Language management and patterns for line breaking

Claudio Beccari

Abstract

Language management is supported by different files according to the language manager babel or polyglossia; they are similar to a certain extent, but differ in the way they handle the language patterns. There are also small differences when using XeLaTeX compared to LuaLaTeX. Obviously patterns are different from language to language, but there are also some languages with variants. Therefore the language-supporting compiler-structure has to manage a variety of situations.

Sommario

La gestione delle lingue è sostenuta da file differenti a seconda che si usi il gestore babel o polyglossia; gli specifici file di ogni lingua sono abbastanza simili ma differiscono specialmente nel modo di gestire i pattern per la cesura in fin di riga. Ci sono anche delle piccole differenze se si usa XeLaTeX o LuaLaTeX. I pattern, ovviamente, sono diversi da lingua a lingua, ma ci sono anche lingue che hanno anche delle varianti. Pertanto il meccanismo di sostegno linguistico dei compilatori deve essere in grado di gestire una varietà di situazioni.

1 Introduction

This paper is half way between a technical document and a tutorial. Some parts are quite professional, and others are for any LaTeX user.

Any LaTeX user knows very well that an important part of the preamble of any source `.tex` main file is dedicated to the language management of the only language used in the document, or to the specification of the various languages that are used in the document; very often, depending on the language(s), it is necessary to specify the font collections to be used in order to use different alphabets. The only users who do not need to specify anything used to be the American ones, because the default language is the American variety of the English language; of course, if they have to cite any stretch of text in a language different from English, they have to do as anybody else. The British, Australian, Canadian, New Zealander English authors have to specify their language variety because their varieties require a little different handling compared to the American one.

The situation is quite different when the document is typeset with pdfLaTeX, compared to what is possible to do with XeLaTeX and LuaLaTeX. In facts pdfLaTeX can use only the babel package, while when XeLaTeX or LuaLaTeX are used it is possible to require either the polyglossia or the babel package.

The babel package, (Braams e Bezos, 2018) has its origin in the early nineties when Johannes Braams invented its mechanism and became its maintainer. After some twenty years the maintenance was taken over by Javier Bezos who introduced many extensions and functionalities.

The polyglossia package (Charette e Reutenauer, 2018) was created with the advent of XeLaTeX and was refined a little bit when LuaLaTeX was stabilised. Now polyglossia can be considered quite stable, and any addition or modification is mostly related to the numerous specific language files.

babel and polyglossia handle a large number of languages, more than eighty, although for some rare ones there might be some differences. According to my experience, the language handling by these two packages offers some pros and cons, so that it is difficult to say which is the most performant. Personally I prefer polyglossia, but I use pdfLaTeX quite often, therefore, possibly, I use babel more often than polyglossia. The differences are described in detail in the following sections.

Both handlers have to resort to language specific files in order to define (*a*) the infix words necessary to typeset language dependent headings, the date, and some other language specific structures; and (*b*) they have to select the proper *hyphenation patterns* in order to let the typesetting engine split in the best way the input strings so as to create perfectly justified paragraphs, with the least number of hyphenated line breaks and, with the support of the microtype package (Schlicht, 2018), the most homogeneous inter word spaces.

The handling of patterns is different for the three LaTeX based typesetting engines. The pdfLaTeX and XeLaTeX typesetting engines require that the patterns are preloaded into their format files, while LuaLaTeX resorts to a different mechanism so that the format file knows only the list of all the available pattern files, but, except for American English, none is preloaded, and the typesetting engine loads at run time the patterns it needs for the only languages that are specified for the document.

In this paper we show how to create a *language description file* for a generic language, and how to

create a *pattern file* for that language. Moreover we shortly describe the service files through which the typesetting engines are specified what to preload, where to find it, what to replace, and so on.

Language description files are pretty easy to create; on the opposite pattern files are pretty difficult, and different languages require different approaches. In the sequel I will cite some of my own work without the purpose of showing an achievement of mine, but to make examples of what and how to do it in order to have the necessary structure for language handling.

I have created pattern files for more than a dozen languages, mainly romance ones, and I followed the same method that, according to my experience and my capabilities, was suitable for all of them. For example I created the patterns for Latin in these varieties: modern (as it is being used in Italy), medieval, classical, ecclesiastical, and liturgical. At the moment these patterns are supported by another team composed of several people with different linguistic backgrounds and well supported by Latinist scholars; they are doing a marvellous work; they started from my patterns, and they are modifying and extending them using another very different approach that will lead them to a much better final result.

2 The required files

We have seen that we need three file levels for handling any language.

1. The language description language files; there are two kinds of such files, because there are two different packages that are used to handle a given language.
   (a) The ⟨*language*⟩`.ldf` to be used with babel is generally matched with a documentation file named ⟨*language*⟩`.pdf` resident in a `doc/generic/babel-`⟨*language*⟩ folder. The latter file describes the features and the functionalities associated to the specific ⟨*language*⟩, such as user settings, commands and/or shortcuts to perform certain actions when a specific text is typeset in that ⟨*language*⟩. The user can access this documentation by means of the terminal command `texdoc babel-`⟨*language*⟩ command.
   (b) The file `gloss-`⟨*language*⟩`.ldf` to be used with polyglossia; it is not matched with a documentation file; actually a short text concerning this ⟨*language*⟩ is included into the polyglossia documentation (Charette e Reutenauer, 2018); the user can access this documentation with the terminal command `texdoc polyglossia`.
2. The pattern files have different extensions: `.tex` for use with pdfLaTeX and XeLaTeX; `.txt` for use with LuaLaTeX. Their names are of the form `hyph-`⟨*language ISO code*⟩⟨*-modifiers*⟩. The ⟨*language ISO code*⟩ is a two or three letter standard name established by the ISO regulations, such as `en` for English, `it` for Italian, `la` for Latin, `kmr` for Kurmanji (Northern Kurdish), and so on. The ⟨*modifiers*⟩ are short strings that specify a specific variety of the ⟨*language*⟩: for example US English file has a full name of `hyph-en-us` while British English one has a full name of `hyph-en-br`; other languages may have longer modifier strings, but the idea is the same. The pattern file may be accompanied by other specific files; for example all languages that use the apostrophe as an elision marker (for example, French, Italian, Catalan,...) as well as a single or (ligated) double closed quotation marks have another matching file named `hyph-quote-`⟨*language ISO code*⟩ containing specific patterns to handle such writing elements.
3. The service macros are basically the ones that list the files described above; these service macros are used only when a format file has to be (re)created. These files have names such as `languages.`⟨*specific extension*⟩ where the ⟨*specific extension*⟩ is different depending on which typesetting engine the format file is created for. These files are maintained by the TeX-hyphen Working Group that takes care of every detail (including licences, since the pattern files are used also by other free or commercial external programs).

The user should not care about these "official" files; s/he needs these latter files only for a local installation that s/he uses for testing the other files s/he created; eventually s/he sends his/her language description and the pattern files to the TeX-hyphen Working Group that, in turn, takes care of their conformance with the TUG regulations in this matter and possibly adds them to the official TeX system distribution.

A partial description of such three kinds of files appears in both the TeX Live documentation (Berry, 2018a) and that of the `tlmgr` program, (Berry, 2018b) that handles the whole management of the TeX Live distribution. For MiKTeX there might be different ways of managing these language details, but in general the end user does not need to access such functionalities for his/her everyday typesetting chores.

3 Syllabification vs. hyphenation

The hyphenation patterns were invented by Frank Liang at the very beginning of the existence of TeX and were described in his doctoral thesis defended

a little later at Stanford University (Liang, 1983). At that time it was still TEX 78, a beta version of the future TEX. But he was essentially concerned with American English hyphenation because at that time this was the only language the TEX program could handle. The idea was a winning one; it was extended to other languages handled by the TEX system; nowadays it has been ported to a number of widespread free and commercial programs; just to name a couple, the Open/Libre Office word processors and the InDesign program.

But patterns deal with hyphenation, not with syllabification; the difference is essential; linguists deal with syllabification with certain criteria; phonologists deal with syllabification with other criteria. Linguistic syllabification deals with grammar rules; phonetic syllabification deals with the pronunciation of a specific language. Both approaches do no directly concern typography. Typography is mostly concerned in line breaking and with splitting words in a way that the reader is not confused while reading his/her text. Of course hyphenation should not infringe grammatical rules, but it is subject to stricter needs that force using as hyphenation break points only a subset of the grammatical ones.

This difference sometimes is confusing anybody who has to deal with line breaking. But it must be clear that the purpose of typography is not to take over the expertise of a linguist; typography main purpose is the reader's comfort in reading.

Actually even the grammar rules are not so strict; on one side every language contains words that are exceptions to the rules; on the other side there are certain texts that presume a different way of syllabification. Some of these texts are poems: the verses must follow a certain syllabic rhythm so that sometimes certain diphthongs are split so as to become hiathus. Other texts are the scripts that a speaker should read in a formal way; the speaker certainly emphasises certain words again by pronouncing some diphthongs as if they were hiathus. These particular situations are not taken into account by hyphenation.

Another point: hyphenation should not take place when the break point is too close to the start or the end of a word; the TEX system typesetting engines at the moment take into consideration such minimum distances by means of two numerical parameters `\lefthyphenmin` and `\righthyphenmin`. For most languages such parameters are preset to the values 2 and 3; these numbers measure the distance from the start and the end of a word in terms of "number of characters"; the TEX-hyphen Working Group is studying how to exclude from this count the self combining characters available with OpenType fonts; actually the `luatex` engine is already capable of doing this "correction".

The TEX-hyphen Working Group is also studying a way to assign different penalties to every possible hyphen point so as to reduce the number of short word fragments; short words just over the minimum length established by the `...hyphenmin` parameters may have just one hyphen point to use, but longer words have more hyphen points and their choice may be subjected to the optimisation algorithm (based on penalties) that the TEX system typesetting engines use to process paragraphs. Therefore in the future there will be more new functionalities associated to hyphenation.

4 The patterns

The patterns are short letter sequences that represent word fragments; these fragments may be of any length, from one letter up. The letters are interspersed with digits from 0 to 9. The value 0 is optional, because it is assumed when no digit is explicitly shown. Any position occupied by an even digit is where break points are forbidden, while on the opposite a position where an odd digit is present implies a possible break point; when in different patterns the same two letters are separated by different digits the highest value prevails in deciding if it is a legal break point or no break is allowed.

4.1 Word strings

The above introduction to patterns is a simplified one. It is better to specify in a better way how the typesetting engines deal with words to hyphenate.

The typesetting engine receives from previous steps of text processing each paragraph in the form of a single long string of tokens; the aim of the paragraph processing module is to divide such long line into lines of equal length, except possibly the last one. The string to process is made of words, inline math, penalties, kernings, punctuation marks, footnotes, macros that shall be expanded when the paragraph is shipped to the output file, and other such objects.

