Hyphenation in TEX and elsewhere, past and future

Mojca Miklavec and Arthur Reutenauer

1 The past eight years: hyph-utf8

Hyphenation, or word division, is an essential feature of TEX and related systems, which was a pioneer in the area. Frank Liang, a student of Donald Knuth, devised an algorithm to efficiently store the information that specifies how to break words. Liang's PhD thesis on the subject was published in August 1983, and TEX82 already included the algorithm. Liang also wrote the program `patgen` that, given a list of hyphenated words, produces a set of *hyphenation patterns* that embed the information.

TEX82 would store only one hyphenation table, but with TEX 3 in 1990 it became possible to include multiple pattern sets, identified by the value of the primitive `\language`. At the same time TEX's character set was extended from 7 bits to 8 bits, thus widening the range of supported encodings. This would prove essential for many languages. Development had indeed started early to devise sets of patterns appropriate for different languages; for example Italian, for which patterns were produced and described in *TUGboat* volume 5, issue 1 in 1984; and French and German, in the next issue. However, with only 7 bits to use, most languages needed a number of tricks to work correctly, some of which could rightly be called dirty, and which were kept even after TEX3 came along.

The terms of use of the different pattern sets, when there were any, were equivalent to those of TEX itself: free to use and distribute, and modified versions should have another name. Most of the time, however, there was no clear licence. When the LATEX Project Public Licence, the LPPL, was created in 1999, some authors adopted it for their patterns, and over the years the majority of the files became available under this licence.

When XETEX and LuaTEX were included in distributions, encoding became once again a problem since these engines expect UTF-8 input by default and couldn't accommodate the various 8-bit encodings the different pattern files were using. In order for XETEX to be added to TEX Live in 2007, its creator Jonathan Kew devised a solution whereby patterns were converted to UTF-8 on the fly when read by XETEX. This worked but seemed awkward, and when the following year it was LuaTEX's turn to be integrated in TEX Live, we felt this decision needed to be reconsidered.

We decided to adopt the converse strategy of what was originally done for XETEX: convert all the files to UTF-8, and devise a system to convert the patterns back to the appropriate 8-bit encoding if necessary. That way XETEX and LuaTEX could read the files in UTF-8 directly, while pdfTEX and Knuth's TEX would also work because they'll see 8-bit versions of the patterns, converted on the fly. It should be noted that all this happens when generating formats, as — except for LuaTEX — this is the only moment when hyphenation patterns are read by TEX, in its iniTEX incarnation. Once that job is finished and the formats are dumped, each engine will be fed the characters in the encoding appropriate to its kind.

The initial work was done in the spring of 2008 and was completed in time to be included in TEX Live 2008, as was the original intention. We also used this opportunity to rationalise the names of hyphenation patterns, most of which used relatively cryptic two- or three-letter codes to identify languages: after some research, we made the decision to use the standard BCP 47, which to our knowledge is the only one that allows the level of precision we need to distinguish between all the languages TEX supports. BCP stands for "Best Current Practice" and is used for a number of specifications by the IETF, the Internet Engineering Task Force. This standard is thus also used in most Web technologies, and the exact same language tags can be used in HTML and HTTP, for example. Since all BCP specifications are published in the RFC (Request for Comments) series, it's probably useful to mention that BCP 47 is currently equivalent to the combination of RFC 5646 and RFC 4647; these numbers may change in the future, when the document is updated.

The result of this effort was the package hyph-utf8 that is now used in MiKTEX as well. It was soon picked up by external projects: Hyphenator, a JavaScript program for supporting hyphenation in browsers; then Firefox, that implemented hyphenation in the browser itself; and finally Apache FOP (Formatting Objects Processor), an XSL-FO implementation. All these programs took patterns directly from our package, usually with just one straightforward conversion to adapt them to their format.

Since then, we've been keeping track of the updates to the hyphenation patterns in the TEX world; most of the time we're in direct contact with authors, who sent us their contribution directly, but we regularly find isolated updates for some languages. We've also welcomed pTEX in TEX Live in 2010, specialised in typesetting Japanese, for which we had to adapt the pattern loading strategy, since it didn't support UTF-8 input; this meant we had

to give up on the idea of converting patterns on the fly, and thus provided 8-bit versions of all patterns that would be used for pTEX only; the UTF-8-encoded files serve as the master data. And we're constantly trying to clarify the licence terms of the patterns. We feel we have a very good momentum in the TEX community, and the `tex-hyphen` mailing list (`http://lists.tug.org/tex-hyphen`), that's been driving the effort since we started it, has become some sort of town square to discuss many language-related topic in the TEX world, far beyond the subject of hyphenation.

The rest of the free software world, however, is a completely different story. The attentive reader will have noticed that some names are missing from the above list of projects we're collaborating with, and indeed we had little interaction with the developers of the existing free word processors. Some conversations took place, to be sure, but there was no concerted effort to collect all patterns in a central place, or decide what list of licences was acceptable for the different projects.

