chapterpagestyle{plain}
}
```

9.12 Customisation

As we have already said, chapter and section numbers should never appear, not even in the numerable objects numbering. But equation and footnote numbers (that by default would be reset at every new chapter must be reset at every new entry; therefore it is necessary to take them away from the chapter reset list and to add them to the section reset list. For adding a counter to a reset list is easy, since it suffices to use the LaTeX kernel command `\@addtoreset`; the opposite command does not exist in the LaTeX kernel nor in the inner `scrbook` class, and it's necessary to define it; we decided to copy its definition from Peter Wilson's `memoir` class, which is a special application of the stack management described in the TeXbook:

```
\providecommand{\@removefromreset}[2]{{%
  \expandafter\let\csname c@#1\endcsname\@removefromreset
  \def\@elt##1{%
    \expandafter\ifx\csname c@##1\endcsname\@removefromreset
    \else
      \noexpand\@elt{##1}%
    \fi}%
  \expandafter\xdef\csname cl@#2\endcsname{%
    \csname cl@#2\endcsname}}}
```

This done it's trivial to remove counters `equation` and `footnote` from the `chapter` counter reset list and to add them to the `section` one:

```
\@removefromreset{equation}{chapter}
\@addtoreset{equation}{section}
\@removefromreset{footnote}{chapter}
\@addtoreset{footnote}{section}
```

We redefine the figure, table and equation typesetting strings, so as not to contain any reference to other counters. We take the occasion to customize also some vertical spacing and to kill any other sectioning command format.

```
\renewcommand*\theequation{\@arabic{\c@equation}}
\renewcommand*\thefigure{\@arabic{\c@figure}}
\renewcommand*\thetable{\@arabic{\c@table}}
\textfloatsep=\intextsep
\dbltextfloatsep=\intextsep
\renewcommand*{\othersectionlevelsformat}[1]{}%
```

We must remark that footnotes are not very common in dictionaries; nevertheless there might be some complicated entry that requires extra explanatory remarks so that we do not exclude a very rare footnote usage.

9.13 New useful commands

Captions do not require any numbering and may be located either before or after the picture they are referred to; an arrow tip (actually a very visible black triangle) points to the object the caption refers to. Notice the `\scaption` does not use any triangle and should never be used; `\scaptiona` is intended to be put *a*bove the picture and its arrow points downwards; `\scaptionb` *b*elow the picture and its arrow points upwards; captions are typeset with the `\small` font size.

The vertical spacing before and after the various captions are all set to `\medskipamount` instead of the default 10 pt; just one of the two spacings is set and the other is set to zero: it depends on whether the caption is above or below the picture and the inner `scrbook` takes care of setting the right spacing.

Among the inner class options we did set the option `captions=nooneline`, so that even those captions that are shorter than one line are typeset flush left as the longer captions. The caption position was obtained by means of the inner class commands `\captionabove` and `\captionbelow`; in any case, but only for the current figure or table, the commands `\figureformat` and `\tableformat` have been neutralized so that they do not print anything, but this was done on purpose only locally in order to open the possibility of numbering tables and figures.

We wanted these new captioning commands capable of accepting the optional argument as well as the standard command `\caption` does; in this case the link anchor may be constructed in such a way as to deal only with the "short caption", that may be composed without any expandable command so that the hyperlink mechanism is not disturbed by such macros.

The dimension `\vercapskip` is defined and used to make a better caption positioning, especially with upper caption so when they appear at the top of a column the don't appear to be slightly raised and the first or only caption line remains aligned with the first line of the adjacent column. Similarly the last caption line below a picture is aligned with the last line of the adjacent column. This correction is not applied with command `\scaption`: another reason to avoid it.

```
\addtokomafont{caption}{\small}
\setlength{\abovecaptionskip}{\medskipamount}
\setlength{\belowcaptionskip}{\z@}
\newskip\vercapskip
%
\newcommand\scaption[2][]{\def\sc@ption{#1}%
```