Penalties are used for both line breaking or page breaking; kerns are used to set adjacent characters in a proper way, but they might influence line breaking; the same is true for ligatures, such as the "f" ones, that might be interrupted by a hyphen in case a line break occurs at that point.

Other macros, such as `\footnote`, may influence hyphenation in the sense that they interfere with the process of finding a word end. But it is not up to the program to "correct" these situations; it is up to the user to provide some way of avoiding them; the simplest one is to put a zero width glob of glue just before the `\footnote` macro (or before any similar offending macro), so that this glue lets the program identify the word end.

In any case the program must identify each word for possible hyphenation; the program considers a "word" any string of consecutive characters that have a positive value of their `\lccode`. In general

any alphabetic letter has already been classified as being of category 11 while any non alphabetic character has been classified as being of category 12.

Characters of category 11 have two associated codes: the "lowercase" and the "uppercase" codes; respectively assigned with the native commands `\lccode`⟨*character*⟩=⟨*value*⟩ and `\uccode`⟨*character*⟩=⟨*value*⟩. Here ⟨*character*⟩ means the internal numerical code used by the engine to address a specific character in a given font; ⟨*value*⟩ is another numerical value to address a character in the same font. Such associations are mostly useful for lower/upper case transformations. Generally speaking users don't set and/or reset lower and upper case codes; they also generally do not know the glyph addresses in a given encoding; therefore a special grammar is used in order to access to such addresses. As an example let us see the situation with characters "a" and "A":

```
\lccode‘\a=‘\a \uccode‘\a=‘\A
\lccode‘\A=‘\a \uccode‘\A=‘\A
```

This means that when lowercasing "a" the same glyph is used, while when uppercasing it, "A" is used; the opposite takes place with "A".

But for what concerns hyphenation the program considers as valid symbols also non alphabetic signs provided they have a positive lower case code. Therefore for languages that use the elision apostrophe, such glyph must receive a positive `\lccode`; the language description file should therefore set this value but it has to reset it to zero, upon changing language. Assigning a category code to glyphs is generally done by the LaTeX kernel and/or by the encoding packages used for input and/or output and/or by the classes and packages that are being used to typeset the document; this action takes place also when a specific language is being typeset with an alphabet different from the Latin one.

All patterns are written with lowercase letters; this implies that in order to find the possible break points, the typesetting engine must deal only with the lowercase codes of the glyphs forming the words. Since other glyphs in a string do not have lowercase positive codes, the engine is capable of isolating single words formed by glyphs of any category, but with positive lowercase codes.

4.2 How patterns work

Let us explain how patterns work with a simple English word: "electricity". The typesetting engine module searches in the English pattern-list all the patterns that can be found into the given word, at its beginning, at its end, and anywhere in the middle. Let us show the various patterns found in the diagram of table 1; for simplicity the first line contains the letter string that the typesetting engine recognises as a word. Here we neglect some details, but the description is correct.

The `hyph-en-us.tex` file contains the original Knuthian 4938 hyphenation patterns for American English. In table 1 line 1 contains the original word; the dots before and after the word are just place holders to show the word boundaries; such dots are used also in the pattern files to mark the pattern strings that may be found at the beginning or at the end of a word. Line 2 contains the default digit 0 between every couple of letters; if no patterns were found in the pattern file, these zeros would imply that there are no legal break points in the word. The pattern file contains only the following patterns `2c1t`, `2c1it`, `2ici`, `4rici` that are made of substrings that can be found in the given word; they are copied in the table lines 3, 4, 5, and 6 with their letters in column with the letters of the original word and their digits in the position they are in the patterns. Eventually line 7 contain the letters in their columns and the highest digit in each numerical column. The only odd digits occur between the couples `ct` and `ci`, therefore the word may be hyphenated as such: *elec-tric-ity*. The grammar might allow other break points, but the preset `...hyphenmin` parameters are set to 2 and 3, therefore the first word fragment must be at least two characters long and the last one must be at least three characters long.

4.3 Grammatical rules

Not all languages have grammatical rules for syllabification and therefore for hyphenation; as we said, even if grammatical rules do exist, every language has several words that form a set of exceptions to "the rule".

In general a syllable contains one vowel, or one diphthong, or one triphthong. The set of vowels is specific of every language; we are mostly used to the Latin script, and we assume the vowels are in the set "aeiou"; some other languages have larger sets, such as "aäåâeèéêioôu", or "aàáâeèéêioôu", and so on. Some languages contain among their vowels also some characters other languages consider consonants: for example in some Slavic grammars the letter "r" is labelled as a vowel; matter of fact how could anybody pronounce "Trst" (the city of Trieste) or "smrt" (death) if the letter "r" did not contain some "voice" in itself?

Diphthongs and triphthongs are made of two or three vowels, respectively; for diphthongs the couple must contain an unstressed semivowel "i", or "u", or "j", or "y", or "w", the last three of which are considered semivowels only in certain languages. Triphthongs must contain two unstressed semivowels either close to one another or sandwiching the third vowel.

One consonant or a cluster of consonants may be part of a syllable when they precede the vocalic part; when these clusters start with specific consonants the latter might remain appended to the vocalic part of the preceding syllable; they gener-

Table 1: The process of analysing the patterns that are found in the word "electricity" and of finding the legal break points

.	e		l		e		c		t		r		i		c		i		t		y	.
.	e	0	l	0	e	0	c	0	t	0	r	0	i	0	c	0	i	0	t	0	y	.
						2	c	1	t													
												2	i	0	c	0	i					
														2	c	1	i	0	t			
										4	r	0	i	0	c	0	i					
.	e	0	l	0	e	2	c	1	t	4	r	2	i	2	c	1	i	0	t	0	y	.

ally are "l, r", and "m, n", but such consonants that remain appended to the previous syllable are specific of every language; for example, in French the consonant "s" remains appended to the preceding syllable, while in Italian it remains at the start of the current syllable. The consonants at the end of a word remain appended the previous vocalic part, as well as all the initial word consonants remain with the vocalic part they precede, irrespective if they start with the special consonants mentioned above. In some languages reinforced (double) consonants are split between syllables.

Compound words may violate the above rules, in the sense that the grammar of the language includes a further rule that a legal break point is located also at the junction of two component words. Some languages, on the opposite, treat compound words as if they were a single word. Italian is one of such languages but the Italian UNI regulation dealing with hyphenation, (UNI 6461, 1969), does not forbid hyphenation at a compound word boundary; this is why a word such as "transatlantico" (compound with "trans" and "atlantico") may be divided either way: *tran-sat-lan-ti-co* and *trans-at-lan-ti-co*. While creating patterns one of these choices must be coherently selected in order to treat all compound words in the same way.

In some languages the break point at a compound word boundary might require also a change of spelling: for example the German word "Bettuch"[1] (bed sheet) gets hyphenated in the form "Bett-tuch". These situations are not dealt with by patterns and the language description files must provide shorthands to handle them.

4.4 Syllabification dictionaries

Some languages are so complicated in their spelling and/or syllabification rules that it is almost impossible to define any reasonably sized set of rules; it is therefore impossible to create a reliable set of patterns so that they provide the greatest part of the work, without recurse to large exception files that list the words that do not follow the rules.

One such language is English, at least the American variety; we are aware that British English has been subjected to a reform in syllabification but we haven't seen yet any pattern file that implements the new standard.

In any case such languages must be treated with the `patgen` program (Liang e Breitenlohner, 1991), that is part of the TEX Live and MiKTEX distributions, together with some of its siblings; the original `patgen` was created at the very beginning of the TEX system on purpose, because the American variety was so complicated that it was impossible to derive from the grammar rules the hyphenation patterns.

The `patgen` solution consists in processing a large list of already hyphenated words in order to extract from them a pattern set that *minimises the probability* of producing wrong break points; may be such set might prove to be insufficient to hyphenate correctly almost any word in the list, and therefore in the language, but it is better than nothing. The original Knuthian patterns generated with `patgen` had an error rate of about 10%; the list of exceptions contains some thousands of words, so that the error rate diminishes significantly but it is not zero. In the past years the list of exceptions was added to the format file, so that nowadays it is difficult but not impossible to get wrong break points.

One detail that makes it impossible to correctly hyphenate English words, is that certain homographs get hyphenated in a different way according to their pronunciation; take for example "he analyses" and "the analyses", where the first "analyses" string gets hyphenated as *a-na-lys-es* while the second one becomes "a-nal-y-ses". Actually these break points are those that might be obtained if both `...hyphenmins` were set to 1; actually they are set by default to 2 and 3 respectively, so that the actual break points are just *ana-lyses* and *anal-y-ses*. The typesetting engines can "read" the text but they cannot "pronounce" the words; they work on spelling, not on sound.

It should be obvious that the quality of the patterns depends very much on the number of words contained in the input list to `patgen`; Frank Liang started with a list of 25 000 words; eventually the actual American English patterns are built with a list of 100 000 words, but the problem shown in the previous paragraph cannot be solved with patterns or exception lists; the problem could be

1. This is the "old spelling" prior to 1996; with the "new spelling" it is Betttuch.

solved if the typesetting engines contained also a *semantic* analysis module for each language, but this would be too heavy on the typesetting process.

Provided one has available a hyphenated word list, or is keen to manually hyphenate several myriad words (remember: one myriad contains 10 000 items) it is difficult to create language patterns for all languages, even those that have clear grammar rules. We are not discouraging the use of `patgen`; we simple underline the difficulties implied with its use.

4.5 Creating patterns by hand

When clear grammar rules are available, hyphenation patterns may be manually created. I followed this approach for more than a dozen languages; I compared my patterns with those that I could derive by using `patgen` and generally I found that I was able with fewer patterns to get a better success percentage (close to 100%) compared to that obtainable with `patgen`. I would not generalise this conclusion, but this is what I found out in my experiments. I agree that with languages such as English the `patgen` approach would be the only affordable.

I created pattern files for a dozen languages that had all clear and simple rules for syllabification; I generally created the patterns so that if the `...hyphenmin` parameters are both set to 1, the hyphenated words are always correct; well, some people might argue that this is an overstatement; the point is that I do not consider wrong a word division that does not contain *all* possible break points. The reasoning is that the reader is uncomfortable if the break point takes place just after or just before a vowel that might behave as a semivowel. In Italian for example the word "paura" (fear) is grammatically syllabified as *pa-u-ra* but I managed that patterns divide it as *pau-ra*; in this word the group "au" is a hiatus, because the vowel "u" is tonic and therefore the vowels don't form a descending diphthong so that they may be grammatically divided; but by doing so, the reader is uncomfortable. When accents are used such situations should not take place, but even so the reader is uncomfortable.

