Hyphenation on Demand

Petr Sojka
Faculty of Informatics
Masaryk University Brno
Botanická 68a, 602 00 Brno
Czech Republic
sojka@informatics.muni.cz

Abstract

The need to fully automate the batch typesetting process increases with the use of TEX as the engine for high-volume and on-the-fly typeset documents which, in turn, leads to the need for programmable hyphenation and line-breaking of the highest quality.

An overview of approaches for building *custom* hyphenation patterns is provided, along with examples. A methodology of the process is given, combining different approaches: one based on morphology and hand-made patterns, and one based on word lists and the program PATGEN. The method aims at modular, easily maintainable, efficient, and portable hyphenation. The bag of tricks used in the process to develop custom hyphenation is described.

Motivation

> *In principle, whether to hyphenate or not is a style question and CSS [Cascading Style Sheets] should develop properties to control hyphenation. In practice, however, for most languages there is no algorithm or dictionary that gives all (and only) correct word breaks, so some help from the author may occasionally be needed.*
> — (Bos, 1999)

Separation of content and presentation in today's open information managment style in the sense of SGML/XML (Goldfarb, 1990; Megginson, 1998) is a challenge for TEX as a batch typesetting tool. The attempts to bring TEX's engine to untangle presentation problems in the WWW arena are numerous (Sutor and Díaz, 1998; Skoupý, 1998).

One bottleneck in the high-volume quality publishing is the proofreading stage — line-breaking and hyphenation handling that need to be fine tuned to the layout of particular publication. Tight deadlines in paper-based document production and high-volume electronic publishing put additional demands for better automation of the typesetting process. The need for multiple presentations of the same data (e.g., for paper and screen) adds another dimension to the problem. Problems with hyphenation are often one of the most difficult. As most TEX users are perfectionists, fixing and tuning hyphenation for every presentation is a tedious, time-consuming task.

We have already dealt with several issues related to hyphenation in TEX (Sojka and Ševeček, 1995; Sojka, 1995). On the basis of our being involved in typesetting tens of thousands of TEX pages of multilingual documents (mostly dictionaries), we want to point out several methods suitable for the development of hyphenation patterns.

Pattern generation

> *There is no place in the world that is linguistically homogeneous, despite the claims of the nationalists around the world.*
> — (Plaice, 1998)

Liang (1983), in his thesis written under Knuth's supervision, developed a general method to solve the hyphenation problem that was adopted in TEX82 (Knuth, 1986a, App. H). He wrote the PATGEN program (Liang and Breitenlohner, 1999), which takes

- a list of already hyphenated words (if any),
- a set of patterns (if any) that describes "rules",
- a list of parameters for the pattern generation process,
- a character code translation file (added in PATGEN 2.1; for details see Haralambous (1994),

and generates

- an enriched set of patterns that "covers" all hyphenation points in the given input word list,
- a word list hyphenated with the enriched set of patterns (optional).

The patterns are loaded into TEX's memory and stored in a data structure (cf. Knuth, 1986b, parts 40 – 43), which is also efficient for retrieval — a variant of *trie* memory (cf. Knuth, 1998, pp. 492 – 512). This data structure allows hyphenation pattern searching in linear time with respect to the pattern length. The algorithm using a trie that "outputs" possible hyphenation positions may be viewed as finite automaton with output (Mealy automaton or transducer).

Pattern development

> *. . . problems [with hyphenation] have more or less disappeared, and I've learnt that this is only because, nowadays,* every *hyphenation in the newspaper is* manually checked *by human proof-readers.*
> — (Jarnefors, 1995)

Studying patterns that are available for various languages shows that PATGEN has only been used for about half of the hyphenation pattern files on CTAN (cf. Table 1 in Sojka and Ševeček, 1995).

There are two approaches to hyphenation pattern development, depending on user preferences. Single authors using TEX as an authoring tool want to minimize system changes and want TEX to behave as a fixed point so that re-typesetting of old articles is easily done, thanks to backwards compatibility. For such users, one set of patterns that is fixed once and for all might be sufficient.