```
\renewcommand*{\figureformat}{}%
\renewcommand*{\tableformat}{}%
\setlength{\belowcaptionskip}{\medskipamount}%
\renewcommand*{\captionformat}{}%
\ifx\sc@ption\@empty\caption{#2}\else\caption[#1]{#2}\fi}
%
\newcommand\scaptiona[2][]{\def\sc@ptiona{#1}%
\renewcommand*{\figureformat}{}%
\renewcommand*{\tableformat}{}%
\renewcommand*{\captionformat}{\DOWNarrow\quad}%
\small\vercapskip\dimexpr\ht\strutbox-1ex\relax
\vspace*{\vercapskip}%
\ifx\sc@ptiona\@empty\captionabove{#2}\else\captionabove[#1]{#2}\fi}
%
\newcommand\scaptionb[2][]{\def\sc@ptionb{#1}%
\renewcommand*{\figureformat}{}%
\renewcommand*{\tableformat}{}%
\renewcommand*{\captionformat}{\UParrow\quad}%
\small\vercapskip\dp\strutbox
\ifx\sc@ptionb\@empty\captionbelow{#2}\else\captionbelow[#1]{#2}\fi
\vspace*{-\vercapskip}}
```

Long structured entries may receive a sort of `minitoc`, a mini table of contents that lists the first two levels of the overall structure; it's the end user duty to provide the correct information for this `minitoc`; there is nothing automatic as it would be with the minitoc package. Remember we are typesetting a dictionary, and a dictionary does not have a table of contents, so that the standard `minitoc` inner workings cannot be used.

In this `minitoc` the first level entries are introduced with command `\mttitle` and are typeset with a black square preceding them; the second level entries are introduced with command `\mtsubtitle` and are typeset with a small indentation and with a right pointing black triangle. The `minitoc` is typeset with a horizontal rule before and another one after its body. These rules are typeset with the definitions contained in macros `\upperline` and `\lowerline`, but the end user should not care about them, unless s/he wants to use these commands for her/his own purpose. A third command `\mtssubtitle` may be used in the entry body to introduce third level sectioning, but it must not be used in the `minitoc` body.

The `minitoc` is realized by means of the `mintoc` environment. This is a `minipage` environment embedded within a `list` environment, whose typesetting parameters have been all set to zero; the `minipage` is used to avoid a page or column break within the `minitoc` body; the `list` environment assures that `minitoc` starts on a new line, irrespective if it directly follows a text line or a blank one, and to avoid spurious spaces before and after the `mintoc` body.

```
\DeclareRobustCommand*\rigaalta{\mbox{}\par\noindent
  \raisebox{0.3ex}[0pt][0pt]{\rule{\columnwidth}{0.2ex}}\par}
  \let\upperline\rigaalta
\DeclareRobustCommand*\rigabassa{\par\noindent
  \raisebox{0.9ex}[0pt][0pt]{\rule{\columnwidth}{0.2ex}}\par}
```

```
  \let\lowerline\rigabassa
\newcommand*\titoletto[1]{\par\noindent
  \hspace{4mm}\RIGHTarrow\space\textit{#1}}
  \let\mtssubtitle\titoletto
\newcommand*\titolo[1]{\par\noindent
  \hspace{1.5mm}\rule{1.6mm}{1.6mm}\enspace\textit{#1}}
  \let\mttitle\titolo
\newcommand*\titolino[1]{\par\noindent
 \hspace{2mm}\textbullet\enspace\textit{#1}}
 \let\mtsubtitle\titolino

\newenvironment{minitoc}{%
\list{}{\topsep\z@\itemsep\z@\parsep\z@\leftmargin\z@\rightmargin\z@
\labelsep\z@\itemindent\z@\listparindent\z@}%
\item\minipage{\columnwidth}\vspace*{-.6\baselineskip}\rigaalta
}{%
\rigabassa\endminipage\endlist
}
\let\miniindice\minitoc
\let\endminiindice\endminitoc
```

In astronomy temperatures are often used as well as angles; here we produce a raised small circle that comes handy in both cases; but it's necessary to recall that: (*a*) the raised small circle should not be separated from the numerical value when it means "angular degrees", for example: 90°; (*b*) the Celsius indication must be spaced with an unbreakable space `\,` from its numerical value, for example: 5000 °C; (*c*) taking into consideration the high values of star temperatures and their nature it would be much wiser to use the kelvin unit, indicated by K without any raided circle, for example: 5000 K; (*d*) when using Italian as the main language the command `\unit` allows to tyepset correctly and with the correct unbreakable spaces: `90\degree` produces 90°; `100\celsius` produces 100 °C; `3200\kelvin` produces 3200 K; `13\unit{h} 21\unit{min} 14\unit{s}` produces 13 h 21 min 14 s. With other languages, by using the Slunits package syntax, the above examples should be: `\unit{90}{\degree}`, `\unit{13}{\hour} \unit{21}{\minute} \unit{14}{\second}`, etc. Use `\unita` in place of `\unit` if the main language is Italian since this babel options defines `\unit` with a different syntax, as shown above.

Nevertheless in astronomy sexagesimal angles are used so often that we defined the command `\degr` to be directly attached to the angle measure, so that spacings are right and that it may be used also in text mode with the current font.

```
\newcommand*\gradi{\textormath{\textdegree}{^\circ}}
\let\degr\gradi
```

By an explicit request, the dictionary is completely typeset without paragraph indentation; most of the time this is not influential in distinguishing the start of a new paragraph.

```
\parindent=\z@
```

The new command `\diff` typesets the differential symbol as requested by the ISO regulations in all applied sciences, that is in upright font and with the proper spacing at its left.