At the time OpenOffice was the most widespread of the word processors from the free software world, and pattern sets had already been adapted for its use, starting some time before hyph-utf8 was created. The conversion was usually done for one file at a time, with no coordination between the languages, much like in the past individual pattern sets were uploaded on a one-off basis to CTAN. On occasion patterns were created for new languages, which we took over when we became aware of it. There was also a lot of talk about the licences of different pattern files, and some changes came back to us because of that. By then all major free software licences were used by pattern files: GPL, LGPL, n-clause BSD for different values of n, MIT, LPPL of course, and some free-form text. The expansion of these acronyms is left as an exercise to the reader (but read on for a partial cheatsheet). In addition, some people were apparently asked to sign a contributor licence agreement (CLA), that is, an express agreement between the authors and the organisation responsible for OpenOffice. We are still unclear as to why it was so, but to our knowledge this hasn't been the case since the Apache Software Foundation took over the maintenance of OpenOffice in 2011.

Not much happened on this front for a few years, until it was time for Google to join the party. In September 2015, and then again in December of that year, many pattern authors were contacted with requests to once again change the licence of the files, with little explanation of why they were asked to do so. It took us some research to understand what was happening: hyphenation had been added to the operating system Android, and Google was thus interested in using the hyphenation patterns available for TEX and other free software, but they had some restrictions relating to licences. As it turned out, the LPPL was the one that caused most problems for them. There was also a secondary suggestion to add the hyphenation patterns to Unicode's Common Locale Database Repository (CLDR), a large project collecting linguistic data for many languages; there were also legal obstacles to that at the time.

There followed several weeks of extensive discussion, hardly interrupted by Christmas and New Year's Eve, between pattern authors, ourselves, and representatives of Google, soon joined by developers from Mozilla (for Firefox), LibreOffice — by now the most active free software office suite — and even Amazon (for software running on Kindle), over the course of which many, many emails were exchanged and went several times round the planet. Conversations that had started among maintainers of patterns for some language inflated to include more and more contributors, spilled over unto mailing lists, took a side road to discuss the respective merits of different licences, turned around to question the motivation of the requesters, deflated back again to wonder what the exact wording of a licence statement should be, and finally died down when absolutely, absolutely everything had been said and pondered, even that which should probably never, ever have been said nor pondered. When the dust finally settled in the cyberspace and the protagonists were recovering from what can only be described, in the words of one of the authors of this article, as "a huge wave of ???", we came to a decision, and we now have a plan, that will be developed in the next section. It also transpired a few months later that in order to include the patterns in the CLDR, it would be necessary for each contributor to sign a CLA, which will probably be very close to the individual CLA for contributors to Android; however, the exact text is not finalised yet.

2 The past few months

We need to say a few words about the LPPL, since it did as mentioned cause the most amount of talk. It is a licence characterised by two main conditions: first, that any work derived from a work under the LPPL identify itself as such, and second, that each work come with one particular person known as the maintainer, responsible for keeping it up-to-date. Both these conditions represent specific challenges, which we'll attempt to explain.

The first one, specified in clause 6.1 of the current version of the LPPL (1.3c, dating from 2008),

is well-known to TEX users as it is equivalent to the condition under which Donald Knuth made TEX available: that any modified instance of the program that didn't fulfil a strict series of tests be given a different name. Earlier versions of the LPPL actually used a wording much closer to Knuth's (it changed with LPPL version 1.3 in 2003). This clause, however, is virtually unheard of outside the TEX community, and whenever external projects want to use hyphenation patterns that are placed under the LPPL, we need to do some education about the terms of the licence (who does that does not matter as long as a conversation is had, but in practice it often boils down to the two authors of this article). This may turn out harder than it looks, as we've experienced some resistance to that notion, that sometimes gets questioned or even ignored; we did for example have a discussion with a lawyer from a technological company who found the wording of the LPPL ambiguous and stated that it "imposes a lot of confused definitions of derivative works". This project rejected the LPPL based on its text alone. Even without such a strong reaction, the LPPL is generally frowned upon and pattern authors are thus often asked to make their files available under a different licence (to "relicense" them), the exact one depending on the project.

The reasons for third-party developers to request another licence are not only psychological: the identification condition of the LPPL, while seemingly simple to comply with, actually makes it incompatible with many licences. Roughly speaking, that's all the copyleft licences — those characterised by the requirement that any modification of the work they apply to be made available under the same conditions — such as the GNU General Public Licence (GPL) and Lesser General Public Licence (LGPL), or the Mozilla Public Licence (MPL). In these cases the patterns have to be relicensed under the copyleft licence, or a licence compatible with it, in order for the external project to be allowed to use them; it is not a matter of taste. This is one main reason for the multiplicity of licences, the other chief one being authors' personal preferences, although often they don't have strong opinions. Attempts to work around that incompatibility are known to not work; in at least one case we are aware of a project trying to incorporate patterns under the LPPL into copylefted code (by documenting the situation and giving proper credit to the original authors, of course), but we now understand this to be a violation of the LPPL.