On the other hand, for publishers and corporate users with high-volume output, it is more efficient to make a long-term investment into development of hyphenation patterns for particular purposes. I remember one TEX user saying that my suggestion to enhance standard hyphenation patterns with custom-made ones to allow better hyphenation of chemical formulæ would save his employer thousands of pounds per year. Of course, with this approach, one has to archive *full* sources for every publication, together with hyphenation patterns and exceptions.

One of the possible reasons PATGEN has not been used more extensively may be the high investment needed to create hyphenated lists of words, or better, a morphological database of a given language.

Pattern bootstrapping and iterative development

> *The road to wisdom?*
> *Well it's plain and simple to express:*
> *Err and err and err again*
> *but less and less and less.*
> — (Hein, 1966)

When developing new patterns, it is good to start with the following bootstrapping technique with iteration, which should avoid the tedious task of manually marking hyphenation points in huge lists of words:

1. Write down the most obvious initial patterns, if any, and/or collect "the closest" ones (e.g., consonant-vowel rules).
2. Extract a small word list for the given language.
3. Hyphenate current word list with current set patterns.
4. Check all hyphenated words and correct them; in the case of errors return to step 3.
5. Collect a bigger word list.
6. Use the previously generated set of patterns to hyphenate the words in this bigger list.
7. Check hyphenated words, and if there are no errors, move to step 9.
8. Correct word list and return to step 6.
9. Generate final patterns with PATGEN with parameters fitted for the particular purpose (tuned for space or efficiency).
10. Merge/combine new patterns with other modules of patterns to fit the particular publishing project.

To find an initial set of patterns, some basic rules of hyphenation in the specific language should be known. Language can be grouped into one of two categories: those that derive hyphenation points according to etymology and those that derive hyphenation according to pronunciation — "syllable-based" hyphenation. For the first group of languages, one should start with patterns for most frequent endings and suffixes and prefixes. For syllable-based hyphenation, patterns based on sequences of consonants and vowels might be used (cf. Chicago Manual of Style, 1993, Section 6.44, and Haralambous, 1999) as first approximation of hyphenation patterns.

As using TEX itself for hyphenation of word lists and development of patterns may be preferred to other possibilities, we will start with this portable solution, using hyphenation of phonetic transcriptions as an example of a syllable-based "language".

Let's start with some plain TEX code to define consonant-vowel (CV) patterns:

```
% ... loading plain.tex
%     without hyphen.tex patterns ...
\patterns{cv1cv cv2c1c ccv1c cccv1c
ccccv1c cccccv1c v2v1 v2v2v1 v2v2v2v1
...
}
```

There is a way to typeset words together with their hyphenation points in TEX; the code from Olšák (1997, with minor modifications) looks like this:

```
\def\showhyphenspar{\begingroup
 \overfullrule=0pt \parindent0pt
 \hbadness=10000 \tt
 \def\par{\setparams\endgraf\composelines}%
 \setbox0=\vbox\bgroup
            \noindent\hskip0pt\relax}

\def\setparams{\leftskip=0pt
 \rightskip=0pt plus 1fil
 \linepenalty1000 \pretolerance=-1
 \hyphenpenalty=-10000}

\def\composelines{%
 \global\setbox1=\hbox{}%
 \loop
   \setbox0=\lastbox \unskip \unpenalty
   \ifhbox0 %
     \global\setbox1=\hbox{%
       \unhbox0\unskip\hskip0pt\unhbox1}%
 \repeat
 \egroup % close \setbox0=\vbox
 \exhyphenpenalty=10000%
 \emergencystretch=4em%
 \unhbox1\endgraf
 \endgroup}
```

Now, we will typeset our word list in the typewriter font without ligatures. To use the CV patterns defined above we need to map word characters properly:

```
% vowels mapping
\lccode`\a=`v  \lccode`\e=`v
\lccode`\i=`v  \lccode`\o=`v
...
% consonants
\lccode`\b=`c  \lccode`\c=`c
\lccode`\d=`c  \lccode`\f=`c
...
\raggedbottom \nopagenumbers
\showhyphenspar
The need to fully automate the
batch typesetting process increases
with the use of word in wordlist
...
\par\bye
```

Finally, extracting hyphenated words from `dvi` the file via the `dvitype` program, we get our word list hyphenated by our simple CV patterns.

