How to make a foreign language pattern file: Romanian

Claudio Beccari, Radu Oprea and Elena Tulei

Abstract

In this tutorial we show how to make hyphenation patterns for a language when you know the grammatical rules for hyphenation (if they exist). We also discuss some points related to typographical hyphenation when compared with grammatical syllabification.

1 Introduction

In this tutorial we complete the argument of multilanguage typesetting that we started in the previous article [3]; there the problem of pattern generation had been omitted because this topic is sort of self contained and deserves its own discussion.

The problem of creating patterns suitable for TEX's hyphenation algorithm is dealt with in Appendix H of Knuth [10], where a certain statement may induce TEX users to abstain from creating pattern tables "since patterns are supposed to be prepared by experts who are well paid for their expertise."

In a way this is desirable, so that pattern tables for the same language are not created continuously, leading to multiple, incompatible, different, dubious, erroneous pattern lists and frustrated users unable to find a reliable pattern set for the language they want to use.

In another sense, pattern creation is essential for the wide-spread dissemination of TEX, which is now being used at least for all the languages examined by Sojka and Ševeček [12], and certainly also for many other modern and ancient languages. An indirect proof is given by the existence of fonts for exotic languages, fonts that could not be of any use if nobody set texts in those languages — setting text implies breaking words at the end of the line, so that some sort of hyphenation patterns must be used.

When the first author of this article started his interest in hyphenation, he had to make up for patching or creating new patterns for his site because the patterns available for Italian at that time did not comply with national regulations and, to be honest, were quite poor. He ended up with new patterns that were published in *TUGboat* [2] and then made their way into the CTAN archives. Recently, version 4.1, which greatly improves hyphenation of technical terms, was submitted to the archive managers.

Due to the presence of several foreign visiting professors and undergraduate and graduate students in his Polytechnic, Beccari was asked to prepare versions of (LA)TEX suitable for French, Spanish, Catalan, and Romanian; for his own pleasure he added Portuguese. He decided to prepare a single multilingual version of the formats so that users would not have to keep different commands in mind and would have the possibility of switching back and forth between languages. The limitations included the following:

1. programs had to run on a cluster of VMS mainframes running a Pascal-derived "big TEX";
2. a large number of isolated and clustered UNIX workstations running C-derived versions of "big TEX", a countless number of DOS personal computers, which has the most varied hardware and software configurations, equipped with the most unpredictable versions of TEX (from version 1.5 (!) to 3.1415), as well as screen and printer drivers;
3. the DOS personal computers were mostly owned by students, whose limited budgets were not sufficient to upgrade and standardize their equipment.

Beccari therefore had to adopt a "poor man" policy so that `.tex` files edited and tested at home could also be run on the mainframes or the UNIX stations. This implied sticking to regular cm fonts and creating a set of "intelligent" accent macros[1] that could look ahead and do some intelligent discretionary break insertion. The results were satisfactory for all the cited languages except Romanian, where TEX runs yielded poor results (many overfull

[1] By "accent macro" we mean every command that places a diacritical mark over or under any letter.

and underfull hboxes) with line widths shorter than ca. 100 mm.

With the collaboration of the second and third authors (Oprea and Tulei), natives speakers of Romanian, we decided to prepare Romanian hyphenation patterns without being confined to cm fonts, yet still referring to the extended dc or em font setup, according to the Cork encoding (see Beccari, elsewhere in this issue [3]).

The attentive reader may ask himself: "Why didn't they simply retrieve the Romanian patterns that Sojka and Ševeček [12] mention had been prepared by Malyshev or by Samarin and Urvantsev?" The answer is simple: the articles mentioned in Sojka and Ševeček deal with Russian; the Romanian patterns are only mentioned in passing; Sojka probably got hold of them, but we did not succeed. Moreover, according to Sojka and Ševeček, these Romanian patterns make up a very large set (4121 patterns) that requires a trie size of 4599 words, which might be too large for a multilingual (LA)TEX implementation where they are supposed to coexist with six or seven other pattern sets (including the US-English patterns that one cannot do without).

2 Grammar vs. typography

Preparing hyphenation patterns for TEX is not a simple grammar exercise; one must know the grammatical rules (if they exist and are not too complicated) of the language for which patterns are being prepared, and one must also know the typographical rules valid for the specific country or for a generic good typographical practice in that language.

It is obvious that typographical hyphenation must comply with grammar rules; it is less obvious that the possible hyphen points *must not* coincide with syllable divisions, but are a subset of the latter. Just to set forth a simple example, in Italian the word *idea* is syllabified in `i-de-a`, but typographically this word has no hyphen points;[2] some of (or all) the syllables may show up again when the word is connected with other words, as in *dell'idea* that is divisible like this: `del-l'i-dea`.

Syllables and typographic divisions differ in another way; in some when reading their mother language, some people feel uncomfortable if they have to read a word split across a hiatus (that, is a pair of vowels that do not form a diphthong); more generally, they feel uncomfortable if the word fragment that goes to the next line begins with a vowel that does not serve as a semi-consonant; this is not certainly the case in English-speaking countries, where a division such as *liq-uid*[3] is not only grammatically correct but also acceptable to the readers — such a division, on the other hand, makes Romance language speakers shiver unpleasantly! With Romance language hyphenation patterns, therefore, vocalic groups should remain undivided, except when semiconsonants are present.

So we have two different concepts, *hyphenation* and *syllabification*: the former deals with typography, while the latter deals with grammar.

A *syllable* is a word fragment that contains at least one vowel or equivalent sound; speakers of the language should be able to pronounce it as an isolated utterance; the set of vowels, diphthongs and triphthongs of a language is specific to that language. Sometimes we are astonished when we see foreign words that apparently do not contain any vowels; this is due to the way we are used to classifying the letters of our alphabet, but there is no doubt that in several Slavic languages the letter *r* plays the rôle of a vowel (*črn* 'black', *smrt* 'death', *Trst* 'Trieste').

The consonants on either side of such sounds may belong to the left or to the right syllable, depending on various rules and the acceptability of pronouncing consonantic clusters by native speakers of a specific language; however, etymological rules may alter this kind of division, which explains why syllabification rules are so different in different languages.