```
\def\diff{\mathop{}\mathclose{\mathrm{d}}}
```

The new command `\BackgroundPicture` is used to define a background picture over which it's possible to typeset other things; this is useful for the title page where the author, the title and other smaller pieces of information are typeset over a full page picture.

```
\newcommand\BackgroundPicture[2]{%
  \setlength{\unitlength}{1pt}%
   \put(-9,\strip@pt\paperheight){%
  \parbox[t][\paperheight][c]{\paperwidth}{%
    \centering\includegraphics[angle=#2,width=\paperwidth]{#1}\par
}}}%
%
\newcommand*{\pointsto}{\textrightarrow}\let\freccia\pointsto
\newcommand*{\longpointsto}{$\longrightarrow$}\let\freccialunga\longpointsto
\newcommand*\Endash{\textrm{\ \textendash\ }}
```

Now a simple command for en-dash delimited parenthetical material:

```
\newcommand\insertremark[1]{\textendash\,#1\,\textendash}
\let\inciso\insertremark
```

Since this definition is done by means of the LaTeX kernel command `\newcommand` without asterisk, the definition is a "long" one and accepts as its argument even a text several paragraphs long; use with caution and don't forget the closing brace.

9.14 Index

The index is produced by default; to this end the command `\makeindex` is already inserted in this class file. Another feature is that the index is produced during the same run as the main document and is input and typeset at the end of the document by means of the commands described hereafter; if you do not want to produce an index, do not use such commands in the source file. In order to typeset the index in one run, it is necessary the the shell editor, with which the source file is edited, launches the typesetting engine *pdflatex* with the `-shell-escape` or the `-enable-write18` option (depending on the particular distribution and implementation of the typesetting engine). With the 2010 distribution of TeX Live the program is automatically enabled to run *makeindex*. The 2.9 version of the MiKTeX distribution might be similarly enabled; in any case if the typesetting engine is not enabled by default it suffices to add the above option to the launching command string of one's text editor; read your text editor documentations in order to find out how to add that option to the launching command. After running *pdflatex* on any source file, go end open its `'log` file and examine the first, say 15 lines: if a line says `\write18 enabled`, your *pdflatex* is OK; otherwise either you succeed enabling your program as said above, or you have to ask a "guru", or you have to manually run *makeiindex* yourself as it was customary in the past years, since LaTeX was born in 1984.

It's not a case that the next command is "possibly sort the index files"; if your typesetting program is not enabled this command does not do anything; similarly if the sorted index file does not exist, the subsequent command just warns that the file is not available; but it does not give any warning if the file is available and is obsolete compared with the unsorted index file. So if you have to produce any index by yourself, pay attention to sort your raw index file by means of *makeindex* otherwise your index is not in sync with your document.

A last remark on indices: the automatic sorting of the row index file is done according to the `indice.ids` index style; if you sort your raw index file by hand, assuming your dictionary main file is named `mydictionary.tex` give this command from the terminal window o command prompt window, *without using your shell editor* Makeindex *button*:

```
makeindex␣-s␣indice.ist␣mydictionary.idx
```

The third phase connected with the production of the index is `\printindex`; this command together with the previous two ones always produces an up to date index provided the typesetting program is enabled to run *makeindex*. If it is not enabled it's the end user responsibility to run *makeindex* at least after the last definitive version of the dictionary.

```
\def\possiblysortindexfiles{\immediate\closeout\@indexfile
\immediate\write18{makeindex -s indice.ist \jobname.idx}}

\makeindex

\providecommand\printindex{%
\IfFileExists{\jobname.ind}{\onecolumn\input{\jobname.ind}}%
  {\ClassWarning{dictionarySRC}{File \jobname.ind is not available}}}
```

We redefine also the `index` environment so that it uses the multicol package; in this way the last index page in two columns has balanced columns. Since there is no table of contents, even if it was possible, we did not add the "Index" title to the `.toc` file.

```
\renewenvironment{theindex}
   {\let\@chapapp\empty
\let\thechapter\relax
        \def\see##1##2{\emph{\seename} ##1}
\chapter*{\HUGE\indexname}%
\@mkboth{\indexname}{\indexname}%
\thispagestyle{plain}\parindent\z@
\parskip\z@ \@plus .3\p@\relax
\columnseprule \z@
\columnsep 15\p@
\raggedright
\let\item\@idxitem
\begin{multicols}{2}}
   {\end{multicols}\clearpage}
```

The LaTeX kernel command `\cleardoublepage` is redefined so as to accept an optional argument that specifies the style of the possible blank page that is going

to be output in order to restart typesetting on an odd page. The default style is `plain` but it's possible to specify `empty`.