How to create patterns: the simplest way is to follow a procedure used also by the TEX-hyphen Team; name with capital letters sets of letters that have the same behaviour for what concerns hyphenation; for example O denoting the open or semi open vowels, C to denote the closed vowels that may form diphthongs and triphthongs; K to denote "normal" consonants; L to denote those consonants that may be appended to the preceding syllable; and so on.

Write down the grammatical rules in terms of these classes, i.e. create "macro patterns" in terms of these classes; then use some external software to expand such macro patterns to create the patterns with actual vowels and consonants. You get a large number of patterns many of them might be superfluous, because some combinations do not exist in a specific language; nevertheless, if a searchable dictionary file is available for that language, try to find specific words that contain unusual letter combinations. Most of the time such words can be found and one realises that most of those unusual patterns were not useless. In any case they might not be part of the normal words of a language because they are derivatives forms from a foreign root; therefore some of those unusual patterns are not to be thrown out.

4.5.1 The Italian example

These foreign roots may give some problems to any language; grammars of Italian state that any consonant cluster belongs to the syllable that contains the following vocalic group if and only if there is another Italian word that starts with that consonant cluster. This works fine most of the time, but it is an imprecise rule: it should be completed with the clause of "... Italian word, *not of Greek etymology*, that starts with...". The difficulty for the user is to know the etymology of every word; this is why the UNI regulation explicitly specifies that the groups `bd, gm, ps, pn, tm,…`, must be divided between two adjacent syllables. This of course excludes such groups at the very beginning of specific words (of Greek etymology) but interferes with compound words containing such words; "psicologia" is definitely of Greek etymology, but what do you do with the Italian word "parapsicologia"? It is necessary to create a special pattern for words containing the string `psic`; what to do with "pneuma" and "apnea"? It is necessary to do the same[2].

This is why the intelligence of the human being comes into play to solve these special situations that require manual intervention. The same intervention is necessary to handle Italian words that derive from foreign names such as "newyorkese", "maxwelliano", "wagneriano", "massmediologo", "leishmaniosi", "lewisite", "wahhabbita", and the list might go on for a long list of other such words.

Just to remain with Italian as an example, the few patterns needed to correctly hyphenate all Italian words without foreign roots (but also many such mixed etymology words) are shown in code listing 2.

It is worth noting what follows.

1. The dot that precedes or follows a given pattern marks that it may be used only at the beginning or, respectively, at the end of a word.

2. Actually in "apnea" the first "a" is a prefix, therefore some dictionaries prefer to use the compound word division.

Code 2: The 355 patterns needed to hyphenate the Italian language

```
\patterns{% Pattern start
.a3p2n
.anti1 .anti3m2n
.bio1
.ca4p3s .circu2m1 contro1
.di2s3cine
.e2x1eu
.fran2k3 .free3
.li3p2sa
.narco1
.opto1 .orto3p2 .para1
.ph2l .ph2r
.poli3p2 .pre1 .p2s
.re1i2scr
.sha2re3
.tran2s3c .tran2s3d .tran2s3l .tran2s3n .tran2s3p .tran2s3r .tran2s3t
.su2b3lu .su2b3r
.wa2g3n .wel2t1
2'2
a1ia a1ie a1io a1iu a1uo a1ya
2at.
e1iu e2w
o1ia o1ie o1io o1iu
1b 2bb 2bc 2bd
2bf 2bm 2bn 2bp 2bs 2bt 2bv b2l b2r 2b. 2b'
1c 2cb 2cc 2cd 2cf 2ck 2cm 2cn 2cq 2cs 2ct 2cz
  2chh c2h 2ch. 2ch'. 2ch''. 2chb ch2r 2chn
  c2l c2r 2c. 2c' .c2
1d 2db 2dd 2dg 2dl 2dm 2dn 2dp d2r 2ds 2dt 2dv 2dw 2d. 2d' .d2
1f 2fb 2fg 2ff 2fn f2l f2r 2fs 2ft 2f. 2f'
1g 2gb 2gd 2gf 2gg g2h g2l 2gm g2n 2gp g2r 2gs 2gt 2gv 2gw
  2gz 2gh2t 2g. 2g'
.h2 1h 2hb 2hd 2hh hi3p2n h2l 2hm 2hn 2hr 2hv 2h. 2h'
.j2 1j 2j. 2j'
.k2 1k 2kg 2kf k2h 2kk k2l 2km k2r 2ks 2kt 2k. 2k'
1l 2lb 2lc 2ld 2l3f2 2lg l2h l2j 2lk 2ll 2lm 2ln 2lp 2lq
  2lr 2ls 2lt 2lv 2lw 2lz 2l. 2l'. 2l''
1m 2mb 2mc 2mf 2ml 2mm 2mn 2mp 2mq 2mr 2ms 2mt
  2mv 2mw 2m.2m'
1n 2nb 2nc 2nd 2nf 2ng 2nk 2nl 2nm 2nn 2np 2nq 2nr 2ns
  n2s3fer 2nt 2nv 2nz n2g3n 2nheit 2n. 2n'
1p 2pd p2h p2l 2pn 3p2ne 2pp p2r 2ps 3p2sic 2pt 2pz 2p. 2p'
1q 2qq 2q. 2q'
1r 2rb 2rc 2rd 2rf r2h 2rg 2rk 2rl 2rm 2rn 2rp 2rq 2rr 2rs
  2rt r2t2s3 2rv 2rx 2rw 2rz 2r. 2r'
1s2 2shm 2sh. 2sh' 2s3s s4s3m 2s3p2n 2stb 2stc 2std 2stf
  2stg 2stm 2stn 2stp 2sts 2stt 2stv 2sz 4s. 4s'. 4s''
.t2 1t 2tb 2tc 2td 2tf 2tg t2h 2th. t2l 2tm 2tn 2tp
  t2r t2s 3t2sch 2tt t2t3s 2tv 2tw t2z 2tzk tz2s 2t. 2t'. 2t''
1v 2vc v2l v2r 2vv 2v. 2v'. 2v''
1w w2h wa2r 2w1y 2w. 2w'
1x 2xb 2xc 2xf 2xh 2xm 2xp 2xt 2xw 2x. 2x'
y1ou y1i 1z
2zb 2zd 2zl 2zn 2zp 2zt 2zs 2zv 2zz 2z. 2z'. 2z'' .z2
}% Pattern end
```

2. The patterns that involve the letters "j, k, w, x, y" involve only words with foreign roots, because in strict Italian they never appear[3].
3. The numerous patterns involving the "h" line deal mostly with foreign root words, because in Italian the "h" glyph is used more as a diacritical mark rather than a pronunceable sound.
4. The fist part of the pattern set concerns only etymological hyphenation of a few prefixes; the other prefixed words may be treated as regular words.
5. The reader can see that, except for a few triphthongs, most patterns do not contain vowels; in this way patterns containing accented vowels are unnecessary.
6. The reader also may notice that the above described scheme of combining the couples of consonants emerges from the pattern set; the patterns containing couples that never appear have been deleted.
7. At the same time the reader may see the difference of the liquid and nasal consonants "l, m, n, r" that may remain appended to the preceding syllable, opposite to the behaviour of the other consonants.
8. Also the "s" unique property at the beginning of a consonant cluster, shows that it remains attached to the cluster, instead of being separated as it happens in all other romance languages.
9. All consonants may be reinforced by doubling; in this case the first occurrence remains attached to the preceding syllable, at least in words with Latin etymology.

4.5.2 The Latin case

Latin is an emblematic situation where there are several variants: modern, medieval, classical, ecclesiastic, liturgical, with prosodic marks, and so on. They differ in the alphabets they use and in the hyphenation rules; infix words may also be different or have a different spelling according to the specific alphabet.

Let us see the main differences.

Modern Latin uses the Latin alphabet; no surprise; but here with Latin alphabet the whole 26 letters alphabet is meant; without the Latin ligatures æ and œ. No accents are used; modern Latin hyphenation rules in Italy are very similar to the Italian ones, but in Germany they use slightly different rules. It is mainly used for schoolbooks, grammars and literature collections, in high schools and partially in universities. But most important, it is the official language of the Holy See. It is not the official language of the Vatican City State, because its official language is Italian. The Pope is the monarch of both entities, but the formal writings signed by the Pope as the Head of the Roman Catholic Christianity are written with this Latin variety.

Modern Latin with prosodic marks uses the macron and the breve diacritical marks and is used in school grammars and dictionaries; the length of each vowel is marked as "long" with the macron, or "short" with the breve; it helps students when reading the hexametres and pentametres of the Latin poems.

Medieval Latin uses a shorter alphabet:

```
æbcdefghiklmnoœpqrstuyz
ÆBCDEFGHIKLMNOŒPQRSTVYZ
```

where the Latin ligatures æ and œ more often than not are contracted to their pronunciation "e"; the use of "I" and "Y" is generally inconsistent; I have seen the spellings Italia and Ytalia; Iesus and Yesus even in the same text. The only sign "u" was used also for the consonant "v", and viceversa for the capital variant. Actually the differentiation between the two signs, not only in Latin, took place by the end of the XII century. It does not require much, except changing Novembris to Nouembris, and adding few patterns to the same pattern file of modern Latin; such peculiar patterns involve "u" just where it plays the role of a "v".

Classical Latin uses "u" and "V" the same way as medieval Latin, but it does not use the Latin ligatures "æ" and "œ". Hyphenation is completely different from the modern and medieval varieties; prefixes and suffixes are provided breakpoints in terms of etymology, instead of grammatical rules only; "i" and "u" never play the role of semivowels, therefore there are no diphthongs; even "ae" and "oe" are pronounced separately by classical latinists, but are not divided. The guttural consonants "c" and "g" remain guttural and never become palatal sounds; they never form digraphs or trigraphs as in other variants of Latin. Moreover all words with a Greek etymology are hyphenated with the Greek rules. As one can imagine, classical Latin requires a completely different management, especially with hyphenation, because the inflexional nature of the language requires not only a strict relationship with etymology but also the analysis of all the inflected forms of verbs, nouns, and adjectives. The French team that is working on this problem is doing a beautiful work thanks, also, to the support of latinist scholars.

The fact that classical Latin does not use the medieval æ and œ ligatures renders the prob-

3. Actually the letters "k" and "x" may be part of words of Greek etymology; "j" is an old fashioned spelling for the semivowel "i"; the five of them appear in neologisms of foreign origin and in proper names.

lem of homographs quite more complicated; for example the inflected form *aeris* comes from the nouns *aer* (air) and *aes* (bronze); the first requires to be hyphenated as *a-e-ris*, the second as *ae-ris*; if the second were written with the diphthong ligature *æs, æris* there would not be homographs any more. In ecclesiastic Latin, on the opposite, the first one would be spelled *aer,..., áeris* and, again there would not be homography any more.