Another way to get the initial word list hyphenated is to use `PATGEN` with initial patterns and no new level, letting `PATGEN` hyphenate the word list that was input. PERL addicts may want to use the PERL hyphenation module (Pazdziora, 1997) for the task.

Once the job of proofreading the word list is finished, we can generate new patterns and collect other words in the language. Using new patterns on the new collection will show the efficiency of the process.

Fine tuning of patterns may be iterated, once `PATGEN` parameters are set, so that nearly 100 % coverage of hyphenation points is achieved in every iteration. The setting of such `PATGEN` parameters may be difficult to find on the first attempt. Setting of these parameters is discussed in Sojka and Ševeček (1995).

Modularity of patterns

It is tractable for some languages to create patterns by hand, simply by writing patterns according to the rules for a given language. This approach is, however, doomed to failure for complex languages with several levels of exceptions. Nevertheless, there are special cases in which we may build pattern modules and concatenate patterns to achieve special purpose behaviour. This applies especially when additional characters (not handled when patterns have been built originally) may occur in words that we still want to hyphenate.

Patterns generated by Raichle (1997) may serve as an example that can be used with any fonts in the standard LATEX eight-bit T1 font encoding, to allow hyphenation after an explicit hyphen. Similar pattern modules can be written for words or chemical formulæ that contain braces and parentheses. These can be combined with "standard" patterns in the needed encodings. Some problems might be caused by the fact that TEX does not allow metrics to be defined for `\lefthyphenmin` and `\righthyphenmin` properly — we might want to say that ligatures, for instance, count as a single letter only or that some characters should not affect hyphenation at all (e.g. parentheses in words like colo(u)r). We must wait until some naming mechanisms for output glyphs (characters) is adopted by the TEX community for handling these issues.

Adding a new primitive for the hyphenmin code — let's call it `\hccode`, a calque on `\lccode` — would cause similar problems: changing it in mid-paragraph would have unpredictable results.[1]

It is advisable to create modules or libraries of special-purpose hyphenation patterns, such as the

[1] ε-TEX v2 has a new feature to fix the `\lccode` values during the pattern read phase.

ones mentioned above, to ease the task of pattern development. These patterns might be written in such as as to be easily adaptable for use with core patterns of a different language.

Common patterns for more languages

Having large hyphenated word lists of several languages the possibility then exists to make multilingual or special-purpose patterns from collections of words by using `PATGEN`. Joining word lists and generating patterns on demand for particular publications is especially useful when the word databases are structured and split into sublists of personal names, geographic names, abbreviations, etc. These patterns are requested when typesetting material in which language switching is not properly done (e.g. on the WWW).

Czech and Slovak are very closely related languages. Although they do not share exactly the same alphabet, rules for hyphenation are similar. That has led us to the idea of making one set of hyphenation patterns to work for both languages, saving on space in a format file that supports both. In the Czech/Slovak standard TEX distribution there is support for different font encodings. For every encoding, hyphenation patterns have to be loaded as there is no character remapping on the level of trie possible. Such Czechoslovak patterns would save patterns for each encoding in use.

It should be mentioned that this approach cannot be taken for any set of languages as there may be, in general, identical words that hyphenate differently in different languages; thus, simply merging word lists to feed `PATGEN` is not sufficient without degrading the performance of patterns by forbidding hyphenation in these conflicting words (e.g. re-cord vs. rec-ord).

Phonetic hyphenation

As an example of custom-made hyphenation patterns, the patterns required to hyphenate a phonetic (IPA) transcription are described in this section. Dictionaries use this extensively — see Fig. 1,[2] taken from Kirsteinová.

The steps used to develop the hyphenation patterns for this dictionary were similar to those described in the previous section on bootstrapping:

1. Write down the most obvious (syllable) patterns.
2. Extract all phonetic words from available texts.
3. Hyphenate this word list with the initial set of patterns.
4. Check and correct all hyphenated words.
5. Generate final quality patterns.

