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 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 and Rubinsky, 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 needs 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 typesetting process. The need for multiple presentation of the same data (e.g. for paper and screen) adds another dimension to the problem. Problems with hyphenation are often those of the most difficult ones. As most TEX users are perfectionists, fixing and tuning hyphenation for every presentation is tedious time consuming task.

We dealt with couple of issues related to the hyphenation in TEX (Sojka and Ševeček, 1995; Sojka, 1995). Based on being involved in the 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 in his thesis (Liang, 1983) under Knuth's supervision developed a general method to solve the hyphenation problem that has been adopted in TEX82 (Knuth, 1986b, Appendix H). He wrote the program PATGEN (Liang and Breitenlohner, 1999) that takes

- a set of patterns (if any) that describes "rules", and/or
- a list of already hyphenated words (if any),
- 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

* [This article has not yet been edited. – Ed.]

- 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, 1986a, parts 40 – 43)] also efficient for retrieval — 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 according to the user preferencies. Single author using TEX as an authoring tools want to minimize system changes and wants TEX behave as fixed point so that retypesetting of old articles is easily done due to backward compatibility. For these user preferencies one set of patterns that is fixed once and for life's length might be sufficient.

On the other hand, for publishers/corporate users with high-volume output, it is more efficient to make 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 by custom made ones to allow better hyphenation of chemical formulæ will save his employer thousands of pounds in a year. With this approach, one has of course to archive for every publication *full* sources together with hyphenation patterns and exceptions.

One of the possible reasons why `PATGEN` has not been used more extensively may be high investment to create hyphenated list of words, or better, morphological database of 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.
— *Piet Hein*

When developing new patterns to save boring work of manually marking hyphenation points in huge lists of words, it is good to start by the following bootstrapping technique with iteration:

1. write down the most obvious initial patterns, if any,
2. extract a small word-list for the given language,
3. hyphenate the word-list from step 2 with the initial patterns of step 1 and/or with "the most closest" ones (e.g. consonant-vowel rules),
4. check all hyphenated words and correct them,
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, continue by step 9,
8. correct word-list and continue by step 6,
9. generate final patterns with `PATGEN` with parameters fitted for particular purpose,
10. merge/combine new patterns with other modules of patterns to fit particular publishing project.

To find an initial set of patterns some basic rules of hyphenation in given language should be known. Two basic categories of languages are known — languages that derive hyphenation points according to ethymology of words, and languages that derive hyphenation with respect to syllables.

In the first case, one should start with patterns for most frequent endings and suffixes and prefixes. For languages with "syllable-close" hyphenation patterns based on sequences of consonants and vowels might be used [cf. (Anonymous, 1993; Haralambous, 1999)]. As using TEX itself for hyphenation of word-list and development of patterns may be preferred to other possibilities, we will

start with this portable solution shown on an example of syllable-close" language (e.g. phonetic hyphenation). Let's start with plainTEX code like:

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

There is a way how to typeset word together with hyphenation points in TEX; the code from (Olšák, 1997) with minor modifications look like:

```
\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 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 ofword in wordlist
...
\par
\bye
```

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

Another approach how to get initial word-list hyphenated are possible, too; we may use `PATGEN` with initial patterns and no new level but let `PATGEN` hyphenate input word-list. PERL addicts may use the PERL hyphenation module (Pazdziora, 1997) for the task.

Once task of proofreading word-list is finished, we may generate new patterns and collect another words of 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 achived in every iteration. The setting of such `PATGEN` parameters may be difficult to find on the first trial. Setting of these parameters is discussed in (Sojka and Ševeček, 1995).

Modularity of patterns

It is tracktable 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.

As an example may serve patterns generated by (Raichle, 1997), that can be used with any T1 encoded fonts to allow hyphenation after an explicit hyphen. Similar pattern modules can be written for word or chemical formulas that contain braces and parenthesis. These can be combined with "standard" patterns in the needed encodings. Some problems might be caused by the fact that TEX doesn't allow to define metrics for `\lefthyphenmin` and `\righthyphenmin` properly — we might want to say that e.g. ligature count as a single letter only or that some characters should be counted as zero (inword braces). We need to wait until some naming mechanisms for output glyphs (characters)

will be adopted in TEX community for handling these issues.

Adding new primitive for hyphenmin code — let's say `\hccode` similarly to `\lccode` would cause similar problems: changing it in midst paragraph would have unpredictable results.[1]

It is advised to create modules/library of special purpose hyphenation patterns like the ones mentioned above to ease the task of pattern development. These patterns might be written to be easily adaptable to be used with core patterns for a given language.