Unluckily TEX has no notion of vowels and consonants, so that one cannot rely on the fact that no (incorrect) breakpoints are executed that isolate groups of consonants; in fact, if you ask TEX to show the hyphen points for *comparands*, for example, while English is the default language, you get `com-para-nds`, which is clearly wrong.

Another point, valid for every language, is the existence of syllabification rules; we dare say that these rules exist for every official language, but what about dialects or regional unofficial languages? It is quite likely that the latter completely lack a standard and recognized set of grammatical rules, and that the language variety might change from village to village both in spelling and word usage. But if you must write a report on philological research dealing

[2] For the moment, let us skip the question of minimum length of the first and last "typographical" word fragments, that TEX deals with the values stored into the internal `\lefthyphenmin` and `\righthyphenmin`.

[3] US hyphenation verified in the Webster dictionary. UK hyphenation, as given by TEX by means of the pattern file `UKhyphen.tex`, is 'li-quid'. Although the UK hyphenation file is huge (55 860 bytes, 8527 patterns, 10 995 trie memory words, 224 ops), it hyphenates the words reported in Appendix H of *The TEXbook* in a funny way.

with such languages, you probably need to set many specimens of text, which means that you need patterns for these languages also.

But even if we examine only official languages, such rules, in particular those regarding syllabification, might be too complicated or might refer to word stress. In the case of complicated rules, the following statement holds true:

too complicated rules = no rules

As for word stress, TEX (or any other program that deals with words instead of numbers) does not know anything about stress, so that it cannot take it into account for hyphenation purposes.

In English (US and UK alike) the rules are quite complicated and refer to stress; the example shown by Knuth in Appendix H of *The TEXbook* regarding the word *record* is typical: the stress is different whether the word is used as a noun or as a verb, therefore the syllabification turns out to be `rec-ord` or `re-cord`! There is no simple way to get around these situations.

Another crucial point is the question of compound words, and prefixes and suffixes. Compound words should preferably be divided at word boundaries, while prefixes and suffixes may be treated in different ways in different languages.

In some languages compound words are very frequent; in other languages, although compound words exist, "compound concepts" are expressed without agglutination — the building of long and complicated compound words from many smaller ones — a feature of Germanic languages. Romance languages prefer constructions that make use of conjunctions and prepositions; English is different yet again, often putting together sequences of words that form a "compound concept" without "gluing" them to one another. For example, in English you have the compound list "manufacturing systems engineering" that in French becomes the prepositional phrase "genie des systèmes de fabrication". In any case, compound words are typical of the jargon found in chemistry, regardless of language, and, no doubt, since chemical names tend to be very long, they should be divided on the word boundaries rather than on syllable boundaries.

Prefixes and suffixes are generally treated in two different ways in different languages: prefixed and suffixed words must be divided according to their etymology, or they may be treated as regular common words, disregarding the presence of prefixes and suffixes. Italian and Portuguese belong to the second class, where prefixes and suffixes may be ignored, while most other languages require etymological division — Romanian belongs to this class.

Another problem with some languages is the fact that hyphenated words change spelling compared to when they are undivided; in German, for example, the orthographic sequence *ck* often gets hyphenated into `k-k`.

All these problems offer the willing pattern creator two alternatives: process a language dictionary with `patgen` or do everything by hand.

`patgen` is a program created by Liang for use with TEX (documentation available from CTAN). You have to feed the program with a large set of hyphenated words (several thousand words) and it will produce the hyphenation patterns that TEX requires. Producing patterns this way is time consuming because you need to create a large file of hyphenated words absolutely without error (either in spelling or hyphenation), but it is the only practicable way when dealing with languages for which the *no rules* statement applies. And this statement must be extended to all languages where stress plays an important rôle in syllabification, and where prefixes and suffixes must be divided *automatically*, according to etymology.

3 Mute vowels

Some languages — French for instance — have mute vowels; that is, vowels that are not pronounced, or are pronounced in an indistinct way. According to grammar rules, these vowels may produce a syllable, but typographic practice avoids splitting words which would leave a mute vowel to start the new line, especially if this happens on the last syllable. For French hyphenation patterns, M. Ferguson [9] explicitly inhibited hyphenation for all mute endings. This is another instance where syllabification and hyphenation are in contrast: if patterns are generated via `patgen`, the syllabified list of words fed to the program must take hyphenation breaks at mute vowels into account.

In any event, we want to stress the point that the word fragments obtained through the process of hyphenation are less than or equal to the number of syllables the word contains. There is no point in measuring hyphenation algorithm performance by counting the number of breakpoints it misses[4] (with a chosen word list), because the point is to measure the number of wrong breakpoints it produces. The

[4] Unless the algorithm is so poor that it misses most breakpoints!

best algorithm should produce no incorrect breakpoints (obviously) but apparently this is not achievable with any algorithm.

Beccari's pattern list for Italian `ithyph.tex`, version 3.5, was supposed to correctly hyphenate all Italian words; after additional checking with dictionaries created by L. Bianchi [6], a version 4.0 had to be produced, in spite of the fact that hundreds of people at his site had used the patterns and no one had ever found an incorrect breakpoint.

4 Compound words

`patgen` can also produce patterns for division of compound words, but the pattern list may become extremely complicated, and since compound technical words are continuously being created, any pattern list becomes obsolete the very moment it is created. We are of the opinion that it is too exacting a request that a computer program provide a complete, fast, accurate, and reliable algorithm suitable for every language and every situation. Some type of manual intervention is therefore necessary; below are some options which have proven useful to us.