```
\renewcommand\cleardoublepage[1][plain]{%
\clearpage\ifodd\value{page}\else\thispagestyle{#1}\null\clearpage\fi}
```

9.15 Bibliography

Even the `bibliography` environment is being revised; it is being redefined so as to insert the "bibliography" name in the headers, if the page style has non empty headers. Without this change the `bibliography` environment is practically identical to the original one. The initial page style for the bibliography is the `plain` one, but the whole bibliography might be enclosed in a `myhedings` page style so that headers appear normally.

Note that this class package contains the `plain_ital.bst` wich gives an national look to the plain bibliography style, by setting the author names in caps and small caps, but also sets the infix words in Italian (for example *curatore* in place of *editor* and other similar replacements). Of course any author will use the bibliography style s/he likes best. The `plain_ital.bst` is the default one only if the main language is Italian.

Notice also that the end user must provide the bibliographic database with extension `.bib` according to the BibTeX syntax and s/he must process this database together with the preferred bibliography style before producing the final dictionary version; we do not provide any command such as we did for the index, that tests if the necessary processed file with extension `.bbl` exists.

After all the end user might produce the bibliography by him/herself and to store the information in a `.tex` file. S/he is free to format every entry the way s/he likes best, although a consistent style is much preferable to hand made solutions; should s/he chose this way, the input command would be `\input{mybiblio}` (with an assumed `.tex` extension) and a possible `\printbiblo` that assumes a `.bbl` extension would be out of order.

```
\AtEndOfClass{\AfterEndPreamble{%
\ifnum\language=\csname l@italian\endcsname
   \bibliographystyle{plain_ital}%
\fi}}
\renewenvironment{thebibliography}[1]{%
\chapter*{\HUGE \bibname}
\markboth{\bibname}{\bibname}
\thispagestyle{plain}
  \list{%
    \@biblabel{\@arabic\c@enumiv}%
  }{%
    \settowidth\labelwidth{\@biblabel{#1}}%
    \leftmargin\labelwidth
    \advance\leftmargin\labelsep
    \@openbib@code
    \usecounter{enumiv}%
    \let\p@enumiv\@empty
```

```
742     \renewcommand*\theenumiv{\@arabic\c@enumiv}%
743   }%
744   \sloppy\clubpenalty4000 \widowpenalty4000
745   \sfcode`\.=\@m\raggedright
746 }{%
747   \def\@noitemerr{%
748     \@latex@warning{Empty `thebibliography' environment}%
749   }%
750   \endlist
751 }
```

9.16 Hanging tables

As mentioned in the Introduction we thought it was possible to use the external margin, at least as much as marginal notes do, to let some tables hang out of the external column, specifically those tables that are larger than `\columnwith` but not that large to be suited to the whole `\textwidth`. We thought that this would have been a better solution than scaling down a largish table to fit within a column; characters in normal size are already small; shrinking would make them too small for a comfortable reading.

The problem with the external margin is that only the external column could contain a hanging table; the code for hanging tables is not difficult but their placement into the external column is prohibitive due to the asynchronous way the typesetting engine uses to form pages with their columns. Therefore it is practically impossible to know in which page and in which column a particular text stretch will fall in. The output routine contains a switch that is set according to the column ordinal number, but this switch is not accessible before the output routine is running. Also the page number can be known only during a second run, if a proper label has been set in the previous run.

Even the command `\afterpage` is of no help; possibly a different command such as `\afteroddpage` or `\afterevenpage` would be of some help if they existed and if they worked also in two column mode. On the opposite, the command `\newpage` in two column mode just breaks the current column, not necessarily the current page.

It is relatively easy, through a suitable use of internal labels, to know if a certain text stretch falls in a recto or a verso page, but the information is available only after the next compilation run. For what concerns the column there is no hope; at least we could not find any suitable mechanism.

We found two manual solutions to this problem; after the results obtained in the first run the end user may decide to maintain the source code or to cut it and paste it a few paragraphs before or after the original position.

Of these two solutions one is fixed and the other is floating; for both solutions the side where the table should be hanging into is determined automatically; with a fixed table containing a numbered caption the risk is to have it output before a preceding floating table. Both solutions, therefore, require some manual

adjustment. It's disagreeable, but up to date we could not find a better solution; manual adjustment, though, does not seem to be too heavy.

First we must declare some dimensional and box registers in order to make the necessary computations.