For external projects, this is not such a problem in practice, however, as authors generally develop their patterns in the spirit of collaboration and open access, and thus readily agree to make their patterns available under any alternative licence when asked. The one issue is logistical: it can sometimes be hard to contact some authors, and there are many external projects trying to have the patterns relicensed, which leads to the only case of reluctance we've experienced: on occasion an author will display a clear (and understandable) expression of frustration at being asked the same question over and over again — a situation named "relicensing fatigue" by a developer used to being on the other end of these conversations. It should be noted that in this situation the authors of this article have thus far been only passive, witnessing discussions between pattern authors and third-party projects.

The other chief condition of the LPPL is that each work placed under it should have a designated maintainer, the one person allowed to make changes without needing to change the identification of the work as per clause 6.1. By default it is the original author of the work, who may nominate another person when they are no longer willing or able to look after the work themselves.

The issue we're having with that condition is that we don't understand who is the maintainer of the pattern files in hyph-utf8: in our case, we are not the original authors of the patterns and we have not been appointed by them as maintainers (except in a few minor cases), but we are definitely the point people responsible for changes in hyph-utf8, whose core is a set of pattern files with new names. Who are thus the maintainers of the individual files? If it is the pattern authors, this would be a statement of fact that isn't true: the original authors are not the people allowed to make arbitrary changes to the file in hyph-utf8; we are. If we are the maintainers, this would — formally — deny the authors any say in their patterns (as they are packaged in major distributions). This is a serious problem in the application of the text of the LPPL.

In practice this doesn't actually make any difference: the pattern authors communicate with us in a number of ways, and since our job is only to package the files in a format amenable to TEX distributions, we usually adopt any change to the actual patterns straightaway; and on the other side, we collaborate with the core developers of said distributions to keep our package up-to-date with the latest requirements, such as for example when we had to devise a new encoding strategy for including the patterns in pTEX formats, or when a bug in XETEX was revealed by the patterns at format-generation time in the development of TEX Live 2016. This is in our opinion the way it should be done: we volunteer our time to work

on the low-level support of the patterns, and the authors volunteer theirs for the linguistic aspect; there doesn't need to be a formal recognition of these roles.

The notion of a maintainer thus brings no practical benefit while introducing theoretical problems that come to light during the post-apocalyptic discussions mentioned above. Two examples will serve to illustrate this fact: in one case, the author of a set of patterns under the LPPL had been asked to relicense their patterns. They seemed relatively willing to do that provided the relicensed files would only be used outside the TEX community, but showed some attachment to the idea that the original files should stay under the LPPL. They then finally stated "I will change the licence if and only if the TEX-hyphen working group allows me to" (meaning us). This is heartbreaking to us: we are not in the business of prescribing to volunteers how they should make their work available to the world; this should be entirely their choice. We are able to make recommendations, of course, but we have refrained as much as possible from interfering in other people's decisions. Since, however, our input was clearly called for at that point, we of course gave our "authorisation", without feeling entitled to do so.

The other situation was even stranger: in that case the original author had appointed a maintainer as they no longer felt in a position to look after the patterns. The maintainer was keeping a strict policy of no modification to the patterns and thus only added a few lines of comment to the file when they took over maintenance. This was before we started hyph-utf8, and when we did, we simply adopted the file with minor modifications. However, during the discussion about relicensing a few months ago, the maintainer, who had not contributed to the actual patterns at all, felt empowered to decide the terms of the new licence on behalf of the original author (who admittedly seemed a little confused by the situation), and then proceeded to dictate the exact comments we were to put in the file, and even the name of the file itself. This is of course not acceptable to us and we did our best to ignore the maintainer's whims.

These examples are cause for concern, because they show a certain amount of misunderstanding around the core conditions of the LPPL, on many authors' part (the two cases above are by far not the only ones); they also are a clear waste of everybody's time, caused solely by use of the LPPL.

It is thus clear to us that the LPPL is a poor choice for a project such as ours and we are going to change our policy and start recommending against it. We recognise its goals and want to achieve them too, which can't be through formal definitions of different roles such as maintainer. In our case, this has no practical effect and generates too much confusion, as can be seen from the examples above. The correct way to proceed, in our opinion, is by fostering healthy discussions among all the parties involved, from package writers, to distributors, to end users. The situation for hyph-utf8 is of course special since the authors of this article are in some way an intermediary between package writers and distribution developers, but that only strengthens the need for close collaboration.