In bigger publishing projects efforts like this pay off very quickly.

akkompagnement *sb* [ækʌmpænjə-ˈmaŋ] *-et, -er* hudební doprovod *m*
alimentationsbidrag *sb* [ælimɛntæ-ˈšoːˈnsˌbiˌdraːˈw] *-et,* - alimenty *pl,* příspěvek *m* na výživné dítěte
befolknings‖eksplosion *sb* [beˈfʌlˈg-neŋs-] *-en, -er* populační exploze *f* ■ **-tilvækst** *-en, -er* přírůstek *m* obyvatelstva ■ **-tæthed** *-en, -er* hustota *f* obyvatelstva
bemærkelsesværdig *adj* [beˈmæʀ-gəlsəsˌvæʀˈdi] *-t, -e* pozoruhodný
beslutningsdygtig *adj* [beˈslud-neŋsˌdøgdi] *-t, -e* schopný rozhodovat; **den lovgivende forsamling var ~** zákonodárné shromáždění bylo schopné se usnášet

Figure 1: Example of phonetic hyphenation in Kirsteinová and Borg (1999).

Hyphenation for an etymological dictionary

In some publications (Rejzek, in prep., for example), a different problem can arise: the possibility of having more than 256 characters used within a single paragraph. This problem cannot, in general, be easily solved[3] within the frame of TEX82. We thus tried Ω, the typesetting system by Plaice and Haralambous, for this purpose. One has to create special virtual fonts (e.g., by using the fontinst package) on top of the Ω ones, in order to typeset it — see Fig. 2.

More hyphenation classes

> *But at least I can point out a minor weakness of TEX's algorithm: all possible hyphenations have the same penalty. This might be ok for english, but for languages like German that have a lot of composite words there should be the ability to assign lower penalties between parts of a composite i.e. Um-brechen should be favored against Umbre-chen.*
> — (Hars, 1999)

[2] The IPA font used is TechPhonetic, downloadable from `http://www.sil.org/ftp/PUB/SOFTWARE/WIN/FONTS/`.

[3] One could try to re-encode all fonts used in parallel in some paragraph such that they share the same `\lccode` mappings, but this exercise would have to be made for each multilingual-intensive publication, again and again.

naivní ‘prostoduchý, dětinský’, *naivita, naivka.* Přes něm. *naiv* z fr. *nai:f* tv., původně ‘přirozený, opravdový’ z lat. *nātīvus* ‘přirozený’ od *nātus,* což je příč. trp. od *nāscī* ‘rodit se’. Srov. ↑*nacionále.*

náruživý ‘vášnivý, silně zaujatý’, *náružívost.* Jen č. Souvisí s č.st. *oruží* ‘zbroj, zbraň, náčiní’ (všesl.). Psl. kořen **-rǫg- (B1, B7)* nejspíš souvisí s lit. *įrangùs* ‘prudký’ (HK), *rángtis* ‘spěchat’, *ren~gtis* ‘chystat se’, *rangà* ‘přístroj, nástroj’, dále asi se střhn. *ranc* ‘rychlý, vířivý pohyb’, něm. *renken* ‘pohybovat se krouživě sem tam’, angl. *wrench* ‘trhnout, vykroutit’, vše by to bylo od ie. **u̯reng-* ‘kroutit, ohýbat’ a vzdáleně příbuzné by bylo ↓*vrhat.*

nebozez ‘vrták na dřevo’, *nebozízek.* Hl. *njeboz,* sln. *nabôzec.* Přejato z germ., forma by ukazovala až na germ. **naba-gaiza* před změnou *-z->-r- (A5, B1),* která už je provedena v sthn. *nabagēr* (něm. *Näber* tv.). První část germ. slova odpovídá něm. *Nabe* (viz ↑*náboj*), druhá něm.st. *Ger* z gót. **gaiza-* ‘něco špičatého’.

Figure 2: Using Ω to typeset paragraphs in which words from languages with more than 256 different characters may appear and be hyphenated in parallel.

