

# Hyphenation

A tutorial for T<sub>E</sub>X users

Petr Sojka

 & Masaryk University, Faculty of Informatics

Brno, Czech Republic

April 30, 2002



# Overview of the Tutorial

- ① Overview, Motivation
- ② Basic Notions, TeX Commands for Hyphenation
- ③ Tips and Tricks, Exercises
- ④ Competing Patterns for Hyphenation Algorithm
- ⑤ Pattern Generation with PATGEN and OPATGEN
- ⑥ Pattern Development: Stratified Sampling and Bootstrapping Techniques
- ⑦ Fine-Tuning and Optimization of Patterns
- ⑧ Tips and Tricks, Exercises for Pattern Generation
- ⑨ Summary, Further Reading

## Motivation

“It is better to break a word with a hyphen than to stretch interword spaces too much. Therefore T<sub>E</sub>X tries to divide words into syllables when there’s no good alternative available.” [Appendix H of TeXbook]

- Hyphenation used since Gutenberg (uniform greyness is of higher priority).  
[Slide]
- Hard problem: hy-phen-a-tion vs. con-cat-e-na-tion (similar but different),  
rec-ord (noun) vs. re-cord (verb), Wachs-tube vs. Wach-stube,  
dem-on-stra-tion vs. de-mon-stra-tive (two letters affect hyphens nine  
positions away).
- Still actual problems, especially in languages with many compound words.  
,,. . . 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)

- High quality needed for high-volume automatic (grrr!) typesetting (of XML).

## Basics

- Up to three paragraph passes in T<sub>E</sub>X.
- First pass (when `\pretolerance` is not negative) without hyphenation.
- Second pass starts with hyphenation of all words in a paragraph.
- When `\tolerance` has not been met and `\emergencystretch` `dimen` is not void, its value is added to the interword stretchability.  
Hyphenation is allowed in the third pass.

## **T<sub>E</sub>X primitives that affect hyphenation**

`\language` is an integer which gives the sequence number of the hyphenation pattern table to use then. When the current language differ from `\language` a “whatsit node” specifying the current language is inserted. No change is done (i.e. no language whatsit inserted) in “inner hmode”! If `\language < 0` or `\language > 255`, then 0 is used internally. Value from 0 to 255, so up to 256 “languages” (hyphen pattern numbers).

`\setlanguage` activates a `\language` hyphenation patterns (inserting the whatsit) without changing the value of `\language`. Useful in inner hmode, see `\language`.

`\patterns` allows to load hyphenation patterns associated with the current language number into T<sub>E</sub>X’s memory for compactification. Command allowed to be used during `initex` phase only, before the first paragraph of a document is set.

`\hyphenation` allows adding to the list of words to be hyphenated differently (exceptions) for the current `\language`. Can be used during both `initex` and `virtex`.

`\lefthyphenmin` & `\righthyphenmin` are numbers specifying the amount of characters starting and ending a word that must never be hyphenated with the **current** language. These values are not bind to the `\language`.

`\lccode` tells T<sub>E</sub>X which is the lower-case character of a (character positions with zero `\lccode` are not considered as word letters).

`\uchyph` when positive, words that start with an uppercase character (=letter does not equal its own `\lccode`), may be hyphenated.

`\discretionary` permits to specify how a character sequence must be hyphenated, providing the `pre-break text` with its discretionary hyphen and the `post-break text`. It inhibits hyphenation in the rest (before and after the discretionary) of the word. `\-` is the abbreviation of the most common case of discretionary break e.g. `\discretionary{-}{ }{ }`. This is a setting inserted in a word, for that word, at input time.

`\hyphenchar` is the hyphen of a font, can be coded to point out the hyphen character to use for a font. `\hyphenchar\font=-1` suppress all hyphenation for the current font. `\hyphenchar\font=' \-` is the common way to provide a default hyphen character for the current font. `\hyphenchar` changes are global.

`\defaultshyphenchar` in most macropackages is usually `\defaultshyphenchar=' \-`: (default hyphen to be used, if `\hyphenchar` isn't set. The '-' character as hyphen character is hard-wired in some other commands (e.g. `\hyphenation`).

`\hyphenpenalty` is a penalty for the line break after discretionary hyphen.

`\exhyphenpenalty` is a penalty for the line break after explicit hyphen.

`\brokenpenalty` is a penalty for a page break at a hyphenated line.

`\doublehyphendemerits` & `\finalhyphendemerits` [integer parameters] are values of demerits assessed to  $\text{T}_{\text{E}}\text{X}$  when breaking a paragraph into lines. These values take their effect when two consecutive lines end with discretionary breaks or if it is the last but one line of the entire paragraph. Demerits are in units of “badness squared”.

`\charsubdef` feature of  $\text{M}\text{I}\text{T}_{\text{E}}\text{X}$ : allows to input 8 bit characters, hyphenate words and substitute the corresponding pair: accent macro and letter of the 7 bit font for printing. No longer needed with  $\text{T}_{\text{E}}\text{X}$  3 and virtual fonts.

## What is a word (to be hyphenated)?

- T<sub>E</sub>X looks for potentially hyphenatable words by searching ahead from each glue item.
- Search bypasses characters whose `\lccode` is zero, or ligatures that begin with such a character. It also bypasses whatsits and **implicit kern** items (inserted by T<sub>E</sub>X because of information stored in a font).
- If search finds a character with nonzero `\lccode` (**starting character**), or if it finds a ligature that begins with such a character, hyphenation algorithm is called. But if any other type of item occurs before suitable starting letter is found, hyphenation is abandoned (until after the next glue item).
- Thus, a box, or rule or mark, or explicit kern must not intervene between glue and hyphenatable word.