```
\newlength\widet@ble
\newlength\wid@t@ble
\newlength\widet@blesh
\newbox\widebox
```

Then we start with the definition of command `\SetTableIntoExternalColumn` that receives just one complex argument, the captioning command, a possible label set up by the composer, and the whole `tabular` environment complete with its opening and closing statements. An environment would probably be easier to use, but we kept this "command" solution that was working pretty well during our first dictionary class versions. Having now available the next floating solution, we thought the end user would not use this command any more, but we kept it for backwards compatibility.

We start to compute the total width available, by summing up the column width, the marginal note width and the marginal note separator width; this represents the maximum width a hanging table may have, and if the table is wider, a message is output, but the table will be typeset anyway, so the composer can evaluate if it's acceptable, or if it's better to set the table in full width. If the actual width does not exceed this maximum, the table is set flush left in a recto page and flush right in a verso one. But in order to save the whole argument, its necessary to freeze the `\caption` and `\label` commands; they will be made active again when the tabular material will be actually output, but in the first steps we need just to measure the the tabular material width; to this end we keep this material within an `lrbox`. With the actual width available it's possible to compute the necessary horizontal shifts necessary to flush left or right the table and finally the box containing the tabular material and the `\caption` and `\label` unfrozen commands, we can output the whole table with its caption and generate the possible label information. The freezing of the `\caption` command works well if there is no optional argument to this command.

```
\newcommand\SetTableIntoExternalColumn[1]{%
\begingroup
    \widet@ble=\dimexpr\columnwidth+\marginparsep+\marginparwidth\relax
    \begin{lrbox}{\widebox}\let\caption\@gobble
    \let\label\@gobble#1\relax
    \end{lrbox}%
    \ifdim\wd\widebox>\widet@ble
       \ClassWarning{dictionarySCR}{The hanging table is too wide!}
       \wid@t@ble=\widet@ble
       \widet@blesh=\z@
    \else
       \wid@t@ble=\wd\widebox
       \widet@blesh=\dimexpr\widet@ble -\wid@t@ble\relax
    \fi
```

After measuring the necessary widths the tabular material is typeset again within the `\widebox` box now used as a vertical box with top alignment `\vtop`; in order to have a correct alignment we set an initial unbreakable vertical space, then a rule across the whole box width, then add a negative vertical space to compensate the vertical advancement that the inner `scrbook` class insists putting over the caption, even if it was declared that captions have to be set over the tabular material; then the whole macro argument with the unfrozen `\caption` and `\label` commands and the complete tabular material; then we add some more vertical space and a rule across the whole box.

```
    \setbox\widebox\vtop{\hsize\wid@t@ble\parindent\z@
    \textwidth=\hsize\columnwidth=\hsize\linewidth=\hsize
    \def\@captype{table}\vspace*{2ex}\hrule
    \vskip-.5\abovecaptionskip\relax#1\par\hrule\vspace*{1ex}}%
```

Now we generate a strut of zero height and depth equal to the `\widebox` depth; this strut has a total height matching the box total height, but it has zero width and does not interfere with the box placement; it just leaves the necessary space in the column being typeset. The command `\marginnote` uses this vertical space by putting to the left (optional argument) or to the right (compulsory argument) another box as wide as the marginal note containing the `\widebox` box hanging to the right or to the left by the the distance necessary to have it perfectly aligned with the internal margin of the external column: these boxes contain mysterious lengths that we found necessary in order to fine tune the contents position required after testing the macro effectiveness. It must be noticed that the parity of the page number is handled by the `\marginnote` macro. Even if marginal notes are slightly movable, the way they are used here make the positioning of the table fixed to the position of the strut.

```
    \rule[\dp\widebox]{\z@}{\z@}% strut
    \marginnote
    [\mbox{}\hspace*{.4em}\hspace*{\widet@blesh}\rlap{\box\widebox}]%
    {\mbox{}\llap{\box\widebox}\hspace*{.4em}\hspace*{\widet@blesh}}%
    [-\dp\widebox]%
\endgroup}
```

For the floating hanging table we need an internal macro `\PaginaTabella` in order to know the page number parity; this is obtained by fetching the second argument of the five grouped pieces of information that the hyperref package creates for the cross reference inner workings; but this number is known after the page has been already output, so this approach requires two runs of the typesetting program; on the first run the information is not available so tat the macro tries to guess it from the current page number, which generally is either correct or ± 1 the true value; we have a 50% chance to guess it right already at the first run, but it's better not to trust an initial good guess.