Ecclesiastic Latin is used in clergy texts, such as missals and breviaries. It is mainly modern Latin with (acute) accents; such accents are used to mark the "rhythm" in word pronunciation. It is done in such a way that clergy of every national mother language background and used to a different rhythmic approach may pray together pronouncing the same words with the same stress. The accents are missing from all words where the stress falls on the penultimate syllable; nevertheless in view of uniform pronunciation, the clergy with different mother language backgrounds may have difficulties to perceive the difference between diphthongs and hiathus, so that if the penultimate syllable contains a group of vowels the tonic one is marked with the accent.

There are very little differences in the infix words, but accents introduce many complications in hyphenation, therefore it is necessary to provide for such differences.

Liturgical Latin is someway between ecclesiastic and classical Latin; it is pronounced the same as ecclesiastic, but it is hyphenated as if it were classical Latin, with minor variations due to the fact that the pronunciation is different from the classical one. It is mainly used in liturgical singing; nowadays, that the national languages are used in all Roman Catholic services, Latin remains in use only when singing the Gregorian plain chant. Matter of fact it is the GregorioTeX Team that is taking care of the hyphenation patterns for liturgical Latin. Meanwhile they revise also the other varieties of Latin.

German style modern Latin is a variant of modern Latin. The spelling is the same, but hyphenation is somewhere between Italian style modern latin, and classical Latin. Evidently in Germany Latin is taught with a different approach than in Italy. Possibly the revision that the GregorioTeX Team is doing will produce a different set of hyphenation patterns even for modern Latin, so that it is not excluded that in the future the actual Italian style hyphenation is replaced by the German style one.

I can report that at the moment the `hyph-la.tex` file contains the patterns for (Italian style) modern and medieval Latin; prosodic marks are dealt with special macros defined in the language description file `latin.ldf` for `babel`; they do not need any particular functionality when using OpenType fonts with LuaTEX and XeLaTEX. The number of patterns in the above pattern file amounts to 334 items. They are valid for both the Italian style modern Latin and the medieval one.

Just as the patterns for Italian were shown before, those for modern and medieval Latin are shown in pattern list 3.

At the moment of writing the patterns for classical Latin are in file `hyph-la-x-classic.tex` and amount to 658 items.

Similarly the patterns for liturgical Latin are stored in file `hyph-la-x-liturgic.tex` and amount to 1730 items. It is not excluded that after the revision by the GregorioTeX Team both such patterns files may double in size.

4.5.3 The Greek case

In Greek we have a similar situation as in Latin. There are some main differences: modern Greek has two spellings: monotonic and polytonic, but in spite of their different accent system they have the same lexicon; modern and ancient Greek are written in a different alphabet than Latin; modern and ancient Greek use a lot of diacritics; the lexicon of ancient Greek is rather different from the modern one. In facts modern Greek lexicon contains groups of consonants that never occur in ancient Greek. Some are special combinations to render sounds absent from Greek but present in loan words.

The language has inflection for verbs, nouns and adjectives and the diacritical marks quite often change and/or do not remain on the same vowel in the inflected forms.

Hyphenation therefore is further complicated by the presence of diacritical marks.

But the modern versions of the language follow simple grammar rules and do not take in consideration prefixes and suffixes. At the time of writing this document the `...hyphenmin` values are both preset to 1, but many typographers in Greece set them to 2 and 3.

Modern monotonic Greek is the most used version of Greek in Greece. The alphabet is the usual 25 lowercase and 24 uppercase letters:

αβγδεζηθικλμνξοπρσςτυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣ ΤΥΦΧΨΩ

It uses just two accents, the "tonos" (in practice the acute accent), and the "dialytika" (the diaeresis). The tonos is used on the tonic syllable of polysyllabic words, and the dialytica when it is necessary to split an apparent diphthong; sometimes the dialytica and the tonos fall on the same vowel; all vowels including iota and upsilon may get the tonos, but the

Code 3: The 334 patterns for modern and medieval Latin

```
\patterns{% Pattern start
.a2b3l .anti1  .anti3m2n .circu2m1 .co2n1iun
.di2s3cine .e2x1 .o2b3
.para1i .para1u .su2b3lu .su2b3r 2s3que.  2s3dem. 3p2sic 3p2neu
æ1 œ1 a1ia a1ie a1io a1iu ae1a ae1o ae1u e1iu io1i o1ia o1ie o1io o1iu uo3u
1b   2bb   2bc   2bd   b2l  2bm  2bn  b2r  2bt  2bs  2b.
1c   2cc   c2h2  c2l   2cm  2cn  2cq  c2r  2cs  2ct  2cz  2c.
1d   2dd   2dg   2dm   d2r  2ds  2dv  2d.
1f   2ff   f2l   2fn   f2r  2ft  2f.
1g   2gg   2gd   2gf   g2l  2gm  g2n  g2r  2gs  2gv  2g.
1h   2hp   2ht   2h.
1j
1k   2kk   k2h2
1l   2lb   2lc   2ld   2lf  l3f2t 2lg 2lk  2ll  2lm  2ln  2lp  2lq  2lr
     2ls   2lt   2lv   2l.
1m   2mm   2mb   2mp   2ml  2mn  2mq  2mr  2mv  2m.
1n   2nb   2nc   2nd   2nf  2ng  2nl  2nm  2nn  2np  2nq  2nr  2ns
     n2s3m n2s3f 2nt   2nv  2nx  2n.
1p   p2h   p2l   2pn   2pp  p2r  2ps  2pt  2pz  2php 2pht 2p.
1qu2
1r   2rb   2rc   2rd   2rf  2rg  r2h  2rl  2rm  2rn  2rp  2rq  2rr  2rs  2rt
     2rv   2rz   2r.
1s2  2s3ph 2s3s  2stb  2stc 2std 2stf 2stg 2st3l     2stm 2stn 2stp 2stq
     2sts  2stt  2stv  2s.  2st.
1t   2tb   2tc   2td   2tf  2tg  t2h  t2l  t2r  2tm  2tn  2tp  2tq  2tt
     2tv   2t.
1v   v2l   v2r   2vv
1x   2xt   2xx   2x.
1z   2z.
% Additional patterns for medieval Latin
a1ua a1ue a1ui a1uo a1uu e1ua e1ue e1ui e1uo e1uu
i1ua i1ue i1ui i1uo i1uu o1ua o1ue o1ui o1uo o1uu
u1ua u1ue u1ui u1uo u1uu
%
a2l1ua a2l1ue a2l1ui a2l1uo a2l1uu e2l1ua e2l1ue e2l1ui e2l1uo e2l1uu
i2l1ua i2l1ue i2l1ui i2l1uo i2l1uu o2l1ua o2l1ue o2l1ui o2l1uo o2l1uu
u2l1ua u2l1ue u2l1ui u2l1uo u2l1uu
%
a2m1ua a2m1ue a2m1ui a2m1uo a2m1uu e2m1ua e2m1ue e2m1ui e2m1uo e2m1uu
i2m1ua i2m1ue i2m1ui i2m1uo i2m1uu o2m1ua o2m1ue o2m1ui o2m1uo o2m1uu
u2m1ua u2m1ue u2m1ui u2m1uo u2m1uu
%
a2n1ua a2n1ue a2n1ui a2n1uo a2n1uu e2n1ua e2n1ue e2n1ui e2n1uo e2n1uu
i2n1ua i2n1ue i2n1ui i2n1uo i2n1uu o2n1ua o2n1ue o2n1ui o2n1uo o2n1uu
u2n1ua u2n1ue u2n1ui u2n1uo u2n1uu
%
a2r1ua a2r1ue a2r1ui a2r1uo a2r1uu e2r1ua e2r1ue e2r1ui e2r1uo e2r1uu
i2r1ua i2r1ue i2r1ui i2r1uo i2r1uu o2r1ua o2r1ue o2r1ui o2r1uo o2r1uu
u2r1ua u2r1ue u2r1ui u2r1uo u2r1uu
}% Pattern end
```

last two ones can get the dialytica or both diacritical marks.

Modern polytonic Greek uses the same lexicon of monotonic Greek but uses the full set of accents as ancient Greek, that is acute, grave and circumflex, plus three diacritical marks: the smooth breath, the rough breath and the dialytica; they may be combined in couples, therefore there are 15 different combinations (not available on all vowels, but some vowels may have that full variety of diacritical marks). It is obvious that hyphenation becomes very complicated, but at least hyphenation follows simple grammar rules.

Ancient Greek of course uses the ancient lexicon; it uses the same diacritical marks as modern polytonic Greek, plus the iota subscripted or adscripted. Hyphenation is further complicated since break points must take into consideration prefixes and suffixes.

The use of a different alphabet sets forth some problems with fonts. As it is well known pdfLaTeX handles only fonts the glyphs of which are coded with just one byte. On the opposite, LuaLaTeX and XeLaTeX use OpenType fonts that may contain glyphs for dozens of languages and require the Unicode encoding that is a multibyte one. Two different approaches are therefore required for such different typesetting engines.

For pdfLaTeX it is necessary to use specially coded Greek fonts where each glyph has a one byte address, while with OpenType fonts LuaLaTeX and XeLaTeX can fetch the glyphs they need directly from the many alphabets coded in those fonts with their multibyte encoding; but, caution: not all fonts contain the Greek script (in particular the OpenType Latin Modern default fonts), therefore it is important to check that the desired font contains it.

With pdfLaTeX a *Local GReek* (LGR) encoding is defined so that with a regular Latin keyboard it is possible to write Greek using a Latin transliteration; with OpenType fonts it is possible to enter the Greek text directly in Greek without any transliteration. Actually the `greek.ldf` language description file (for `babel`) uses a LICR (LaTeX Internal Character Representation) technique such that also with pdfLaTeX it is possible to enter Greek text in Greek. The problem, therefore, is not the input text, but the keyboard itself. Often modern computers have available a virtual keyboard to be acted upon with the mouse or by tapping a touchscreen, so that a real keyboard is now relatively superfluous.

In any case the LGR encoding maps the Latin characters to the Greek ones as shown in table 4. The various diacritical marks are prefixed to the letters, except the iota sub- or ad-scripted, that follows the letter; they are represented by the ASCII chracters ` ' " > < ~ | (respectively for grave, acute, circumflex, soft breath, rough breath, diaeresis, iota sub- or ad-scripted); they may also be paired; they do not need macros, but if macros are used a better kerned text is obtained and hyphenations should work better (but see below); therefore one can enter a transliterated text as `a>ut~h|` or `a\>ut\~h|` where the second input form is recommended; but, if a Greek real or virtual keyboard is available, it is possible to enter directly αὐτῆ. The internal LICR character representation fetches directly the marked letters and uses the correct kernings.