(LA)TEX offers the macro `\-` that allows hyphenation in specific points. If you use `\-` within a word, this word can be hyphenated at that and only that position.[5]

Another macro with more flexibility is shown below:

```
\def\hz{\nobreak\hskip0pt \relax}
\def\allowhyphens{\hz\-\hz}
```

This new definition allows you to insert a discretionary break without inhibiting hyphenation in the rest of the word. The only drawback to such a definition is that the name is too long. German TEX users get around this problem by making the double quote character `"` active. Its definition is quite complicated because it has to do a lot of things in different circumstances with a minimum of keyboarding: the double quote prefixed to a vowel inserts the umlaut (dieresis), while prefixed to certain consonants it inserts the appropriate `\discretionary` command (for example, `"ck` expands to

```
\discretionary{k-}{k}{ck}
```

plus no breaks or zero skips to allow hyphenation in the rest of the word). Among other things, the double quote character can change `"-` into the "soft" discretionary break obtained via `\allowhyphens`, defined above.

Active characters are no problem, provided you add them to the lists of special characters whose "activity' or "specialness" is turned off when you go into verbatim mode. But it is a shame that all the standard input characters obtainable with a US keyboard have already been used for special TEX purposes.

In fact we would have preferred that *a single character* be used in place of the control sequence `\allowhyphens`, so as to speed up keyboarding. Moreover, when using dc fonts, it should be possible to refer to character `"17` (called a *compound word marker*). If such a character could be included within the hyphenation patterns, and if it could be easily inserted into the TEX input file, much keyborading could be saved. Unfortunately the character belongs to the set of the "illegal" ones that (LA)TEX refuses to read.

For a while, Beccari (compare [2]) used the underscore `_` as the single-character command, but other TEXies at his site were so used to using the underscore for other purposes (for example in file names) that his choice was doomed to failure; he had to delete the definition in order to avoid continuous quarreling with his colleagues.

In any case — call it `"-`, `\allowhyphens`, or `_` — this macro is suitable for separating prefixes, not for compound words, whose division should take precedence over syllable division; see the discussion in Sojka and Ševeček [12]. This question of marking the boundaries of compound words is still an open one, and we hope that the LATEX3 team finds a suitable solution.

5 Patterns

According to Appendix H of *The TEXbook*, a pattern is a sequence of lowercase alphabetic letters (and characters of category 12, "other") separated by digits.[6] The meaning of the digits d_i and the letters l_i is as follows: the pattern $d_1 l_1 d_2 l_2 \cdots l_{n-1} d_n$ implies that if the sequence of letters $l_1 l_2 \cdots l_{n-1}$ appears in a word, the hyphenation "weights" between such letters are d_1, d_2, ..., d_n, where odd digits allow hyphenation, even digits inhibit hyphenation and, if two or more (different) patterns specify different weights between the same letters, the highest digit prevails. The digit `0`, being the smallest, is optional and may generally be omitted.

A pattern list is a sequence of different patterns separated by spaces (or single end-of-line marks) as if they were words of a single paragraph given as arguments to the `\patterns` primitive command.

[5] Unless you have put several discretionary `\-` breaks in the same word.

[6] The special sign "." marks the beginning or end of a word.

Such a command may be processed only by the initialization version of TEX and is illegal while using (LA)TEX in its normal operating version. The order of patterns in the pattern list should not have any influence, but the list can be maintained much more easily if it is alphabetized on letters only (i.e. disregarding digits).

When extended characters are used, the patterns may contain control sequences, but these are restricted to those that expand to single characters. If you use the technique outlined in Beccari [3], you will have no problems, even if you use the standard (LA)TEX accent macros; just remember to put a digit (possibly `0`) after the control sequences that map to standard characters — i.e. `\oe`, `\ae`, `\aa`, `\o`, `\ss`, and ł.

6 Hyphenation exception lists

TEX can do an excellent job hyphenating words in a particular language but it is not perfect, because every language uses words borrowed from other languages or created with foreign roots and local endings. In some instances, especially when patterns are generated with `patgen`, unusual words may have been omitted from the list fed to `patgen` so that these words might get hyphenated incorrectly.

These exceptions can be addressed by using "hyphenation exception lists", one for each language, that consists of space-delimited hyphenated words given as arguments to the `\hyphenation` primitive. Originally Knuth prepared a list of 14 exceptions for English, but several years of usage have produced a continuously growing list of exceptions that is maintained in the CTAN archives. On this subject it is interesting to read the words by B. Beeton that accompany the 1992 US English exception log [4].

We are of the opinion that exception lists should be avoided as much as possible, at least for those languages with patterns made by hand: if manual creation was possible, it means that the syllabification rules were simple enough to consider most, if not all, normal situations. Hyphenation exception lists then become useful only in specific documents where unusual words are used: — first and/or family names, foreign toponyms, chemical compound names, and the like. The one exception to this seems to be English: while the number of hyphenation rules would seem to argue that the manual approach could not be considered reasonable, the language is widely used and a manually-prepared hyphenation exception log is regularly maintained and updated.

In most other languages hyphenation exception lists could become too difficult to create, especially if such languages are flexive.[7] With such languages — and all Romance languages belong to this class — the conjugation of a verb might include from 60 to 80 different forms, so that if an exception involves the stem of a verb, some 60 to 80 different entries must be made in the `\hyphenation` list for that one verb.

As a concluding remark it is worth noting that the patterns inserted via the `\patterns` command are static: once they have been processed by `initex`, you cannot change them, unless you create a new format. Hyphenation exceptions introduced via the `\hyphenation` command are dynamic, and you can add new exceptions at any point in your compuscript. Any new exception gets added to the list valid for the current language, and if a word is entered twice (supposedly with different hyphen points) the last one is the one TEX uses. Hyphenation exceptions are global and are not limited by group delimiters.

7 TEX hyphenation mechanism

TEX's hyphenation mechanism consists in examining each word to find if it appears in the `\hyphenation` argument of the current language; if so, it hyphenates the word accordingly; otherwise, it examines all possible patterns for dividing the word, patterns that must appear in the pattern list for the current language. This done, it compares and saves the highest digits that the several patterns have produced between the same pairs of letters, and inserts (implicit) discretionary breaks where odd digits appear.