```
\newcount\t@bella \t@bella=\z@
\newcount\p@gina
\def\PaginaTabella#1{\ifx#1\relax\value{page}\else
\expandafter\diz@secondoffive#1\fi}
```

For compatibility with the `tabular` environment, we define a new one with a syntax very similar to it. With the functionality of the internal label the right or left setting of the table can be determined in an automatic way; the internal label requires a numerical counter `\t@bella` (this is a TEX not a LATEX counter) the value of which, stepped as each table is being processed, is used to generate the unique internal label name; the TEX counter `\p@gina` is just instrumental to help making the internal computations. The environment defined hereafter is to be used within a `table` environment so that the captioning and labeling is done outside this new environment; the `table` environment keeps it floating, though.

The new environment `Wtabular` is therefore used exactly as `tabular` and the identical syntax makes them compatible even if `tabular` contained some optional argument. The internal macro `\@tempB` is used to keep the internally generated label name; then the tabular material, whose column descriptors are listed in the only compulsory `Wtabular` argument, is being typeset and kept within the `\widetable` box by means of the `lrbox` environment.

The closing environment commands retrieve the page number by executing the command `PaginaTabella`; if it's the first time the environment is being typeset the page number is assumed to be that of the current page but, from the second run on, that internal label contains the required correct information and everything is processed as it should. If the tabular material is too wide, a warning is issued but no other special action is taken. The fine work done by the internal label operations allows to determine the horizontal alignment of the tabular material; in facts the macro `\@tempA` is being defined according to the parity of the actual page number and the tabular material is output within a `\makebox` box with the suitable alignment code. The only glitch of this hanging table environment is that a possible caption is typeset justified with respect to the column width; if its desired to typeset the caption as wide as the tabular material, then it should be inserted in the first tabular row, by means of a `\multicolumn` command and a `p...` descriptor, and by adjusting the necessary space below the first row.

```
\newenvironment{Wtabular}[2][c]{%
   \widet@ble\linewidth
   \advance\widet@ble\columnsep
   \advance\widet@ble\marginparwidth
   \global\advance\t@bella\@ne
   \edef\@tempB{dzt@\the\t@bella}%
   \expandafter\label\expandafter{\@tempB}%
   \begin{lrbox}{\widebox}%
   \begin{tabular}{#2}%
   }{%
   \end{tabular}\end{lrbox}%
   \p@gina=
      \expandafter\PaginaTabella%
            \expandafter{\csname r@\@tempB\endcsname}\relax
   \ifodd\p@gina
     \def\@tempA{l}
   \else
     \def\@tempA{r}
```

```
   \fi
   \ifdim\wd\widebox>\widet@ble
      \ClassWarning{dictionarySCR}{Table is too wide!}%
   \fi
   \makebox[\columnwidth][\@tempA]{\box\widebox}%
   \ignorespaces}
```

9.17 Colored tables

For tables with shaded background we defined the new environments `shadedtabular` and `shadedWtabular`. We did not do anything for shading the hanging table typeset by means of `\SetTableIntoExternalColumn`, which remains available only for unshaded tables.

The need to have ad hoc environments, without directly using the `shaded` environment, defined by the already loaded framed package, derives from the fact that this shading environment shades the full text or column width, while we want to shade only the table. This implies that we have to typeset the table within a box, take the box measures, and then create a shaded box of the same dimensions as the table box, that, on turn, will be laid over the background colored box.

For both environment the background color is 'straw'; but the optional argument allows to chose any other desired color already defined by this class or defined by the user by means of the `\definecolor` command.

```

\newenvironment{shadedtabular}[1][straw]{% "paglierino"
\setlength{\FrameSep}{0pt}%
\def\FrameCommand{\fboxsep=\FrameSep \colorbox{#1}}% from "shaded"
\setbox254\hbox\bgroup\tabular}{%
\endtabular\egroup
\centering
\begin{MakeFramed}{\hsize\wd254 \FrameRestore}%
\box254\relax
\end{MakeFramed}\vspace*{-.75\baselineskip}\ignorespacesafterend}%

\newenvironment{shadedWtabular}[2][straw]{% "paglierino"
   \setlength{\FrameSep}{0pt}%
   \def\FrameCommand{\fboxsep=\FrameSep \colorbox{#1}}% from "shaded"
   \widet@ble\linewidth
   \advance\widet@ble\columnsep
   \advance\widet@ble\marginparwidth
   \global\advance\t@bella\@ne
   \edef\@tempB{dzt@\the\t@bella}%
   \expandafter\label\expandafter{\@tempB}%
   \begin{lrbox}{\widebox}%
   \begin{tabular}{#2}%
   }{%
   \end{tabular}\end{lrbox}%
   \p@gina=\expandafter\PaginaTabella
                        \expandafter{\csname r@\@tempB\endcsname}\relax
```

```
  \ifodd\p@gina
    \def\@tempA{l}%
  \else
    \def\@tempA{r}%
  \fi
  \ifdim\wd\widebox>\widet@ble
     \ClassWarning{dictionarySCR}{Table is too wide!}%
  \fi
  \setbox254\vbox{\hsize\wd\widebox
  \begin{MakeFramed}{\hsize\wd\widebox \FrameRestore}%
    \box\widebox\end{MakeFramed}\vspace*{-.75\baselineskip}}%
  \makebox[\columnwidth][\@tempA]{\box254}%
  \ignorespacesafterend}

```

9.18 Catalogs

We now prepare the macros for entering the "catalogs"; as we said in the Introduction these are simply long tables, very often `\textwidth` wide, concerning any discipline; since this dictionary class was prepared for astronomy, it's pretty obvious we will exemplify with star catalogs and the macros for setting up their coordinates. The idea is the same for other thematic long tables, but the possibly necessary extra macros have to be set up by the end user; if users want to contribute other macro sets suitable for other disciplines, we would be glad to acknowledge them and add them to a "Contributed" sub folder or give the links to other packages.

We start with the environment `catalogs`, a very normal environment that changes the page style to `myheadings` and sets the typesetting mode to `\onecolumn` while memorizing the possibility of reverting to `\twocolumn` mode upon exiting the environment. The only unusual thing might be the boolean variable `at@twocolumn`; it is used by the inner `scrbook` to decide how to typeset chapter titles, whether at full width or in one column mode. Upon opening the environment we set also the boolean variable `@restonecol` that allows setting or resetting one column mode when closing the environment.