Table 4: Mapping of the Latin alphabet to the Greek one

abcdefghijklmnopqrstuvwxyz
αβςδεφγηιθκλμνοπχρστυ ωξψζ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
AB ΔΕΦΓΗΙΘΚΛΜΝΟΠΧΡΣΤΥ ΩΞΨΖ

But all this influences the hyphenation patterns; at the moment of writing the patterns to be included into the format file for `pdftex` work only with the Latin transliteration. With XeLaTeX there are no problems when the user specifies the Greek language and optionally selects one of its variants; with LuaLaTeX at the moment only the hyphenation for modern monotonic Greek is available; further on it will be shown a patch in order to extend the functionality of LuaLaTeX in connection with the varieties of Greek (of course, it is not necessary to patch anything if meanwhile the patch we suggested to the maintainers has already been applied).

With pdfLaTeX and its Latin transliteration, the Latin "v" key is not useless: it types an invisible strut character just as high as lowercase letters in order to have something to put diacritics on; it is more a service trick; it is used also to hide the actual word termination in order to type an isolated σ; in facts the ligature mechanism embedded into the LGR encoded Greek fonts is so strong that typing "s" at the end of a word gets transformed into a final sigma ς, so that non Greek users do not have to remember to insert "c" at the end of the words when using the Latin transliteration.

All these peculiarities imply a complicated pattern file; at the moment of writing this paper the existing pattern files to be used with pdfLaTeX work correctly only with Latin transliteration, not when the LICR codes or the direct Greek script input are used. To be precise, they do work also in these situations but, when accented glyphs are encountered, some break points may be missing.

The pattern files presently available when using pdfLaTeX were prepared by Dimitri Filippou several years ago, when only the Latin transliteration was available. I tried to upgrade those patterns, but without a specific testing made by actual Greek writers, I can't upload such new pattern files. In

the pattern list 5 I show these patterns for the monotonic variety; those for the polytonic and ancient varieties are too long to be shown here. In any case neither the patterns of table 5 (shown just to underline the special codes that are needed to identify the single byte 8-bit codes, from 128 to 255 in the LGR encoding) nor those for the polytonic variety is yet publicly available. Dimitri Filippou already provided the patterns for the three varieties to be used with LuaLaTeX and XeLaTeX, but nothing has been done to update the situation with pdfLaTeX; it is understandable: with the availability of XeLaTeX and LuaLaTeX, the importance of pdfLaTeX is strongly diminished for the Greek users.

The modification consists in replicating the patterns that involve ligatures between diacritics and vowels by replacing the accent-vowel pair with the upper half address of the LGR encoding; to do this it is necessary to code such letters with their double caret notation hexadecimal address; this means that the two (lowercase) characters that follow a couple of carets `^^`, form the hexadecimal address of the accented character; this is clearly evident in the seven lines that follow the comment `vowels`. The vowels iota and upsilon can receive the tonos alone, the dialytika alone, and both the dialytika and tonos.

Most of the patterns that are included in the pattern list 5 come from the pattern file `grmhyph5.tex` created by Dimitri Filippou; the upper half addresses of the LGR encoded Greek fonts have been added, and a very small number of consonant patterns has been added in order to handle a certain number of particular words, especially loan words from foreign languages and/or toponyms.

For modern polytonic and ancient Greek patterns the one byte situation is similar but it is much more complicated because of the multitude of diacritical marks and in both variants the inflection may move and/or change the accents, so that the number of patterns may be to six times larger than for monotonic modern Greek.

When using XeLaTeX and LuaLaTeX Dimitri Filippou created the `hyph-grc.tex` for ancient Greek that contains about 2000 patterns in Greek script. He also coded Greek patterns in Greek script for monotonic and polytonic Greek; their names are `hyph-el-monoton.tex` and `hyph-el-polyton.tex`.

4.5.4 General considerations

From the above it may seam that preparing pattern files requires an excellent ability in using "strange" codes; but it is not that difficult, since anybody with a good deal of patience and attentions can do the coding. The difficult part is to have a good knowledge of the language in order to extend the initial codes set forth by some scripting language, say Python or Java, to perform all the combinations required by the "macro patterns" discussed above. The pattern creator must be aware of the multitude of exceptions to the general rule without needing to create also the list of exceptions; for American English the *TUGboat* editor, Barbara Beeton, has been in charge of collecting and maintaining the list of exceptions that cannot be handled by the various pattern sets available for US English. But, as it has already been emphasised, American English has so many and "strange" rules that it is virtually impossible to manually create any set of patterns, so that recourse to `patgen` is the only practicable way to do the job. In any case with or without `patgen` homographs remain a big problem.

It is worth noting that there are some languages, especially typeset with other scripts, that do not use hyphenation; oriental languages that use ideographs do not hyphenate anything; their line breaking is on word boundaries or "anywhere". Arabic scripts do not use hyphenation but produce justified texts by horizontally stretching the "kashida", a curved line that decorates the ending forms of most letters.

5 The language description file

Hyphenation by itself is a necessary part while managing typesetting in different languages. Nevertheless some typesetters prefer typesetting ragged right; others produce block justification avoiding hyphenation of any kind; in order to avoid white meandering lines through the text, due to wide inter *word* spaces, they use tracking, that consists in enlarging the space between *characters*. For most book designers slight tracking is something to be used only when typesetting in all caps and/or small caps. With regular lowercase letters, with or without serifs, they claim that those who use tracking should be treated as those who steal sheep; this very strong sentence is understandable if one recalls that in the past centuries sheep thieves used to be condemned to death by hanging.

Instead of tracking, the best way to reduce to a minimum the number of hyphenated line breaks consists in using the microtype package. The TeX system typesetting engines follow an algorithm that minimises the badness of each paragraph; this is not the right place to discuss the details, but every LaTeX user knows how well paragraphs typeset by these programs are done. In any case package microtype does a very fine work in further improving this paragraph typesetting excellency.

The best results are obtained with pdfLaTeX and LuaLaTeX by applying two techniques: *protrusion* and *character expansion*. Protrusion lets small parts of the first and the last characters in a line to protrude in the adjacent margin. Character expansion consists in very slight stretching or shrinking of all the characters in a line so as to ren-

Code 5: The extended list of monotonic Greek hyphenation patterns for use with pdfLaTeX

```
\patterns{% Pattern start
%%%% vowels
a1 ^^881
e1 ^^e81
h1 ^^a01
i1 ^^d01 ^^f21 ^^f01
o1 ^^ec1
u1 ^^f61 ^^f41 ^^d41
w1 ^^b8
%%%%
a2i a2'i a2^^d0 a2u a2'u a2^^d4 'a3u ^^883u
e2i e2'i e2^^d0 e2u e2'u e2^^d4 'e3u ^^e83u
h2u h2'u h2^^d4 'h3u ^^a03u
o2i o2'i o2^^d0 o2u o2'u o2^^d4 'o3u ^^ec3u
u2i u2'i u2^^d0 'u3i ^^d43i
%%%%
a2h 'a3h. ^^883h. a2''i a2^^f0 'a3i. ^^883i. a2''u a2^^f4
e2''i e2^^f0 e2''u e2^^f4 o2ei o2h 'o3h. ^^ec3h. o2''i o2^^f0 'o3i. ^^ec3i.
%%%%
i2a i2'a i2^^88 i2e i2'e i2^^e8 i2o i2'o
o3''i3'o o3^^f03^^ec
i2w i2'w i2^^b8 .i3 'i3
h2a h2'a h2^^88 h2e h2'e h2^^e8 h2o h2'o h2^^ec h2w h2'w h2^^b8 .h3 .'h3 .^^a03
u2a u2'a u2^^88 u2o u2'o u2^^ec u2w u2'w u2^^b8 .u3 .'u3 .^^d43
%%%%
4b. 4g. 4gk. 4d.  4z.  4j.  4k. 4l. 4m. 4mp. 4n. 4nt. 4x. 4p. 4r.
4s. 4c. 4t.  4tz. 4ts. 4tc. 4f. 4q. 4y.
%%%%
4''  4b''  4g'' 4gk'' 4d'' 4z'' 4j'' 4k''  4l''  4m'' 4mp''
4n'' 4nt'' 4x'' 4p''  4r'' 4s'' 4t'' 4tz'' 4ts'' 4f'' 4q'' 4y''
%%%%
.b4 .g4 .d4 .z4 .j4 .k4 .l4 .m4 .n4 .x4 .p4 .r4 .s4 .t4 .f4 .q4 .y4
%%%%
4b1b 4g1g 4d1d 4z1z 4j1j 4k1k 4l1l 4m1m 4n1n 4p1p 4r1r
4s1s 4t1t 4f1f 4q1q 4y1y
%%%%
4b1z 4b1j 4b1k 4b1m 4b1n 4b1x 4b1p 4b1s 4b1t 4b1f 4b1q 4b1y
4g1b 4g1z 4g1j 4g1m 4r5g2m 4g1x 4g1p 4g1s 4g1t 4g1f 4g1q 4g1y
4d1b 4d1g 4d1z 4d1j 4d1k 4d1l 4d1x 4d1p 4d1s 4d1t 4d1f 4d1q 4d1y
4z1b 4z1g 4z1d 4z1j 4z1k 4z1l 4z1m tz2m 4z1n 4z1x 4z1p 4z1r 4z1s 4z1t 4z1f 4z1q 4z1y
4j1b 4j1g 4j1d 4j1z 4j1k 4j1m 4r5j2m sj2m 4j1x 4j1p 4j1s 4j1t 4j1f 4j1q 4j1y
4k1b 4k1g 4k1d 4k1z 4k1j 4k1m 4l5k2m  4r5k2m  4k1x 4k1p 4k1s 4k1f 4n5k2f  4k1q 4k1y
4l1b 4l1g 4l1d 4l1z 4l1j 4l1k 4l1m 4l1n 4l1x 4l1p 4l1r 4l1s 4l1t 4l1f 4l1q 4l1y
4m1b 4m1g 4m1d 4m1z 4m1j 4m1k 4m1l 4m1x 4m1r 4m1s 4m1t 4m1f 4m1q 4m1y
4n1b 4n1g 4n1d 4n1z 4n1j 4n1k 4n1l 4n1m 4n1x 4n1p 4n1r 4n1s 4n1f 4n1q 4n1y
4x1b 4x1g 4x1d 4x1z 4x1j 4x1k 4x1l 4x1m 4x1n 4x1p 4x1r 4x1s 4x1t 4g5x2t 4r5x2t 4x1f 4x1q 4x1y
4p1b 4p1g 4p1d 4p1z 4p1j 4p1k 4p1m 4p1x 4p1s 4p1f 4p1q 4p1y
4r1b 4r1g 4r1d 4r1z 4r1j 4r1k 4r1l 4r1m 4r1n  4r1p 4r1s 4r1t 4r1f 4r1q 4r1y
4s1d 4s1z 4s1n 4s1x 4s1r 4s1y
4t1b 4t1g 4t1d 4t1j 4t1k 4t1n 4t1x 4t1p 4t1f st2f 4t1q 4t1y
4f1b 4f1g 4f1d 4f1z 4f1k 4f1m 4f1n 4r5f2n 4f1x 4f1p 4f1s 4f1q 4f1y
4q1b 4q1g 4q1d 4q1z 4q1k 4q1m 4r5q2m 4q1x 4q1p 4q1s 4q1f 4q1y
4y1b 4y1g 4y1d 4y1z 4y1j 4y1k 4y1l 4y1m 4y1n 4y1x 4y1p 4y1r 4y1s 4y1t 4m5y2t 4y1f 4y1q
%%%%
4g5k2f 4g1kt 4m1pt 4n1tz 4n1ts
%%%%
4br. 4gl. 4kl. 4kt. 4gkc. 4gks. 4kc. 4ks. 4lc. 4ls.
4mpl. 4mpn. 4mpr. 4mc.  4ms. 4nc. 4ns. 4rc. 4rs.
4sk. 4st. 4tl. 4tr. 4ntc. 4nts. 4ft. 4qt.
%%%%
4gk1mp 4gk1nt 4gk1tz 4gk1ts 4mp1nt 4mp1tz
4mp1ts 4nt1mp 4ts1gk 4ts1mp 4ts1nt
}% Pattern end
```

der more homogeneous the inter word spaces. The results are visible in this very document, where the narrow columns still require some hyphenated line breaks but the inter word spaces do not let any meandering rivulets appear trough the paragraphs. XeLaTeX cannot use character expansion, but even by using only protrusion it does a fine work.