To tell the whole truth, TEX actually does not insert such discretionary breaks before a certain number of letters from the beginning of the word and after another (possibly different) number of letters from the end; with version 3.xx of TEX, such numbers, call them λ and ρ respectively, can be set with the commands

$$\texttt{\lefthyphenmin} = \lambda$$
$$\texttt{\righthyphenmin} = \rho$$

With US English, the default values are $\lambda = 2$ and $\rho = 3$, but for UK English, the left limit is $\lambda = 3$; for Italian, where there are no mute vowels, it is correct to put $\lambda = 2$ and $\rho = 2$, although, if the line width is sufficiently large, $\lambda = 3$ and $\rho = 3$ is more elegant. These two numbers, therefore, relate to the

[7] 'Flexive' describes languages where nouns, adjectives and verbs have different forms (or spellings, if you wish) to show additional elements of meaning: singular or plural, masculine or feminine, case (nominative, accusative, etc., as in Latin) for nouns and/or adjectives, and the many forms due to verb conjugation.

typographic style, not to the hyphenation mechanism, and may be varied according to individual language/national typographical practice and to the particular style one prefers. This implies that patterns and hyphenation exceptions lists should not consider these two values and should produce correct breakpoints even if they are set at $\lambda = 1$ and $\rho = 1$.

8 Tools

Constructing a pattern list for a language whose hyphenation rules are not too complicated is not a difficult task; you just have to organize yourself with the following "tools":

1. a grammar or a very good personal knowledge of the language;
2. a dictionary where syllabification is shown;[8]
3. any national or language regulations, if they exist, that establish rules for typographical hyphenation;
4. if available, an excellent tool would be an orthographic dictionary in computerized form; for Italian we found the one prepared by Luigi Bianchi (to be used with `AMSpell` [6]);
5. a good handbook for typographical typesetting practice.

Another couple of tools are the `\showhyphens` macro provided by TEX, and a macro set provided by Eijkhout [8], called `\printhyphens`.

The first tool, `\showhyphens`, outputs TEX hyphenation on the screen and into the `.log` file in the form of the usual warning message for underfull hboxes. Eijkhout's macros instruct TEX to set one word per line with hyphens inserted at the breakpoints identified by TEX's hyphenation algorithm. The second approach is more elegant, and allows you to produce a clean hardcopy; its only drawback is that fragile commands tend to break up. `\showhyphens` is much simpler, but its output (from the `.log` file) does not contain the extended characters; on the other hand, the output may contain strange symbols[9] or "double caret" sequences, so that its interpretation requires a little skill by the user.

Then you should translate the "spoken" syllabification rules into "abstract" statements of the form:

> if v_i and c_i are vowels and consonants respectively, hyphenate as follows:
>
> - $v_1c_1v_2 \rightarrow v_1\text{-}c_1v_2$
> - $v_1c_1c_2v_2 \rightarrow v_1c_1\text{-}c_2v_2$
> - and so on (these rules are just examples and do not refer to any particular language)

If you succeed in translating all the "spoken" rules in the above form, then you can proceed to construct the pattern list. Otherwise, `patgen` is probably the only practical solution: you must start from the very beginning and construct a huge list (several thousand words) of perfectly spelled and hyphenated words taken from a language dictionary, then process this list by means of `patgen`. But before starting this heavy task be sure to do everything you can in order to find out if someone else has already done it; public archives are there for that purpose!

We will now describe the manual procedure. Something to keep in mind as you translate the spoken rules into abstract form: prepare a list of words that contain examples of application of the rules; counter examples are also precious elements in this list. Both can later be used to easily check the performance of your hyphenation algorithm.

9 Romanian syllabification "spoken" rules

Our colleague Tulei was able to produce a list of Romanian syllabification spoken rules in three forms which are not completely equivalent so that some intelligent interpretation must be introduced:

1. First form [1]:
   (a) if one vowel is followed by a single consonant, the latter belongs to the following syllable;
   (b) two vowels that do not form a diphthong belong to different syllables;
   (c) *i* and *u* between other vowels behave as semi-consonants and start a new syllable;
   (d) if a vowel is followed by several consonants the first consonant belongs to the left syllable and the other consonants to the right syllable with the exception of the following items;
   (e) if one of the consonants *b*, *c*, *d*, *f*, *g*, *p*, *t*, *v* is followed by *l* or *r*, the pair cannot be split;
   (f) the groups *ct*, *cţ*, and *pt* preceded by one or more consonants get split between the first and the second elements of the group;
   (g) prefixed words are separated according to etymology if the component words maintain their integrity.
2. Second form [13]:

[8] This is not a trivial recommendation; except for English, we do not know of any language for which syllabification is a standard feature of every dictionary.

[9] They are the symbols that the computer has in its internal tables for driving the screen or the printer in correspondence with the internal TEX character "numbers".

(a) a single consonant between two vowels belongs to the second syllable;

(b) two consonants between two vowels are separated unless the second is an *l* or an *r*;

(c) three or more consonants between two vowels are divided so as to leave one or two consonants with the second syllable, the latter case occurring when the last consonant is *l* or *r*;

(d) vowels forming a hiatus are divided;

(e) prefixed words are divided according to their etymology.

3. Third form [5]:

(a) vowels forming a hiatus are divided;

(b) a single consonant between two vowels belongs to the second syllable; for the application of this rule the digraphs *ch* and *gh* are considered a single consonant, as are digraphs imported from other languages which are not "regular" in Romanian words, such as, for example, *sh*;

(c) two consonants are divided unless an *l* or an *r* follows one of the consonants *b, c, d, f, g, h, p, t, v*;

(d) three consonants are divided after the first one if the group ends with *l* or *r*; otherwise, they are divided after the second consonant. The "regular" Romanian consonant triplets of the latter group are: *lpt, mpt, mpţ, ncs, nct, ncţ, ndv, rct, rtf, stm*;

(e) four or five consecutive consonants are divided after the first one, except in adapted words and neologisms.

For rule 3e the underlying concept seems to be that division must take place where the second syllable can be pronounced by a normal Romanian speaker. In other words, the second syllable must start with the longest set of consonants that can be found at the beginning of other Romanian words. The correct division is therefore `ang-strom`, not `an-gstrom` because there is no Romanian word starting with *gstr*, but there are plenty starting with *str*.

Another remark: among the grammars examined, none considers the group *ngv* that appears in at least one word that is linguistically important: *lingvistic*. By analogy with the previous remark, since no Romanian word starts with *gv*, we have decided to adopt the division `ng-v`.