Some suggestions on handling multiple hyphenation classes were suggested in Sojka (1995). A prototype implementation of ε-TEX and `PATGEN` has recently been done (Classen, 1998). For wider adoption of such improvements availability of large word lists and development of new patterns is crucial. Many of the methods mentioned above could be used to develop such multi-class/multi-purpose patterns. Allen (1990) contains such a word list, which shows that some publishers do pay attention to linebreaking details.

Speed considerations

Even though hyphenation searches using a trie data structure is fast, searching for unnecessary hyphenation points is a waste of time. It is advisable to tell TEX where words shouldn't be hyphenated. Comparing several possibilities for suppressing hyphenation, the option of setting `\lefthyphenmin` to 65 is slightly faster than switching to `\language`, which has no patterns. These solutions outperform the `\hyphenpenalty 10000` solution by a fair amount (cf. Arsenau, 1994).

Reuse of patterns

Sometimes we need the same patterns with different `\lefthyphenmin` and `\righthyphenmin` parameters. The suggested approach is not to limit hyphens close to word boundaries during the pattern generation phase but to use TEX's `\setlanguage` primitive. This can be done to achieve special hyphenation handling for the last word in a paragraph (e.g., a higher `\righthyphenmin`) given proper markup by a preprocessing filter. For example:

```
\newcount\tmpcount
\def\lastwordinpar#1{%
\tmpcount=\righthyphenmin
\righthyphenmin5
\setlanguage\language #1
\expandafter\righthyphenmin\the\tmpcount
\setlanguage\language}

\showhyphens{demand}
\lastwordinpar{demand\showhyphens{demand}}
\bye
```

Future work

> *If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.*
> — (Knuth, 1989)

It seems inevitable that embedding of language-specific support modules will be necessary for *the* typesetting system in the future. These demands may not only apply for hyphenation but also for spelling or even grammar checkers. As even people using WYSIWYG systems may use tools that help to visualise possible typos (in color, etc.) on the fly, the computing power of today's machines is surely sufficient to do the same in batch processing with even better results.

The idea of using patterns to capture mappings specific for particular languages or dialect modules can be further generalized for different purposes and mappings. The use of the theory of finite-state transducers (Mohri, 1996; Mohri, 1997; Roche and Schabes, 1996) to implement other classes of language modules looks promising.

Summary

Some computerized typesetting methods in frequent use today may render a conservative approach to word division impractical. Compromise may therefore be necessary pending the development of more sophisticated technology.
— Chicago Manual of Style (1993, Section 6.43)

We have outlined some of the possibilities offered by TEX and PATGEN for the development of customized hyphenation patterns. We have suggested bootstrapping and iterative techniques to facilitate pattern development. We also suggest wider employment of PATGEN and preparation of hyphenated word lists and modules of patterns for easy preparation of hyphenation patterns on demand in today's age of digital typography (Knuth, 1999).

Acknowledgements. We thank Bernd Raichle for valuable comments and corrections to the paper. We are indebted to the Proceedings editor for wording improvements. The presentation of this work has been made possible through support from the Ministery of Education, Youth and Physical Training (MŠMT ČR grant VS97028).

References

Allen, R.E. *The Oxford Spelling Dictionary*, volume II of *The Oxford Library of English Usage*. Oxford University Press, 1990.

Arsenau, Donald. "Benchmarking paragraphs without hyphenation". Posting to the Usenet group news:comp.text.tex on Dec 13, 1994.

Bos, Bert. "Internationalization / Localization". http://www.w3.org/International/O-HTML-hyphenation.html, 1999.

Chicago Manual of Style. *The Chicago Manual of Style*, 14th edition, 1993.

Classen, Matthias. "An extension of TEX's hyphenation algorithm". ftp://peano.mathematik.uni-freiburg.de/pub/etex/hyphenation/, 1998.

Goldfarb, Charles F. *The SGML Handbook*. Clarendon Press, Oxford, 1990.

Haralambous, Yannis. "A Small Tutorial on the Multilingual Features of PATGEN2". In electronic form, available from CTAN as info/patgen2.tutorial, 1994.