The choice of using microtype may be one of the tasks assigned to the language description files. The interested reader can examine the babel documentation Braams e Bezos (2018) that describes a model or a template for creating language description files; here we summarise the tasks that are generally assigned to such files.

Any language description file should do the following.

1. It makes sure the file is read just once.
2. It examines the presence of options, language modifiers and similar user choices in order to execute the proper settings.
3. For the language it describes, or for a language variant, it controls if suitable hyphenation pattern files have been loaded in the format file or can be loaded at run time (only LuaLaTeX). If no patterns are available a suitable set of alternate patterns are chosen; very often the default US English patterns are selected; sometimes those of another language; sometimes those of the "pseudolanguage" `nohyphenation` so that the almost empty, patternless file `zerohyph.tex` is loaded.
4. It checks if the `\captions`⟨*language*⟩ macro is defined or if the infix word macro definitions should be let equivalent to similar macros for another variant of the language. The above macro contains the definitions of "name" macros that contain the infix words such as "Chapter", "Table of contents", and the like. In this way the LaTeX kernel macros or the class macros use such "name" macros without the need to have different definitions for each language; these different definitions are necessary, but are included in each language definition file.
5. It defines how to set the date; in particular it redefines the macro `\today` to adapt its format and month name to the specific language. Everybody knows that even in English the American-style date follows the format "⟨*month*⟩ ⟨*day number*⟩, ⟨*year*⟩", while the British style follows the format "⟨*day number with ordinal suffix*⟩ ⟨*month*⟩ ⟨*year*⟩". Other languages may follow different formats. The Scandinavian countries use the numerical ISO format "⟨*year*⟩-⟨*month*⟩-⟨*day*⟩"; others may use their ordinal suffixes and prepositions before the month name and/or year number. In addition some languages may require other date formats, such as the roman numbering or other kinds of numbering. Historical eras generally are omitted, since the current era is assumed; but some languages accept negative year numbers so that special macros have to be created for them.
6. Some languages require shortcuts in order to perform frequent tasks; for this purpose they generally use *active characters* to be defined as it is necessary for any specific task. Active characters are of category 13. The most common one is the tilde, "~", that is a LaTeX kernel specific command used to insert a non breakable space between two words; it is generally named "tie". Most languages define other active characters, the most frequent of which is the straight double quote "; but according to the specific needs of each language other active characters may be necessary. In French the "high" punctuation marks are defined as active characters in order to automatically insert an unbreakable space before the regular punctuation glyphs; similarly an unbreakable space is necessary after open guillemets[4] and before closed guillemets. In Italian such spaces indadvertedly inserted in the source file must be eliminated. In ecclesiastic Latin such guillemet spaces are optional, but if used they must have a thinner unbreakable space than in French. In German quotation marks are different from those used in other western languages and require special shortcuts.
   The actual German shortcuts are shown in table 6 that replicates the table contained in the `ngerman.ldf` file.
7. Some languages define also characters that are active only in math mode; in Italian, for example, the comma is optionally defined in a specific way so that in math mode it recognises if it is followed by a digit, so as to behave as a decimal separator, else it behaves as a punctuation mark.
8. Other settings may be optionally activated or deactivated at user command; fore example, when typesetting Latin text, prosodic marks can be turned on or off with the specific user commands `\ProsodicMaksOn` and `\ProsodicMarksOff`. In Italian even the straight double quotes can be activated or deactivated.
9. In any case each language definition file has to activate its macros and general settings, because the situation must be restored upon changing language.
10. All language definition files for babel must be pure ASCII texts; no accented characters are allowed, because the file must be "encoding

4. The babel documentation uses the french word *guillemet*, but the code uses commands such as `\guillemotleft` and `\guillemotright`.

Table 6: Shortcuts defined for the German Language with the new orthography

''a	Umlaut ⟨ä⟩ (shorthand for \''a). Similar shorthands are available for all other lower- and uppercase vowels (umlauts: ''a, ''o, ''u, ''A, ''O, ''U; tremata: ''e, ''i, ''E, ''I).
''s	German ⟨ß⟩ (shorthand for \ss {}).
''z	German ⟨ß⟩ (shorthand for \ss {}). Differs to ''s in uppercase version.
''S	\uppercase {''s}, typeset as ⟨SS⟩(⟨ß⟩ must be written as ⟨SS⟩ in uppercase writing).
''Z	\uppercase {''z}, typeset as ⟨SZ⟩. In traditional spelling, ⟨ß⟩ could also be written as ⟨SZ⟩ instead of ⟨SS⟩ in uppercase writing. Note that, with reformed orthography, the ⟨SZ⟩ variant has been deprecated in favour of ⟨SS⟩ only.
''|	Disable ligature at this position (e. g., at morpheme boundaries, as in Auf''|lage).
''-	An additional breakpoint that does still allow for hyphenation at the breakpoints preset in the hyphenation patterns (as opposed to \-).
''=	An explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain -); useful for long compounds such as IT''=Dienstleisterinnen.
''~	An explicit hyphen without a breakpoint. Useful for cases where the hyphen should stick at the following syllable, e. g., bergauf und ''~ab .
'''	A breakpoint that does not output a hyphen if the line break is performed (consider parenthetical extensions as in (pseudo''~)''wissenschaftlich).
''/	A slash that allows for a line break. As opposed to \slash {}, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.
''‘	German left double quotes ⟨„⟩.
'''	German right double quotes ⟨“⟩.
''<	French/Swiss left double quotes ⟨«⟩.
''>	French/Swiss right double quotes ⟨»⟩.

neutral". This is true even for LGR encoded Greek fonts, where the specific LICR macros must be used to spell the infix Greek words. The situation is different for polyglossia because this package works only with LuaLaTeX and XeLaTeX that can use OpenType fonts in order to manage many different scripts and certainly also non-ASCII glyphs.

We describe a language definition file prepared for polyglossia and an unusual language: Occitan. This choice is just to show a possible situation to face when a user has to deal with a rare language; there are dozens of minority languages that might be the object of essays or linguistic research. Of course the linguist who is doing research in this field must know the minority langue s/he is researching on, therefore s/he should not have difficulties creating the pattern files and even less difficulty creating the relative language description file.

5.1 The Greek case with LuaLaTeX

The differences between LuaLaTeX and the other typesetting programs consist mainly in the fact that pattern files are loaded at run time, and only for the languages explicitly named through the calls by the \setmainlanguage, \setotherlanguages, and \setotherlanguage in the preamble of the main source file. In the past it used to be impossible to load different pattern files for the language variants, at least if was difficult and no user commands were available. The gloss-greek.ldf language description file worked correctly with polyglossia when the document was typeset with XeLaTeX, but could not correctly work with LuaLaTeX.

Recently the package luahyphenrules (Bezos, 2016) was added to the TeX Live distribution. By means of this package it is possible to let LuaLaTeX behave as XeLaTeX by loading at run time the pattern files for the referenced languages and their variants as they are listed in the service file language.dat used by XeLaTeX; therefore even LuaLaTeX, by using this package, has available the hyphenation pattern files for all variants of Greek. I therefore created for myself a patched gloss-greek.ldf that contains such enhancement; the patch is discussed further on. Please remember, though, that the greek language name is used in the \setmainlanguage and \setotherlanguage mandatory arguments, while in the language changing specific commands the name used in the \HyphenRules argument must be used to refer to a specific variant; therefore in a document body written in English, a Greek citation in ancient Greek requires the Greek text to be enclosed in a normal greek environment, but within its body any command that refers to the language should use the ancientgreek argument, as it it shown with code 7.

The patch shown starting on page 20 is relatively simple and certainly should be improved. Notice that the code contained between the lines that mark the start and end of the pack are partially additions (from lines 19 to 40) and partially modifications of some existing code in the unpatched

Code 7: The document to test the patched `gloss-greek.ldf`

```
% !TEX encoding = UTF-8 Unicode
% !TEX TS-program = LuaLaTeX

\documentclass[12pt]{article}

\usepackage{fontspec}
\setmainfont{CMU Serif}

\usepackage{polyglossia}
\setmainlanguage{italian}
\setotherlanguage{english}
\setotherlanguage[variant=ancient]{greek}

\begin{document}
\begin{otherlanguage}{greek}
Language: \the\language\ \languagename
\iflanguage{ancientgreek}{\ αὐτῇ}{}
\end{otherlanguage}

\end{document}
```

`gloss-greek.ldf` file. The dots in line 102 represent the remaining untouched part of the original `gloss-greek.ldf` file.

The name of the modified file remains the same, because polyglossia reads that very file for that language; in order to avoid conflicts this file may be saved in the personal `texmf` tree, so that the typesetting languages find this personal copy with precedence to the standard one; as mentioned in this paper, this personal copy should be renamed or deleted if and when a new release of official one contains a similar or better functionality.

The patch may exhibit innocuous error messages, but I always got out the correctly typeset document. I tested this patch with the document shown in the code listing 7 with both LuaLaTeX and XeLaTeX (with an "intelligent" shell editor capable to interpret the autoconfiguration initial comments, this amounts to change the prefix `Lua` with `Xe` in line 2 of the code shown in the code listing 7), by replacing the option `ancient` with `poly` or `mono`, and, correspondingly, by replacing the language name `ancientgreek` with `polygreek` or `monogreek` in line 17. If in line 15 the environment `otherlanguage` argument is set to `italian` or `english` the current language number and name is printed, but the Greek word is omitted. The attentive reader who is going to test the given code will notice that with both typesetting engines the language number for `english` is always 0 (zero), while for the other languages the number shown when using XeLaTeX is one unity lower than that shown when using LuaLaTeX; it is not an error, but it is implicit in the background macros that are being used to do the job; this is not the right place to discuss such details.