Common patterns for more languages

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

Czech and Slovak are very close languages. Although they don't share exactly the same alphabet, rules for hyphenation are similar. That led us to the experiment of making one set of hyphenation patterns that will work for both languages saving on space in 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 every encoding in use.

Preliminary results showed feasibility of this approach without resorting to compromises and we hope to release first version of the patterns at the time of TUG'99 conference.

It should be mentioned that this approach cannot be taken for any set of languages as there may be, in general, same words that hyphenate differently in different languages and thus simply merging word-lists to feed `PATGEN` is not sufficient without degrading the performance of patterns by forbiding hyphenation in these conflicting words..

[1] ε-TEX V2 has a new feature to fix the `\lccode` values during the pattern read phase.

Phonetic hyphenation

As an example of custom made hyphenation patterns may serve patterns to be used to hyphenate a phonetic (IPA) transcription. Publications like dictionaries use it extensively — see Figure 1.

akkompagnement *sb* [ækʌmpænjə-maŋ] *-et, -er* hudební doprovod *m*
alimentationsbidrag *sb* [ælimɛntæ-šo:'nsˌbiˌdra:'w] *-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-gəlsəsˌvæʀ'di] *-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 use of phonetic hyphenation in (Kirsteinová, in preparation).

The steps used to develop these hyphenation patterns for this dictionary were similar to those described in Section **?!?**:

1. write down the most obvious (syllble) 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 a bigger publishing projects efforts like this pay back really soon.

Hyphenation for ethymological dictionary

In some publications — like (Rejzek, in preparation) — one can face another problem: the possible number of different characters that were used in words of languages used in one paragraph exceeded 256. This problem cannot in general be easily solved[2] within the frame of TEX82. We thus tried Ω

[2] One might try hard to reencode all fonts used in parallel in some paragraph such that they share the same `\lccode`

typesetting system (Plaice, 1994; Haralambous and Plaice, 1994; Plaice and Haralambous, 1996; Haralambous, 1996; Haralambous and Plaice, 1998) for this purpose. One had to create special virtual fonts (e.g. by using the fontinst package) upon Ω ones to typeset it — see Fig. 2.

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

náruživý 'vášnivý, silně zaujatý', *náruživost.* 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 stř hn. *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.

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)*

mappings, but this exercise has to be made for every heavily multilingual publication again and again.

Couple of suggestions how to handle multiple hyphenation classes were suggested in (Sojka, 1995). Prototype implementation of ε-TEX and `PATGEN` has recently been implemented (Classen, 1998). For wider adoption of such improvements availability of big word-lists is crucial. Many methods mentioned above might be used for development of these multiclass/multipurpose patterns. Publication of OUP (Allen, 1990) with such word-listshows that some publishers pays attention to these details of linebreaking.

Speed considerations

Even though hyphenation searching 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 to suppress hyphenation, the way by setting `\lefthyphenmin` to 65 is slightly faster than switching to `\language` with no patterns. These solutions outperforms the `\hyphenpenalty 10000` solution by a fair amount [cf. (Arsenau, 1994)].