The *cratima* or *liniuţă de unire* (intra-word dash or hyphen) is used to connect most compound words, but it is also used to mark vocalic elision, much as French and English use an apostrophe. This is a minor problem because TEX hyphenates words containing the intra-word dash or hyphen only in correspondence with the hyphen character; in the case where the *cratima* is used for marking vocalic elision there should be no line break.

Consonants:	b, c, d, f, g, h, j, k, l, m, n, p, r, s, ş, t, ţ, v, x, z
Vowels:	a, â, ă, e, i, î, o, u
Special letters:	q, w, y

Table 1: Classification of Romanian letters

10 Romanian "abstract" rules

Let us put the Romanian spoken rules into abstract form. As usual, let c_i be the consonants, l_i the liquid consonants *l* or *r*, v_i the vowels in general, o_i the vowels belonging to the set $\{a, â, ă, e, o\}$, and `x` is a specific letter (*x* in this example). This yields the following abstract rules:

$$v_1c_1v_2 \quad \to \quad v_1\texttt{-}c_1v_2 \tag{1}$$

if $c_1 \in \{l_i\}$ and $c_2 \notin \{l_i\}$ then

$$v_1c_2c_1v_2 \quad \to \quad v_1\texttt{-}c_2c_1v_2 \tag{2}$$

$$v_1c_3c_2c_1v_2 \quad \to \quad v_1c_3\texttt{-}c_2c_1v_2 \tag{3}$$

else

$$v_1c_2c_1v_2 \quad \to \quad v_1c_2\texttt{-}c_1v_2 \tag{4}$$

if $c_2c_1 = \texttt{ct}$ or
$c_2c_1 = \texttt{cţ}$ or
$c_2c_1 = \texttt{pt}$ or
$c_2c_1 = \texttt{pţ}$ then

$$v_1c_3c_2c_1v_2 \quad \to \quad v_1c_3c_2\texttt{-}c_1v_2 \tag{5}$$

else

$$v_1c_3c_2c_1v_2 \quad \to \quad v_1c_3\texttt{-}c_2c_1v_2 \tag{6}$$

$$v_1c_4c_3c_2c_1v_2 \quad \to \quad v_1c_4\texttt{-}c_3c_2c_1v_2 \tag{7}$$

$$v_1c_5c_4c_3c_2c_1v_2 \quad \to \quad v_1c_5c_4\texttt{-}c_3c_2c_1v_2 \tag{8}$$

end if end if

if $v_1 = v_2$ then

$$v_1v_2 \quad \to \quad v_1\texttt{-}v_2 \tag{9}$$

else if $v_1 = \texttt{i}$ or $v_1 = \texttt{u}$ then

$$o_1v_1o_2 \quad \to \quad o_1\texttt{-}v_1o_2 \tag{10}$$

end if

In order to apply the above rules it is necessary to define the sets of vowels and consonants; in Romanian we have the situation summarized in Table 1. The letters *q*, *w*, and *y* are classified as special because they appear only in words not strictly Romanian, that is imported or adapted from foreign languages; in order to hyphenate at least some of these imported words, such letters will be included in the patterns. Furthermore the digraphs *ch*, *gh*,

sh, and the like, that are not listed in Table 1, must be counted as a single consonant.

We considered hiati, diphthongs and triphthongs only to a limited extent. Romanian is very rich in diphthongs and triphthongs, but the same couples or triplets of vowels may be indivisible or form a hiatus in different words, or in the conjugation or declination of the same word they might play a different rôle and become divisible when they used not to be — and vice versa. Since it is so difficult to classify pairs or triplets of vowels as diphthongs or triphthongs, the best thing to do is not to divide them at all, except when rules 9–10 are applicable. Furthermore, let us remember the typographic point of view of avoiding line breaks within a vocalic group.

11 Romanian patterns

With Romanian, as with other languages where the rules refer to vowels and consonants, it is necessary to establish if the former or the latter play a more important rôle in hyphenation. If we exclude the division of vocalic clusters (except when rule 10 applies), there is no doubt that consonants are the discriminating elements, so that patterns may be built focusing on consonants.

When we want to apply rules 1–10, we must not implement the patterns by making all the combinations of letters implied by such rules. We would otherwise create a set of patterns that would be unnecessarily large, because it would contain combinations that never occur in the real language.

Furthermore we use the smallest weight-digits available, remembering that `0` is also implicitly used when we do not write anything. So we start with omitting all vowels, unless specifically required, and start producing the following patterns that should take care af all single intervocalic consonants, and all single consonants at the beginning and ending of words:

```
1b 1c 1d 1f 1g 1h 1j 1k 1l 1m 1n 1p 1q 1r
1s 1ş 1t 1ţ 1v 1x 1z
.b2 .c2 .d2 .f2 .g2 .h2 .j2 .k2 .m2 .p2
.s2 .ş .t2 .ţ .v2 .z2
2b. 2c. 2d. 2f. 2g. 2h. 2j. 2k. 2l.
2m. 2n. 2p. 2r. 4s. 2ş. 4t. 2v. 2x.
2z.
```

Now we consider the *l* and *r* rules starting with statement 2; in passing, we also add the digraphs *ch* and *gh*, the digraph *sh* that appears so often in foreign words, and the group *tz* that appears in several adapted technical words:

```
c2h g2h s2h t2z
b2l c2l d2l f2l g2l h2l k2l p2l t2l v2l
b2r c2r d2r f2r g2r h2r k2r p2r t2r v2r
```