5.2 The Occitan language description file for polyglossia

Let us comment the code shown in appendix B. Lines 1 and 2 contain the usual *incipit* that every file should have; language definition files for babel have the almost equivalent statement `\ProvideLanguage`.

Lines from 3 to 9 contain the general settings for the language, in particular the parameters `hyphenmins` whose values represent the minimum lengths of the first and respectively the last word fragments when hyphenation can take place. The boolean flag `frenchspacing` set to `true` abolishes the different space factor to use for the full stop and the other "high" punctuation marks (as it is customary in English, but not in French and many other languages). The other boolean parameter `indentfirst` set to `true` requires that also the first paragraph of a sectional unit is indented as the paragraphs that follow it. Finally the boolean flag `fontsetup` set to `true` allows polyglossia to associate a specific font to a specific language.

Line 10 authorises to use and define those babel commands that define shorthands (shortcuts) for this specific language. Lines 12 to 16 load the specific babel module to define shorthands.

From line 21 to line 31 the straight double quote character is defined active and is given the main definition to distinguish its role in math vs text mode. Some service macros and the specific alternatives that the active character can execute, together with the macro to turn the active char off, are defined in lines 32 to 62.

From line 63 to line 85 the infix names are defined; since this code is to be used with polyglossia, accented characters are freely used. The Occitan date requires a special way to prefix month names and to write day numbers; actually the cardinal number 1 is substituted with its ordinal numeral, while the other day numbers are typeset with digits.

From line 76 to the end of the file there are the "housekeeping" macros to be executed when the language is set and when it is reset. Notice the reference to the two byte code point `''2019` that refers to the apostrofe; when setting Occitan the apostrophe is given its specific address, a positive number, as the lower case code, so that it is treated as a legal word character; when resetting its `\lccode` is given the value zero (`\z@` stands for 0).

5.3 The Cimbrian language definition file

Let us assume that a philologist/linguist wants to write an essay on the Cimbrian language. This is a (rare) germanic language spoken in northern Italy by a small community estimated to two thousand people; of course this number is below the critical mass, so that UNESCO classifies it as an endangered language (Wikipedia, 2018). It is reasonable

that linguists would like to record its lexicon, its pronunciation, its grammar, and so on. A sample of Cimbrian, at least one of its varieties, taken from Wikipedia (2018), is shown in page 22 together with its German and English translations.

But in order to use the TEX system programs our linguist should create the necessary equipment for typesetting anything in Cimbrian. Therefore he has to create the Cimbrian pattern file, and the Cimbrian language description file.

In a first step our linguist could concentrate on the language description file. The model offered from the Occitan file can be used; all occurrences of the string `occitan` should be changed to `cimbrian` and the short string `oc` appearing in some macros should be changed to `cim`; according to the ISO 639-3 regulation, `cim` is the official letter code for this language. Our linguist should not have any problems replacing the Occitan infix words and the date strings in order to comply with the Cimbrian language. He should pay attention to the apostrophe, possibly specifying the same settings used for Occitan.

But missing a Cimbrian pattern file, our linguist could temporarily specify among the `\PolyglossiaSetup` command arguments `hyphenation = germanb`; after all Cimbrian is a far relative of German. The short Cinmbrian text in page 22 has been typeset with the German hyphen patterns and apparently none of the few line breaks appears to be wrong. In his experiments at the beginning he will find wrong hyphen points, but while experimenting with various texts he collected from the local Cimbrian speaker area, he might classify a certain number of corrections to add to the German patterns. He therefore copies the `hyph-de-1901.txt` or `hyph-de-1996.txt` file to a new file `hyph-cim.txt` (the `.txt` extension is specific for LuaLATEX); he will replace the prologue of the file with suitable information and licence statement. He will gradually replace or add new patterns according to the rules specified in previous sections and such that the hyphenation errors in his Cimbrian text vanish.

In order to speed up this work our linguist is recommended to use the testhyphens package (Beccari, 2015). He creates himself a working directory, say, CimbrianTestFolder, where he saves the above named `hyph-cim.txt` new file. In this directory he creates and saves the following source `.tex` file, naming it, say, `TestCimbrianHyphens.tex`:

```
% !TEX TS-program = LuaLaTeX
% !TEX encoding = UTF-8 Unicode
\documentclass[12pt]{article}
 \usepackage{fontspec}
 \defaultfontfeatures{Ligatures={NoCommon,
 NoDiscretionary, NoHistoric, NoRequired,
 NoContextual}}
 \setmainfont{CMU serif}
\usepackage{luacode}
\usepackage{testhyphens}
\usepackage{multicol}

\begin{luacode}
local patfile = io.open('./hyph-cim.txt')
langobject = lang.new()
lang.patterns(langobject, patfile:read('*all'))
patfile:close()
\end{luacode}

\language=%
   \directlua{tex.sprint(lang.id(langobject))}
\begin{document}

Language \the\language

\begin{multicols}{2}
\lefthyphenmin=3
\righthyphenmin=2
\lccode`\'=\`\' % possibly needed
                % to handle apostrophes
\begin{checkhyphens}
% Cimbrian word list
...
\end{checkhyphens}
\end{multicols}
```

He will change the values of the `...hyphenmin` macros to the values he thinks to be adequate to the Cimbrian language; the numbers shown are the most widely used among the more than eighty languages dealt with by the TEX system. He will delete the comment line `% Cimbrian word list` and the line of dots that immediately follows, and replace them with a list of Cimbrian words; for example the words that form his quoted Cimbrian texts; for his convenience he will copy his whole text and globally put after each inter word space a new line character so as to have one word followed by one space per line; he will resort to a suitable text editor that can sort the words; in the worst case he can use a command line command to sort the file lines. By so doing he can easily delete duplicate words.

At this point he processes the document with LuaLATEX. The program loads the specified pattern file and the specified packages and produce the PDF output file. The format is a two column document containing all the input words, one per line, fully hyphenated. In this way it is very easy to control the result, find possible errors, correct the `hyph-cim.txt` file accordingly, and restart the process.

When there will be no more errors our linguist finds another Cimbrian text, adds it to the word list; sorts the words one per line, eliminates the duplicates and restarts the process.

There is some work to do, but having available a decent number of Cimbrian texts, the process may last few hours in total; it requires a lot of intelligence in order to analyse every hyphenated word so as to decide if the hyphens are correct;

Cimbrian

Pan khriage dar forte vo Lusern hat se gebeart gerecht. Di earstn tage von khriage, dar kommandant a Tschechoslowako hebat in forte gebelt augem un hat ausgezoget di bais bandiara un is vongant pin soldan. A trunkhantar soldado alua is no gestant sem in forte. Bia da soin zuakhent di Balischan zo giana drin in forte, is se darbkeht dar trunkhante soldado un hat agehevt z'schiasa.

German

Während des Krieges wehrte sich die Festung von Lusern vortrefflich. Die ersten Tage wollte sie ein tschechischer Kommandant aufgeben, indem er die weiße Fahne hisste und mit der Besatzung abzog. Nur ein betrunkener Soldat blieb zurück in der Festung. Als die anstürmenden Italiener in die Festung eindringen wollten, um sie in Besitz zu nehmen, erwachte der betrunkene Soldat von seinem Rausch und fing an, das Maschinengewehr knattern zu lassen.

English

During the war, the fort of Lusern resisted superbly. In the first few days a Czech commander wanted to give up, hoisting the white flag and withdrawing the garrison. Only one drunken soldier remained in the fort. When the Italians came storming into the fort to occupy it, the drunken soldier awoke from his intoxication and began to let the machine gun rattle.

in case they are not, intelligence comes again to rescue to decide which patterns to correct or which new patterns to add to the pattern file.

Once the work is done our linguist is very satisfied; he can write his paper in/on Cimbrian, but as a LaTeX user he feels necessary to share his work with other TeXies. He therefore sends his `hyph-cim.txt` file to the TeX-hyphen Team. On turn the Team processes the file, transforms it to the other formats needed by pdfLaTeX, XeLaTeX and pLaTeX (the version that is used in Japan to handle Japanese fonts but also for writing texts in western languages on any subject); the Japanese TeXies are very active in every discipline and may be interested even in studying minority European languages. The Team provides also to edit the various service macros necessary to configure the format file initialisation; not only, but they maintain the whole procedure running smoothly for the benefit of the whole community.

6 The service files

Anyone who creates pattern files and/or language description files does not need only a structure to test the patterns as described above. That person probably wants to use those files for his/her documents. Therefore s/he has to install those files so as to make them visible to the TeX system even before the Teams working at TUG installs them in an update for the general TeX user community.

It is therefore time to use a personal TeX directory tree. With TeX Live this tree is rooted in the user `$HOME` directory, where that symbol means different things with different operating systems: for Linux platforms it is `~/`; for Mac it is `~/Library/`; for recent Windows operating systems it is `C:\Users\`⟨*username*⟩`\`. The tree base directory is named `texmf` and its ramification should be a subset of the main TeX system tree one; therefore it has a ramification such as `tex/latex/` to which another directory such as `MyTeXLiveFiles` may be added. This directory shall contain the language description files; with the TeX Live installation it is not necessary to update the file name database.

Things are more complicated with pattern files; they should be saved into other branches of the personal tree, but the really important files are the `language-local.dat`, `language-local.def`, and `language-local.dat.lua` ones; such files must contain only the personal additions or deletions to the already existing files; by running

```
tlmgr generate language
```

with administrator or root privileges. This action regenerates the various `language.*` needed to rebuild the format files.

Eventually, having created the proper files and saved them in the proper folders, the final touch is to run, with administrator or root privileges, the `fmtutil-sys` program specifying the format files where it is desired to install the new language functionalities; in general they will be pdfLaTeX, LuaLaTeX and XeLaTeX. Personally I prefer to recreate only the pdfLaTeX and LuaLaTeX formats.

Caution: if and when the new functionalities are added to the TeX Live distribution, the branches of the personal tree just discussed should be eliminated from the computer. Any upgrade of the TeX Live distribution never touches the personal tree so that they are not upgraded any more unless the user provides by him/herself.

7 Conclusion

What has been described in this paper describes what there is behind the language processing done by the various typesetting programs based on LaTeX.

The processing steps needed to extend such language processing tasks are all pretty delicate, but after all they are not so complicated. They require a lot of work and we, the LaTeX users, are grateful to the various contributors of pattern and language description files. They saved us a lot of work. But if we have a sufficient knowledge of a language, we

too can contribute our work to save a lot of work to the members of the TEX community.