```
\newenvironment{cataloghi}{\clearpage
\pagestyle{myheadings}%
\@at@twocolumnfalse
\if@twocolumn
\@restonecolfalse
\onecolumn
\else
\@restonecoltrue
\fi
}{\if@restonecol\onecolumn\else\twocolumn\@at@twocolumntrue\fi}
\let\catalogs\cataloghi
\let\endcatalogs\endcataloghi

```

The definition of macro `\insertcatalog` comes next; it wants three arguments the first of which is optional, with an empty default value; it allows to typeset a colored long table rather than a black and white one; any optional argument, different from an empty string, is good for setting up the colored table; we suggest you use the word `color` but any non empty string will do. The full syntax is:

`\insertcatalog[`⟨*colored*⟩`]{`⟨*file name*⟩`}{`⟨*header contents*⟩`}`

The simple macro is:

```
\newcommand\insertcatalog[3][]{\clearpage
\begingroup
\markboth{#3}{#3}
%
\def\@tempB{#1}\ifx\@tempB\empty
    \def\RCi{\relax}\let\RCuno\RCi%
    \def\RCii{\relax}\let\RCdue\RCii%
    \def\RCiii{\relax}\let\RCtre\RCiii%
    \def\RCiv{\relax}\let\RCquattro\RCiv%
    \def\CCyellow{\relax}\let\CCgialla\CCyellow%
    \def\CCYellow{\relax}\let\CCgiallone\CCYellow
    \let\RCodd\relax \let\RCdispari\RCodd
    \let\RCeven\RCiv \let\RCpari\RCeven
    \let\CC\CCYellow
    \newcolumntype{L}{l}%
    \newcolumntype{R}{r}%
    \newcolumntype{C}{c}%
    \newcolumntype{M}{>{$}r<$}%
    \newcolumntype{A}{>{\RA}r}%
    \newcolumntype{B}{>{$}l<$}%
\else
    \def\RCi{\rowcolor{azzurrino}}\let\RCuno\RCi%
    \def\RCii{\rowcolor{giallochiarissimo}}\let\RCdue\RCii%
    \def\RCiii{\rowcolor{aranciochiarissimo}}\let\RCtre\RCiii%
    \def\RCiv{\rowcolor{paglierino}}\let\RCquattro\RCiv%
    \def\CCyellow{\cellcolor{giallochiaro}}\let\CCgialla\CCyellow%
    \def\CCYellow{\cellcolor{giallocarico}}\let\CCgiallone\CCYellow
    \let\RCodd\relax \let\RCdispari\RCodd
    \let\RCeven\RCiv \let\RCpari\RCeven
    \let\CC\CCYellow
    \newcolumntype{L}{>{\columncolor{giallochiarissimo}}l}%
    \newcolumntype{R}{>{\columncolor{giallochiarissimo}}r}%
    \newcolumntype{C}{>{\columncolor{giallochiarissimo}}c}%
    \newcolumntype{M}{>{\columncolor{giallochiarissimo}$}r<$}%
    \newcolumntype{A}{>{\columncolor{giallochiarissimo}\RA}r}%
    \newcolumntype{B}{>{\columncolor{giallochiarissimo}$}l<$}%
\fi
%
\tabcolsep=5.6pt
\edef\Tstrut{\vrule\@height\dimexpr\ht\strutbox+2\p@\relax\@width\z@}
\edef\Bstrut{\vrule\@depth\dimexpr\dp\strutbox+2\p@\relax\@width\z@}
```