Next, we allow word division between other consonants, inhibiting word division before the first one of the pair; as well, we introduce some patterns for dealing with the special letters *w* and *y*:

```
2bc 2bd 2bj 2bm 2bn 2bp 2bs b3s2t 2bt
2bţ 2bv
2cc 2cd 2ck 2cm 2cn 2cs 2ct 2cţ 2cv 2cz
2dg 2dh 2dj 2dk 2dm 2dq 2ds 2dv 2dw
2fn 2fs 2ft
2gd 2gm 2gn 2gt 2g3s2 2gv 2gz
2jm
2hn
2lb 2lc 2ld 2lf 2lg 2lj 2lk 2lm 2ln 2lp
2lq 2lr 2ls 2lt 2lţ 2lv 2lz
2mb 2mf 2mk 2ml 2mn 2mp 2m3s2 2mţ
2nb 2nc 2nd n3d2v 2nf 2ng 2nj 2nl 2nm
2nn 2nq 2nr 2ns n3s2a. n3s2ă n3s2e
n3s2i n3s2o n3s2cr n3s2f ns3h n3s2pl
n3s2pr n3s2t 2nş n3ş2c n3ş2t 2nt 2nţ
2nv 2nz n3z2dr
2pc 2pn 2ps 2pt 2pţ
2rb 2rc 2rd 2rf 2rg 2rh 2rj 2rk 2rl 2rm
2rn 2rp 2rq 2rr 2rs r3s2t 2rş 2rt 2rţ
2rv 2rx 2rz
2sb 2sc 2sd 2sf 2sg 2sj 2sk 2sl 2sm 2sn
2sp 2sq 2sr 2ss 2st 2sv 2sz
2şn
2şt
2tb 2tc 2td 2tf 2tg 2tm 2tn 2tp 2ts 2tt
2tv 2tw
2vn
1w wa2r
2xc 2xm 2xp 2xt
1y 2yb 2yl 2ym 2yn 2yr 2ys
2zb 2zc 2zd 2zf 2zg 2zl 2zm 2zn 2zp 2zr
2zs 2zt 2zv
```

Some of the patterns have a non-repetitive look in order to take care of rules 6, 7 and 8; for example `n3s2t` overrides `2st` so that a word like *monstru* can be hyphenated correctly as `mon-stru`. With just the "regular" pattern `2ns`, the hyphenation would have been `mons-tru`.

Finally, we introduce the patterns that involve vowels; in particular, we inhibit hyphenation when the last syllable (and the whole word altogether) ends with an *i*. This is useful because the final *i* is almost mute in the endings of some masculine non-articulated nouns and adjectives, and in such cases line breaks are not allowed [7]. But since TEX does not know anything about the language, we take

a conservative approach and inhibit division before any ending syllable terminated with an *i*.

```
a1ia a1a a1ia a1ie a1io ă1ie ă1oa â1ia
e1e e1ia i1i i2ii. 2i. o1o o1ua o1uă
u1u u1ia u1al. u1os. u1ism u1ist u1iţt
2bi. 2ci. 2di. 2fi. 2gi. 2li. 2mi.
2ni. 2pi. 2ri. 2si. 2şi. 2ti. 2tri.
2ţi. 2vi. 2zi.
```

Eventually we add a few patterns that work with some prefixes:

```
.ante1 .anti1
.contra1
.de3s2cri
.de2z1aco .de2z1amă .de2z1apro
.de2z1avan .de2z1infec .de2z1ord
.i2n1ad .i2n1am .i2n1of .î2n1ăsp
.î2n1ad .î2n3s2 .în3ş2
.ne1a .ne1î .nema2i3 .ne3s2ta
.re1ac .re1î
.su2b1ord .su2b3r
.supra1
.tran2s1 .tran4s3pl .tran4s3f
```

12 Tests and conclusion

We were able to obtain a small pattern file containing 350 patterns[10] and requiring 31 ops. This pattern set is much simpler than that mentioned in Sojka and Ševeček, and the results are simpler too. However, although the ratio between the number of patterns cited by these two authors and our set is approximately 12, our results are not 12 times poorer. It might be interesting to test them on a large set of Romanian words in order to compare the differences. As for our experience, we used these patterns to typeset a large number of Romanian documents but never experienced an incorrect line break.

Using Eijkhout's `\printhyphens` macro, we can verify what we got: feeding this macro some test words[11] we get the following:

a-do-les-cen-ti-lor	in-dus-triei
a-pro-band	in-e-gal
ba-ia	jert-fă
Bu-cu-resti	mon-stru
con-struc-tor	post-de-cem-bris-tă
dum-ne-zeu	vre-mea
drep-tu-lui	Wa-shing-to-nu-lui
dez-a-van-ta-jul	watt-me-tru
fla-shul	

[10] One pattern not mentioned here is `2-2`, which makes it possible to distinguish between the *cratima* (referred to in section 9) and the hyphen character, by using appropriate category codes.

[11] For testing purposes, we set $\lambda = 1$ and $\rho = 1$; in normal Romanian typesetting, it is better to set $\lambda = 2$ and $\rho = 2$.

Let us point out that with the help of such tools one can verify both the validity of the pattern set and if any patterns are missing. For example, if a word turns out to have incorrect hyphen points, it is possible to find out why: — wrong patterns? missing patterns? wrong weights? — and to proceed with corrections. This is how we discovered that we had originally missed the pattern `2tt`, which refers to a double consonant not present in "normal" Romanian; in fact, without this pattern, *wattmetru* gets hyphenated as `wa-tt-me-tru`, which is obviously wrong since it has a word fragment without vowels. Similar situations can be easily spotted by means of the word list that had been prepared together with the patterns, as suggested above. With the help of a dictionary it is possible to spot unusual or doubtful words, include them in the test word list, and find out if they get hyphenated correctly.

A few concluding remarks. Romanian is not particularly easy nor particularly difficult to hyphenate. If one gives up the possibility of TEX doing the whole task of separating the prefixes, and is willing to using `"-` or similar macros for inserting discretionary soft breaks in the prefixed words, the pattern file one gets is of very modest size although it produces correct hyphenation in most circumstances. Such a small file is compatible with the memory limitations of small implementations of TEX the program; (LA)TEX can be initialized with several hyphenation patterns at the same time, allowing the user to create a multilanguage tool that allows him/her to typeset texts in several languages without the need for changing software when passing from one language to another.

Acknowledgments

We would like to thank the precious cooperation of Mihai Lazarescu and Janetta Mereuta who helped very much with the queries and discussions on the Internet and with the creation of a test word list with normal and unusual Romanian words.