Acknowledgements

A special grateful thank-you is due to the TEX-hyphen Working Group; they do a marvellous work, and without them language management would be at the level it was in the mid-nineties.

References

Beccari, Claudio (2015). *The `testhyphens` package*. TUG. Readable with `texdoc testhyphens`.

Berry, Karl (2018a). *The TEX Live Guide – 2018*. TUG. Readable with `texdoc texlive`.

— (2018b). *TLMGR(1)*. TUG. Readable with `texdoc tlmgr`.

Bezos, Javier (2016). *luahyphenrules –Loading patterns in LuaLATEX with language.dat*. TUG. Readable with `texdoc luahyphenrules`.

Braams, Johannes e Javier Bezos (2018). *Babel*. TUG. Readable with `texdoc babel`.

Charette, François e Arthur Reutenauer (2018). *Polyglossia*. TUG. Readable with `texdoc polyglossia`.

Liang, Frank (1983). «Word Hy-phen-a-tion by Com-put-er». The original thesis, defended at Standford University, was scanned and made available at `www.tug.org/docs/liang/liang-thesis-hires.pdf`.

Liang, Frank e Peter Breitenlohner (1991). *PATGEN(1)*. TUG. This is the PDF version of the manual revision after the upgrade of 1991 to adapt the program to TEX 3. Readable with `texdoc patgen-man1`.

Schlicht, R. (2018). *The `microtype` package –Subliminal refinements towards typographical perfection*. TUG. Readable with `texdoc microtype`.

UNI 6461 (1969). *Divisione delle parole in fin di linea*. Ente Italiano di Unificazione, Milano.

Wikipedia (2018). «The Cimbrian Language». `https://en.wikipedia.org/wiki/Cimbrian_language`. Last checked 2018-11-10.

A Patch to modify the gloss-greek.ldf in order to have LuaLaTeX load the pattern files for the three Greek variants

```
\ProvidesFile{gloss-greek.ldf}[polyglossia: patched module for Greek]
\PolyglossiaSetup{greek}{
  script=Greek,
  scripttag=grek,
  frenchspacing=true,
  indentfirst=true,
  fontsetup=true,
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% The code in this file was initially adapted from the antomega
%% module for Greek. Currently large parts of it derive from the
%% package xgreek.sty (c) Apostolos Syropoulos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This file imported from xgreek fixes the \lccode and \uccode
% of Greek letters:
\input{xgreek-fixes.def}
%  ------------------------------------------------- BEGIN PATCH
\ifx\directlua\undefined\else\RequirePackage{luahyphenrules}\fi

\define@key{greek}{variant}{monotonic}{%
   \ifx\directlua\undefined\language\l@monogreek
   \else\HyphenRules{monogreek}\fi%
}
\define@key{greek}{variant}{mono}{%
   \ifx\directlua\undefined\language\l@monogreek
   \else\HyphenRules{monogreek}\fi%
}
\define@key{greek}{variant}{polytonic}{%
   \ifx\directlua\undefined\language\l@polygreek
   \else\HyphenRules{greek}\fi%
}
\define@key{greek}{variant}{poly}{%
   \ifx\directlua\undefined\language\l@polygreek
   \else\HyphenRules{greek}\fi%
}
\define@key{greek}{variant}{ancient}{%
   \ifx\directlua\undefined\language\l@ancientgreek
   \else\HyphenRules{ancientgreek}\fi%
}

%TODO: set these in \define@key instead:
\ifx\l@greek\@undefined
  \ifx\l@polygreek\@undefined
    \xpg@nopatterns{Greek}%
    \adddialect\l@greek\l@nohyphenation
  \else
    \let\l@greek\l@polygreek
  \fi
\fi
\ifx\l@monogreek\@undefined
  \xpg@warning{No hyphenation patterns were loaded for Monotonic Greek\MessageBreak
         I will use the patterns loaded for \string\l@greek instead}
  \adddialect\l@monogreek\l@greek
\fi
\ifx\l@ancientgreek\@undefined
  \xpg@warning{No hyphenation patterns were loaded for Ancient Greek\MessageBreak
```

```
         I will use the patterns loaded for \string\l@greek instead}
  \adddialect\l@ancientgreek\l@greek

%set monotonic as default
\def\greek@variant{\l@monogreek}%                monotonic
\def\captionsgreek{\monogreekcaptions}%
\def\dategreek{\datemonogreek}%
\fi

\def\tmp@mono{mono}
\def\tmp@monotonic{monotonic}
\def\tmp@poly{poly}
\def\tmp@polytonic{polytonic}
\def\tmp@ancient{ancient}
\def\tmp@ancientgreek{ancientgreek}

\define@key{greek}{variant}[monotonic]{%
  \def\@tmpa{#1}%
  \ifx\@tmpa\tmp@mono\def\@tmpa{monotonic}\fi
  \ifx\@tmpa\tmp@poly\def\@tmpa{polytonic}\fi
  \ifx\@tmpa\tmp@ancientgreek\def\@tmpa{ancient}\fi
  \ifx\@tmpa\tmp@polytonic%                       polytonic
    \def\greek@variant{\l@greek}%
    \def\captionsgreek{\polygreekcaptions}%
    \def\dategreek{\datepolygreek}%
    \edef\greek@language{\noexpand\language=\greek@variant}
    \xpg@info{Option: Polytonic Greek}%
  \else
    \ifx\@tmpa\tmp@ancient%                       ancient
      \def\greek@variant{\l@ancientgreek}%
      \def\captionsgreek{\ancientgreekcaptions}%
      \def\dategreek{\dateancientgreek}%
      \edef\greek@language{\noexpand\language=\greek@variant}
      \xpg@info{Option: Ancient Greek}%
    \else %                                       monotonic
      \def\greek@variant{\l@monogreek}% monotonic
      \def\captionsgreek{\monogreekcaptions}%
      \def\dategreek{\datemonogreek}%
      \edef\greek@language{\noexpand\language=\greek@variant}
      \xpg@info{Option: Monotonic Greek}%
    \fi
  \fi}

% ------------------------------------------------------ END PATCH
% The original code continues hereafter
...
\endinput
```

B The Occitan language definition file

```
\ProvidesFile{gloss-occitan.ldf}[2016/02/04 v0.3 polyglossia:
     module for Occitan]
\PolyglossiaSetup{occitan}{
  hyphennames={occitan},
  hyphenmins={2,2},
  frenchspacing=true,
  indentfirst=true,
  fontsetup=true,
}
```

```
\define@boolkey{occitan}[occitan@]{babelshorthands}[true]{}

\ifsystem@babelshorthands
  \setkeys{occitan}{babelshorthands=true}
\else
  \setkeys{occitan}{babelshorthands=false}
\fi
\ifcsundef{initiate@active@char}{%
\input{babelsh.def}%
\initiate@active@char{''}%
}{}
\def\occitan@shorthands{%
  \bbl@activate{''}%
  \def\language@group{occitan}%
  \declare@shorthand{occitan}{''}{%
    \relax\ifmmode
      \def\xpgoc@next{''}%
    \else
      \def\xpgoc@next{\futurelet\xpgoc@temp\xpgoc@cwm}%
    \fi
  \xpgoc@next}%
}
\def\xpgoc@@cwm{\nobreak\discretionary{-}{}{}\nobreak\hskip\z@skip}
\def\xpgoc@ponchinterior{%
      \nobreak\discretionary{-}{}{\mbox{$\cdot$}}\nobreak\hskip\z@skip}
\def\xpgoc@cwm{\let\xpgoc@@next\relax
  \ifcat\noexpand\xpgoc@temp a%
                          \def\xpgoc@@next{\xpgoc@@cwm}%
  \else
    \if\noexpand\xpgoc@temp \string|%
      \def\xpgoc@@next##1{\xpgoc@@cwm}%
    \else
      \if\noexpand\xpgoc@temp \string<%
        \def\xpgoc@@next##1{«\ignorespaces}%
      \else
        \if\noexpand\xpgoc@temp \string>%
          \def\xpgoc@@next##1{\unskip»}%
        \else
          \if\noexpand\xpgoc@temp\string/%
            \def\xpgoc@@next##1{\slash}%
          \else
            \if\noexpand\xpgoc@temp\string.%
              \def\xpgoc@@next##1{\xpgoc@ponchinterior}%
            \fi
          \fi
        \fi
      \fi
    \fi
  \fi
  \xpgoc@@next}
\def\nooccitan@shorthands{%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{''}}%
}
\def\captionsoccitan{%
```

```
   \def\refname{Referéncias}%
   \def\abstractname{Resumit}%
   \def\bibname{Bibliografia}%
   \def\prefacename{Prefaci}%
   \def\chaptername{Capítol}%
   \def\appendixname{Annèx}%
   \def\contentsname{Ensenhador}%
   \def\listfigurename{Taula de las figuras}%
   \def\listtablename{Taula dels tablèus}%
   \def\indexname{Indèx}%
   \def\figurename{Figura}%
   \def\tablename{Tablèu}%
   \def\partname{Partida}%
   \def\pagename{Pagina}%
   \def\seename{vejatz}%
   \def\alsoname{vejatz tanben}%
   \def\enclname{Pèça junta}%
   \def\ccname{còpia a}%
   \def\headtoname{A}%
   \def\proofname{Demostracion}%
   \def\glossaryname{Glossari}%
}
\def\dateoccitan{%
   \def\occitanmonth{\ifcase\month\or
      de~genièr\or
      de~febrièr\or
      de~març\or
      d'abril\or
      de~mai\or
      de~junh\or
      de~julhet\or
      d'agost\or
      de~setembre\or
      d'octobre\or
      de~novembre\or
      de~decembre\fi
   }%
   \def\occitanday{\ifcase\day\or
      1èr\else% primièr
      \number\day\fi% all other numbers
   }%
   \def\today{\occitanday\space \occitanmonth\space de~\number\year}%
}
\let\xpgoc@savedvalues\empty
\AtEndPreamble{% the user or the class might define different values
  \edef\xpgoc@savedvalues{%
    \clubpenalty=\the\clubpenalty\space
    \@clubpenalty=\the\@clubpenalty\space
    \widowpenalty=\the\widowpenalty\space
    \finalhyphendemerits=\the\finalhyphendemerits}
}
\def\noextras@occitan{%
   \lccode\string'2019=\z@
   \nooccitan@shorthands
```

```
    \xpgoc@savedvalues
}
\def\blockextras@occitan{%
    \lccode\string'2019=\string'2019
    \clubpenalty=3000 \@clubpenalty=3000 \widowpenalty=3000
    \finalhyphendemerits=50000000
    \ifoccitan@babelshorthands\occitan@shorthands\fi
}

\def\inlineextras@occitan{%
    \lccode\string'2019=\string'2019
    \ifoccitan@babelshorthands\occitan@shorthands\fi
}
```

▷ Claudio Beccari
claudio dot beccari at gmail dot com