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 by using TEX's `\setlanguage` primitive. This can be used to achieve special hyphenation handling for the last word in a paragraph (e.g. higher `\righthyphenmin`) given proper markup by a preprocessing filter. The reader may try:

```
\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 linking to language specific support modules will be necessary for The typesetting system in the future. These demands may not

only apply for hyphenation, but for spelling or even grammar checker. As even people using WYSIWYG systems may use tools that help to visualise possible typos (in color etc.) on the fly, computing power of today's computers 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/ mappings. The use of the theory of finite-state transducers (Mohri, 1996; Mohri, 1997; Roche and Schabes, 1996) looks promising to implement another class of language modules.

Summary

We have outlined possibilities offered by TEX and PATGEN for the development of customized hyphenation patterns. We have suggested bootstrapping and iterative techniques to pattern development. We 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).

Acknowledgement

We thank Bernd raichle for valuable comments to the paper. The presentation of this work has been made possible due to the support of 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. 1089

Anonymous. *The Chicago Manual of Style*. University of Chicago Press, fourteenth edition edition, 1993. 1086

Arsenau, Donald. "Benchmarking paragraphs without hyphenation". Posting to the Usenet group news:comp.text.tex on Dec 13, 1994. 1089

Bos, Bert. "Internationalization / Localization". http://www.w3.org/International/O-HTML-hyphenation.html, 1999. 1085

Classen, Matthias. "An extension of TEX's hyphenation algorithm". ftp://peano.mathematik.uni-freiburg.de/pub/etex/hyphenation/, 1998. 1089

Goldfarb, Charles F. and Y. Rubinsky. *The SGML handbook*. Clarendon Press, Oxford, UK, 1990. 1085

Haralambous, Yannis. "A Small Tutorial on the Multilingual Features of PATGEN2". in electronic form, available from CTAN as info/patgen2.tutorial, 1994. 1085

Haralambous, Yannis. "ΩTimes and ΩHelvetica fonts under development: Step One". *TUGboat* **17**(2), 126–146, 1996. 1089

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. 1086

Haralambous, Yannis and J. Plaice. "First applications of Ω: Adobe Poetica, Arabic, Greek, Khmer". *TUGboat* **15**(3), 344–352, 1994. 1089

Haralambous, Yannis and J. Plaice. "The Design and Use of a Multiple-Alphabet Font with Ω". In *Lecture Notes in Computer Science 1375*, edited by Roger D. Hersch and Jacques André and Heather Brown, pages 126–137. Springer-Verlag, 1998. 1089

Hars, Florian. "Typo-l email discussion list". 1999. 1089

Jarnefors, Olle. "ISO-10646 email discussion list". 1995. 1086

Kirsteinová, B. *Dánsko-český slovník (Danish-Czech dictionary)*. LEDA, Prague, Czech Republic, in preparation. 1088

Knuth, Donald E. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986b. 1085

Knuth, Donald E. *TEX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986a. 1086

Knuth, Donald Ervin. "Theory and Practice". keynote address for the 11th World Computer Congress (Information Processing '89), 1989. 1089

Knuth, Donald Ervin. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, 1998. 1086

Knuth, Donald Ervin. *Digital Typography*. CSLI Lecture Notes 78. Center for the Study of Language and Information, Stanford, California, 1999. 1090

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. 1085

Liang, Franklin Mark. *Word Hy-phen-a-tion by Com-put-er*. Ph.D. thesis, Department of Computer Science, Stanford University, 1983. 1085

Megginson, David. *Structuring XML Documents*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1998. 1085

Mohri, Mehryar. "On some applications of finite-state automata theory to natural language processing". *Natural Language Engineering* **2**(1), 61–80, 1996. 1090

Mohri, Mehryar. "Finite-State Transducers in Language and Speech Processing". *Computational Linguistics* **23**(2), 269–311, 1997. 1090

Olšák, Petr. *TEXbook naruby (in Czech)*. Konvoj, Brno, 1997. 1087

Pazdziora, Jan. "TeX::Hyphen — hyphenate words using TEX's patterns". CPAN: modules/by-authors/Jan_Pazdziora/TeX-Hyphen-0.10.tar.gz, 1997. 1087

Plaice, John. "Progress in the Omega project". *TUGboat* **15**(3), 320–324, 1994. 1089

Plaice, John. "pdftex email discussion list". http://www.tug.org/archives/pdftex/msg01913.html, 1998. 1085

Plaice, John and Y. Haralambous. "The latest developments in Ω". *TUGboat* **17**(2), 181–183, 1996. 1089

Raichle, Bernd. "Hyphenation patterns for words containing explicit hyphens". CTAN/language/hyphenation/hypht1.tex, 1997. 1087

Rejzek, Jan. *Etymologický slovník českého jazyka (Ethymological dictionary of Czech language)*. LEDA, Prague, Czech Republic, in preparation. 1088

Roche, Emmanuel and Y. Schabes. *Finite-State Language Processing*. The MIT Press, 1996. 1090

Skoupý, Karel. "$\mathcal{NTS}$: a New Typesetting System". *TUGboat* **18**(3), 318–322, 1998. 1085

Sojka, Petr. "Notes on Compound Word Hyphenation in TEX". *TUGboat* **16**(3), 290–297, 1995. 1085, 1089

Sojka, Petr and P. Ševeček. "Hyphenation in TEX — Quo Vadis?". *TUGboat* **16**(3), 280–289, 1995. 1085, 1086, 1087

Sutor, Robert S. and A. L. Díaz. "IBM techplorer: Scientific Publishing for the Internet". *Cahiers Gutenberg* **28–29**, 295–308, 1998. 1085