It has been suggested that since we're effectively responsible for all the patterns in TEX distributions it was also our role to defend their status under the LPPL and enforce it in other projects, if necessary. We don't agree. We're looking after hyphenation patterns and the licence shouldn't matter. What's more, the LPPL doesn't come with any procedure to enforce it, or any compliance team that could help track down violations and rectify them — contrary to other licences such as the GPL and the LGPL, where one can assign copyright to the Free Software Foundation (FSF), that also provides guidelines on how to investigate and remove suspected violations. In any case, that's a discussion we'd (much) rather not be part of, even if we're sometimes co-opted into it.

For that matter, we don't even have guidelines on what the desired level of stability for the hyphenation patterns is. There is no agreement on what type of modifications are acceptable, and depending on the language the authors may develop them actively, or not at all. We follow the changes while trying to be conservative for long-established pattern files (this is relatively easy to do since files of a certain age generally get few updates); but there is nothing to prevent someone from one day sending mass changes to decades-old patterns for a major language, and a difficult conversation will need to take place at that point: will we ignore the changes, or create a new file for the same language, or . . . ? We are willing to follow any reasonable policy that is agreed upon, but we can't make it ourselves: that clearly has to be done by the wider community.

In short, we want to ensure the best possible future for the patterns and this goes through collaboration with all the projects that are interested in developing them. If we kept strictly to the LPPL, this would create an artificial divide among the languages, separating those that can be shared with other projects and those that can't; inevitably the latter would tend to get less attention and become second-class citizens. We had already noticed that among the languages whose patterns had moved away

from the LPPL were some of the world's major languages and we wanted to look into this fact in more detail. We thus ordered them by decreasing number of speakers; having grouped together all language variants together, we had exactly 60 languages. From this we took the top third, to see how many were still under the LPPL, and the result was striking: only 2 out of 20. And even then, the status of these two was quite artificial, since one had seen its licence being chosen arbitrarily as the "popular" choice, and the other one actually comes with several variant files, some of which are not under the LPPL; the one that ships in hyph-utf8 is, but external projects could just as well take the ones whose licences are more acceptable to them. One can of course argue about the methodology and the difficulty of determining the number of speakers for each language, but this figure alone clearly points to the fact that the divide is already there. It is very likely this had been brought about by the fact that major languages naturally got more attention and that pattern authors have thus been asked to relicense more often. The "less important" languages are thus at risk of seeing their patterns progressively lose ground.

In conclusion, we think that in spite of all the difficulties, we are at a point where we're finally able to put TEX at the centre of all efforts to create good hyphenation patterns in the free software world. Let's use that opportunity and not close ourselves to change; we want to become proactive in that effort and we now have a plan: we are going to start recommending to pattern authors to switch to the MIT licence, a very permissive one that has a simple text; other permissive licences would of course be acceptable, if that's the author's preference. We also want to start talking about hyphenation proper instead of politico-legal issues. We accept that the latter is inevitable but we've had far too much of it lately, as the disproportionate structure of this article makes clear.

Back to the work in progress: we have made a few updates for TEX Live 2016. Apart from some licence business (as usual ...), it has a few low-level changes: we've broken up the main package into different language groups to reflect the many packages called hyphen-⟨*language*⟩; the patterns used to all be in the former package while the latter were shells merely containing instructions on how to populate `language.dat`. We have also rewritten the top of each pattern file to make the comments machine-readable, in an effort to make the many language files easier to process; and we've renamed the plain text versions of the patterns to more human-friendly names. Finally, we have migrated the repository to GitHub for better visibility. These changes, however modest, should be seen as preparatory work for making our package more palatable to external projects.

In a separate effort, we also started looking into `patgen` and, in an endeavour to understand in detail how it worked — and to support UTF-8 input — we rewrote it in Ruby using object-oriented programming. As a result, it is hopelessly slow, by a factor of about 60, but we have been able to reproduce the results exactly, and we are going to look into making it more efficient, and hopefully enhance it. The code is currently available at `https://github.com/hyphenation/hydra` and will be distributed through CTAN and TEX Live in due time.

3 The next few months

The future plans right now include:

- Setting up a new website for the project at `http://www.hyphenation.org`.
- Getting as many pattern authors as possible to agree to the MIT licence and Unicode's CLA.
- Unify all the patterns from different sources and finally become the central hub for all things hyphenation in the free software world.

4 The future

Some more distant plans involve looking at the hyphenation algorithm in more detail: the hyphenation library used by the free word processors, `libhyphen`, actually does a little more than TEX's original algorithm, which is not surprising since the current version dates from the mid-2000s, and it would certainly be nice to see if the additional features couldn't be included into TEX engines too. There's always more to do!

◇ Mojca Miklavec
Sežana, Slovenia

◇ Arthur Reutenauer
late of the Royal Opera House,
Covent Garden
`arthur.reutenauer (at)`
`normalesup dot org`