Haralambous, Yannis. "From Unicode to Typography, A Case Study: The Greek Script". Proceedings of 14th International Unicode Conference, preprint available from http://genepi.louis-jean.com/omega/boston99.pdf, 1999.

Hars, Florian. "Typo-l email discussion list". 1999.

Hein, Piet. *Grooks*. MIT Press, Cambridge, Massachusetts, 1966.

Jarnefors, Olle. "ISO-10646 email discussion list". 1995.

Kirsteinová, Blanka and B. Borg. *Dánsko-český slovník, Dansk-Tjekkisk Ordbog [Danish-Czech dictionary]*. LEDA, Prague, Czech Republic, 1999.

Knuth, Donald E. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986a.

Knuth, Donald E. *TEX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986b.

Knuth, Donald E. "Theory and Practice". Keynote address for the 11th World Computer Congress (Information Processing '89), 1989.

Knuth, Donald E. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, 1998.

Knuth, Donald E. *Digital Typography*. CSLI Lecture Notes 78. Center for the Study of Language and Information, Stanford, California, 1999.

Liang, Frank. *Word Hy-phen-a-tion by Com-put-er*. Ph.D. thesis, Department of Computer Science, Stanford University, 1983.

Liang, Frank and P. Breitenlohner. "PATtern GENeration Program for the TEX82 Hyphenator". Electronic documentation of PATGEN program version 2.3 from web2c distribution on CTAN, 1999.

Megginson, David. *Structuring XML Documents*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1998.

Mohri, Mehryar. "On some applications of finite-state automata theory to natural language processing". *Natural Language Engineering* **2**(1), 61–80, 1996.

Mohri, Mehryar. "Finite-State Transducers in Language and Speech Processing". *Computational Linguistics* **23**(2), 269–311, 1997.

Olšák, Petr. *TEXbook naruby [TEXbook topsy-turvy]*. Konvoj, Brno, 1997.

Pazdziora, Jan. "TeX::Hyphen — hyphenate words using TEX's patterns". CPAN: modules/by-authors/Jan_Pazdziora/TeX-Hyphen-0.10.tar.gz, 1997.

Plaice, John. "pdftex email discussion list". http://www.tug.org/archives/pdftex/msg01913.html, 1998.

Raichle, Bernd. "Hyphenation patterns for words containing explicit hyphens". CTAN/language/hyphenation/hypht1.tex, 1997.

Rejzek, Jan. *Etymologický slovník českého jazyka [Czech Etymological Dictionary]*. LEDA, Prague, Czech Republic, in prep..

Roche, Emmanuel and Y. Schabes. *Finite-State Language Processing*. MIT Press, 1996.

Skoupý, Karel. "$\mathcal{NTS}$: A New Typesetting System". *TUGboat* **18**(3), 318–322, 1998.

Sojka, Petr. "Notes on Compound Word Hyphenation in TEX". *TUGboat* **16**(3), 290–297, 1995.

Sojka, Petr and P. Ševeček. "Hyphenation in TEX — Quo Vadis?". *TUGboat* **16**(3), 280–289, 1995.

Sutor, Robert S. and A. L. Díaz. "IBM techplorer: Scientific Publishing for the Internet". *Cahiers Gutenberg* **28–29**, 295–308, 1998.

The Young Man of Vancouver

There was a young man of Vancouver
who thought he admired Anita Hoover
but he looked at some macros
which ran under Windows
and now all he can think of is `\over`s

—Sebastian Rahtz

The TUG conference

Down the TEXing path we go
with a Sparc its not so slow
Up the network nodes we run
`\href` links can be so much fun
Round the browser wars we dodge
Sans MathML—a real hodge-podge
Home at last—the Web is **fast**—we wait for LATEX3
While Frank and David trade ideas,
Chris seeks terminology

—Christina Thiele

The Young Lady of Stanford

There was a young lady from Stanford
who delighted to play with Mac Word
she met a Don Knuth
who told her the truth
and now what she enjoys is absurd

—Sebastian Rahtz and Patrick Ion