```
\edef\TBstrut{\vrule\@depth\dimexpr\dp\strutbox+2\p@\relax
\@height\dimexpr \ht\strutbox+2\p@\relax\@width\z@}
\belowrulesep\z@ \aboverulesep\z@
%
\input{#2}
\endgroup}

```

The row and column default colors are specified in Italian, and explained in English in table 2 on page 21.If the user decides to change her/his mind concerning colors, s/he should only change the `color` optional argument to an empty string and needs not change anything else in the previous colored table settings.

We proceed with the definition of the command `RA` for typesetting the right ascensions of heavenly bodies; this macro is a delimited argument one, so the delimiters must be carefully specified; the delimiters are simple spaces, *including a trailing one.*

`\RA`␣⟨*hours*⟩␣⟨*minutes,tenths*⟩␣

⟨*hours*⟩ are specified with a one or two decimal digits in the range `00-23` (but no check is made on the upper limit); minutes are optionally followed by a decimal separator (a point or a comma) and a one digit value for their tenths; the integer part of the minutes value is written with a one or two digit number in the range `00-59` but, again non check is made on the upper limit.

The macro `\RA` is sort of complicated because it requires a lexical analysis of what is being input. Therefore:

- Hours are handled in such a way as to ignore a possible `\ignorespaces` that might precede the numerical value when the command is used as a table column modifier. The remaining number is assigned to the TEX counter `\ore` in order to check its value and the number of digits it contains.
- The second argument is assigned to a dimension register; irrespective of the presence of the optional separator and the tenths value, TEX does not care which separator is used, provided it is a point or a comma. The length value is stripped by its "pt" component by means of the LATEX kernel macro `\strip@pt`, and the result assigned to an auxiliary macro. Notice that `\strip@pt` does not save the fractional part if this is missing or it's zero. Therefore it's necessary to check the existence of a fractional part by means of the `\split@dec` internal macro; this macro loads the TEX counters `\minuti` and `\decimi` so as to have available all the required right ascension numerical components. As usual the minutes component is typeset with a leading zero if it consists of just one digit.

Notice that command `\RA` works well both in text and math mode; it's necessary to recall that in text mode the space after the minutes and the optional tenths is required and used by the macro itself; therefore any punctuation mark or any word must be separated by a further explicit space. You have to insert in the source file something such as `\RA␣17␣45,3␣.` or `{\RA␣17␣45,3␣}.`; with words

you need the the equivalent strings `\RA␣17␣45,3␣\␣and` or `{\RA␣17␣45,3␣}␣and`. You don't need to insert any unit of measurement, since they are correctly inserted by the macro according to the ISO regulations. The end user *must not* specify the `\RA` command when entering data into an `A` column in a catalog table: s/he will enter `... &␣17␣45,3␣& ...` paying attention to the compulsory spaces, without forgetting the trailing one just before the `&` sign.

```
\def\RA#1 #2 {% don't delete the space in front of the open brace
\begingroup
\countdef\ore 254\relax \countdef\minuti 252\relax
\countdef\decimi 250\relax
\let\ignorespaces\empty
\edef\@tempa{#1}
\ifnum\language=\l@english
  \unit{\expandafter\ore\@tempa\relax
    \ifnum\ore=\z@00\else
    \ifnum\ore>9\relax\else0\relax\fi\number\ore\fi}{\hour}\space
  \unit{\@tempdima#2\p@\edef\@tempa{\strip@pt\@tempdima}%
    \expandafter\split@dec\@tempa..\relax
    \ifnum\minuti=0\relax00\else
    \ifnum\minuti>9\relax\else0\relax\fi\number\minuti\fi\dizdecsep%
    \number\decimi}{\minute}
\else
\expandafter\ore\@tempa\relax
\ifnum\ore=\z@00\else
\ifnum\ore>9\relax\else0\relax\fi\number\ore\fi\unit{h}\space
\@tempdima#2\p@\edef\@tempa{\strip@pt\@tempdima}%
\expandafter\split@dec\@tempa..\relax
\ifnum\minuti=0\relax00\else
\ifnum\minuti>9\relax\else0\relax\fi\number\minuti\fi\dizdecsep%
    \number\decimi\unit{min}%
\fi
\endgroup}
\let\AR\RA
```

The service macro `\split@dec` is another one with delimited arguments separated by periods and terminated with `\relax`. The third argument is there to collect what is useless, while the first one retrieves the integer part of the minutes and the second one the optional tenths value, if these had a non zero value, but if they were zero or not specified, the macro call assures that some empty entities are there anyhow. It suffices to control the possible emptiness of the second argument; if it's empty it suffices to load zero into the TEX counter `\decimi`. The macro therefore is the following:

```
\def\split@dec#1.#2.#3\relax{\minuti=#1\relax \def\@tempb{#2}%
\ifx\@tempb\empty\decimi=0\else
\decimi=#2\relax\fi}
```

Here we eventually set the decimal separator used by `\RA` to separate the integer from the fractional part of the minutes value; this will be a point if the main language is English, or a comma otherwise. Here we are at the end of the class

and the `\AtEndOfClass` appears very pleonastic. Actually it looks forward in the case we add more functionality to this calls and add further code after this statement.

```
\AtEndOfClass{\AfterEndPreamble{%
\edef\dizdecsep{%
\ifnum\language=\l@english{\noexpand\ensuremath{\plaindot}}\else{,}\fi}}}%
```

And this concludes the class description.