References

[1] A.V., *Îndreptar ortografic, ortoepic şi de punctuaţie*, Bucureşti, 1971 (ed. III-a).

[2] Beccari C., "Computer Aided Hyphenation for Italian and Modern Latin", *TUGboat* 13(1):23–33, April 1992.

[3] Beccari C., "Configuring TEX or LATEX for typesetting in several languages", *TUGboat* 16(1):18–30, March 1995.

[4] Beeton B., "Hyphenation Exception Log", *TUGboat* 13(4):452–457, December 1992.

[5] Beldescu G., *Ortografia actuala a limbii române*, Bucureşti: Editura Ştiinţifică şi Enciclopedica, 1985.

[6] Bianchi L., `italian.zip`. [Includes the complete set of tools for checking correct Italian spelling in ASCII and (LA)TEX compuscripts; available from `ftp.yorku.ca` or contacting directly L. Bianchi at `lbianchi@sol.yorku.ca`.]

[7] Breban V., Bojan M., Comşulea E., Negomineanu D., Şerban V., Teiuş S., *Limba română corectă*, Bucureşti: Editura Ştiinţifică, 1973.

[8] Eijkhout V., "The bag of tricks", *TUGboat* 14(4):424, December 1993.

[9] Ferguson M., `frhyph.tex`, `frhyph7.tex`, `frhyph8.tex`. [Hyphenation pattern files available from the CTAN archives in the directory `/tex-archive/languages/french`.]

[10] Knuth D.E., *The TEXbook*, Reading Mass.: Addison-Wesley, 1990.

[11] Pusztai A. and Ardelean Gh., *LATEX, Ghid de utilizare*, Bucureşti: Editura Tehnică, 1994.

[12] Sojka P., Ševeček P., "Hyphenation in TEX — Quo Vadis?", *Proceedings of the Eighth European TEX Conference* (Sept. 26–30, 1994, Gdańsk, Poland), 59–68.

[13] Zamsa E., *Limba română, recapitulari şi exerciţii*, Bucureşti: Editura Ştiinţifică, 1991.

Appendix

Useful shorthand macros

Although Table 1 shows that just five diacriticized characters appear in the Romanian alphabet, the language makes frequent use of these characters, so that the ASCII source `.tex` file is filled with sequences such as `\u{a}`, `\^a`, `\^{\i}`, `\c{s}`, `\c{t}` (and their uppercase counterparts), to the point where the source file is almost unreadable. A single word may contain several such characters, as for example *ţîşnitură*, `\c{t}\^{\i}\c{s}nitur\u{a}`, that contains four of them, so that to key in the source file is quite error prone, and finding and correcting possible errors turns out to be quite difficult.

If you have a Romanian keyboard, you can easily map the input Romanian characters (with internal code higher than 127) to the appropriate sequences: on the screen the text appears without backslashes and braces, but if you have to send your file to somebody else on some computer network global substitutions have to be made to put back the cumbersome sequences.

Another approach would be to make some character active and then define it in a suitable way, so that you may read your source file with a minimum of extra characters interspersed.

German and Dutch users, who have similar problems, have chosen the double quote `"`; nothing prevents us from doing the same for Romanian, but the double quote is used so many times explicitly or behind the scenes by TEX, especially for representing internal codes in hexadecimal notation, that we prefer another, more "innocent" character — the exclamation mark `!`.

A LATEX guide has been published recently in Romania [11], where the apostrophe sign has been made active and defined in such a way as to save a lot of keystrokes while keying in the source text. We wonder if the authors actually used the macros they suggest on page 109 of their guide because the `\newcommand` command cannot (ordinarily) operate on active caracters, but only on control words — they probably used a regular `\def` command. Several other TEX users around Romania[12] who answered our questions on the Internet revealed that they are using definitions similar to the ones published in the Romanian LATEX guide. But the same criticism remains: the apostrophe is used internally by the plain and LATEX format files for identifying octal numbers, and it becomes really difficult to create correct definitions so that character "activity" does not interfere with octal notation.

Therefore we will stick with our choice of the exclamation mark, but what we state here can be applied to any other "innocent" character. If you use babel, then you should add the exclamation mark to the special TEX characters and provide an argument to `\extrasromanian` such as, for example:

```
\addto\extrasromanian{%
  \babel@add@special\!}
\addto\extrasromanian{%
  \babel@remove@special\!}
\addto\extrasromanian{%
  \babel@savevariable{\catcode`\!}%
  \catcode`\!\active}
```

In the style file there must be a statement that memorizes the meaning of `!` before changing catcode:

```
\let\@xcl@@=!
```

and a set of definitions for the active exclamation mark:

```
\begingroup
\catcode`\!=13
\gdef!{\protect\p@xcl@}
\endgroup
```

[12] They are too numerous to be cited here, but we take this opportunity to thank them warmly.

Some macros must be defined in order to test if what follows the exclamation mark is a non-expandable token (i.e. a character token or a primitive TEX command), or has an expansion:

```
\def\@Macro@Meaning#1->#2?{%
              \def\@Expansion{#2}}
%
\def\TestMacro#1#2#3{%
    \expandafter
      \@Macro@Meaning\meaning#1->?%
    \ifx\@Expansion\empty
      \def\@Action{#3}%
    \else
      \def\@Action{#2}%
    \fi
    \@Action}
```

More definitions are needed, and the apparently tortuous path followed before arriving at the desired goal is due to the necessity of making such macros robust, so that they do not break apart when they appear in moving arguments, such as when writing to auxiliary files for cross-reference purposes or for preparing indices or tables of contents.

```
\def\p@xcl@{\ifmmode
              \@xcl@@
           \else
              \expandafter\ExCl
           \fi}
%
\def\ExCl{\futurelet\eXcl\exCl}
%
\def\excL{\Excl{}}
%
\def\exCl{\TestMacro\eXcl{\excL}{%
           \ifcat\eXcl a%
              \let\eXcL\Excl
           \else
              \ifcat\eXcl "%
                 \let\eXcL\Excl
              \else
                 \let\eXcL\excL
              \fi
           \fi
           \eXcL}}
```

Finally the definitions we were aiming at:

```
\def\Excl#1{\expandafter
\ifx\csname @xcl@#1\endcsname \relax
    \@xcl@@\space #1%
\else
    \csname @xcl@#1\endcsname
\fi}
%
\def\hz{\nobreak\hskip\z@}
```

!a	ă	!A	Ă
!i	î	!I	Î
!s	ş	!S	Ş
!t	ţ	!T	Ţ
!␣	!	!"	`
!-	-	!|	

Table 2: Sequences obtained with the active exclamation mark and their results. In math mode the exclamation mark preserves its meaning.

```
%
\expandafter
  \def\csname @xcl@a\endcsname{\u{a}}
\expandafter
  \def\csname @xcl@A\endcsname{\u{A}}
\expandafter
  \def\csname @xcl@s\endcsname{\c{s}}
\expandafter
  \def\csname @xcl@S\endcsname{\c{S}}
\expandafter
  \def\csname @xcl@t\endcsname{\c{t}}
\expandafter
  \def\csname @xcl@T\endcsname{\c{T}}
\expandafter
  \def\csname @xcl@i\endcsname{\^{\i}}
\expandafter
  \def\csname @xcl@I\endcsname{\^I}
\expandafter
  \def\csname @xcl@"\endcsname{^^12}
\expandafter
  \def\csname @xcl@-\endcsname{\char"7F}
\expandafter
  \def\csname @xcl@|\endcsname{\hz\-\hz}
```

When you select the Romanian language the exclamation mark becomes active and you get the shorthand notations summarized in Table 2. The last line of this table requires some additional comment.

1. The sequence `!|` produces a discretionary break that does not inhibit hyphenation in the rest of the word, and is useful for marking the boundary between a prefix and a word stem. This sequence complements the regular TEX sequence `\-` that inserts a discretionary break but inhibits hyphenation in the rest of the word.
2. The sequence `!-` produces the short dash that is used for connecting compound words (quite rare in Romanian); it inserts the `\hyphenchar` that TEX treats in a special way for what concerns line breaking. In fact, when this character appears, TEX breaks the line, if necessary, only after the short dash.

3. The frequent *cratima*, i.e. the short dash to be used in place of an elided vowel or for connecting enclitic pronouns (across which line breaks are forbidden), is obtained by the usual `-` sign, but to avoid having TEX treat it as the `hyphenchar`, it is necessary (a) to chose a different entry in the font table for the hyphen character, and (b) to assign the "minus sign" an `\lccode` different from 0, so that in text mode it can be treated as a regular letter and it is possible to enter a special pattern, `2-2` in the pattern list, so that line breaks are forbidden across it. By so doing, the number of ops increases by 1, but still remains very reasonable.

   The extended fonts actually include two short dashes, one with hexadecimal code `"2D`, and the other with hexadecimal code `"7F`. For Romanian, it is therefore necessary to assign a lowercase code to `"2D` so as to use it as a *cratima*, and to declare `"7F` to be the `\hyphenchar` for the current extended font.

   In order to revert to the original situation when a different language is selected, it is necessary to restore the `\lccode`s and the `\hyphenchar` assignments. We did not investigate how this is done in babel, but we did it with our implementation, which is less general, compared with the facilities offered by babel.

It is worth noting that the exclamation mark followed by a character different from one of those listed in Table 2 reproduces itself; since this mark is commonly used at the end of a sentence, and therefore is followed by a space or an end-of-line mark, a space is inserted by default by the macro expansion.

A sample (source) text follows, so as to appreciate the shorthand notations just introduced.

```
Problema desp!ar!tirii cuvintelor !in
silabe se pune mai ales !in scriere,
unde se ivesc dificult!a!ti c\^and
este vorba de vocale sau de consoane
succesive.  Dar problema desp!ar!tirii
!in silabe este str\^ans legat!a !si de
pro!|nun!tarea corect!a a unor cuvinte
care con!tin diftongi, triftongi sau
vocale in hiat, dup!a cum se va vedea
mai departe.

La categoria de nume geografice
teritorial!-administrative, {\em
!Indreptarul ortografic} prevede c!a
acestea !"se scriu cu ini!tial!a
majuscul!a la toate cuvintele
componente‘‘, preciz\^andu-se !in
acela!si timp c!a: !"Numele generice
{\em deal, fluviu, insul!a, lac, munte,
peninsul!a, r\^au, vale} etc., c\^and
nu fac parte din denumire, se scriu cu
ini!tial!a mic!a‘‘.
```

Note that *â* still requires the full accent sequence,[13] but it is just three keystrokes, compared with the numerous keystrokes required by the other special characters. Notice the sequence `!-` that connects a compound word (after which TEX will break the line if necessary) and the *cratima* that connects the enclitic pronoun (where hyphenation is forbidden). And finally, notice the reversed and lowered quotation marks and the soft discretionary inserted after the prefix *pro*.[14] The corresponding typeset text appears as such:

> Problema despărţirii cuvintelor în silabe se pune mai ales în scriere, unde se ivesc dificultăţi când este vorba de vocale sau de consoane succesive. Dar problema despărţirii în silabe este strâns legată şi de pronunţarea corectă a unor cuvinte care conţin diftongi, triftongi sau vocale in hiat, după cum se va vedea mai departe.
>
> La categoria de nume geografice teritorial-administrative, *Îndreptarul ortografic* prevede că acestea `se scriu cu iniţială majusculă la toate cuvintele componente", precizându-se în acelaşi timp că: `Numele generice *deal, fluviu, insulă, lac, munte, peninsulă, râu, vale* etc., când nu fac parte din denumire, se scriu cu iniţială mică".

◇ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
Email: beccari@polito.it

◇ Radu Oprea
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy

◇ Elena Tulei
Universitatea Tehnică de Construcţii
Bucureşti, România

[13] Up to a couple of years ago, *â* was used only in the word *România* and its derivatives. Since the last spelling reform this sign is used more often; in the example we have used the modern spelling.

[14] Actually there is no need for this discretionary break — it was put there just to show its use. With our patterns we did not succeed in finding a sample text containing something that really requires the use of a soft discretionary break.