- If a suitable starting letter is found, hyphenation is abandoned unless `\hyphenchar` of a font used is between 0 and 255, and unless a character of that number exists in the font.
- $\text{T}_{\text{E}}\text{X}$  continues to scan “admissible items”: a character in a current font with nonzero `\lccode`, ligature formed entirely with such characters, and an implicit kern. First inadmissible item terminates this process.
- If a trial word  $l_1, l_2, l_n$  has been found, hyphenation is still abandoned unless  $n \geq \lambda + \rho$ , where  $\lambda = \max(1, \text{\code\leftthyphenmin})$   
 $\rho = \max(1, \text{\code\rightthyphenmin})$
- Items immediately following the trial word must consist of zero or more characters, ligatures, and implicit kerns, followed immediately by either glue or an explicit kern or a penalty or a whatsit or an item of vertical material from `\mark`, `\insert` or `\vadjust`. Thus a box or rule or math formula or or discretionary following too closely upon a trial word will inhibit hyphenation
- ... (greatly simplified)

## Fine-tuning the linebreaking parameters

- Preset values are wrong for most cases. They have to be **fine-tuned** according to the **design** for particular document.
- Parameters (tolerancies, penalties, etc) have to be set with respect to the line length, language, font family, fontsize: hard to do it algorithmically.

## Tips and tricks

### How to suppress hyphenation?

- `\hyphenpenalty 10000` (independent of language).
- `\language 255` (language with no patterns).
- `\righthyphenmin 64` (probably the quickest).
- putting material in an `\hbox`.
- adding a word to the `\hyphenation` list.  
adding new patterns (e.g. of new odd level)

## How to allow hyphenation?

`\allowhyphens [macro] (\nobreak\hskip0pt)`

setting `\lccode` of a letter to nonzero (in addition to new patterns!, cf. `hypht1.tex` by Bernd Raichle). `( , ) , -` can be handled that way. Better solution than using active characters that break other things.

adding `\-` at all the word break

adding to `\hyphenation` list.

adding new patterns (eg. of new even level)

## Problems with lccode mapping

“ $\text{T}_{\text{E}}\text{X}$  does some things wrong, because it mixes between the input character encoding and font position encoding in ‘hmode’ (e.g. a token ‘ $\text{A}$ ’ with `\catcode 11` (= input encoding) is the same as `\‘\A` (= font encoding) and produces the character in position 65 of the current font. In ‘mmode’ you can distinguish between input and font encoding using `\mathcode` (and `\mathchar`) and the something similar should exist for the normal text mode.

Because of this mixing, the `\lccode` array is used for `\lowercase` (= input character coding) and for the hyphenation process (= font encoding, because an ‘hlist’ consists of font/char pairs). For this reason, `\lowercase/uppercase` can’t be used without restrictions for lowercasing/uppercasing text in the output.” (Bernd)

$\epsilon$ - $\text{T}_{\text{E}}\text{X}$  does solve this problem by allowing to globally save lccode mapping that is used during the `initex` phase.

`\savingshyphcodes` [integer parameter]  $\epsilon$ -**TEX** primitive. When `\patterns` command is executed and `\savingshyphcodes` has a positive value, the current `\lccode` values are saved as hyphenation codes for the current language. When the patterns have been compressed (always true with eVIRTEX) and hyphenation codes have been saved for the current language, they are used instead of `\lccode` values for hyphenation purposes (reading hyphenation exceptions and hyphenating words).

## Exercises

- How to suppress hyphenation at a page break?
- How to minimize hyphenation occurrence in one language of a document?
- How to set/fine-tune the parameters of a line-breaking algorithm such that we have (on average) five hyphenations on a page?
- How to achieve that the last word of a paragraph be hyphenated at least five letters from the right?

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

- How to typeset in a language with more than 256 characters in the alphabet?  
Using Omega [Slide] (or excessive usage of lccode and specially designed font encodings.)

## Hyphenation algorithm: problems to solve

“In theory, there is no difference between theory and practice,  
but in practice, there is. (Stephen D. Poe)

- Storing the whole hyphenated word-list space inefficient.
- Algorithms for hyphenation in T<sub>E</sub>X77 hardwired for English only.
- There are “long-distance” dependencies.
- **discreteness**: small change in input  $\Rightarrow$  fundamental change in output
- Dependence on semantics of a sentence: **rec-ord** and **re-cord**
- **exceptions** (unwanted connotation), exceptions of exceptions, . . . (Haller’s book for Czech)
- Stability of a language—language evolves new words, new hyphenation rules (German); no “insect in amber” solution. We cannot solve it “once and forever” (new words, . . .).

- **hard generalization:** compound words; word boundaries often have to be entered manually: `eigh\discretionary{t}{t}{t}een`

?

# Patterns

“**pattern** ORIGIN Middle English *patron* ‘something serving as a model’, from Old French. The change in sense is from the idea of patron giving an example to be copied. Metathesis in the second syllable occurred in the 16th cent. By 1700 *patron* ceased to be used of things, and the two forms became differentiated in sense.” (NODE, 1998 edition)

- Patterns: way of communication, scale of intelligence. Rhythm patterns in music conveying message. Patterns of disambiguation, patterns of parsing strategies, letter patterns, . . . , you name it.
- Algebraically, **patterns** are words over monoid  $\langle \Sigma \cup V, \cdot, \varepsilon \rangle$ .  
Binding of data to patterns: **classifying patterns** as words over  $\Sigma \cup \{.\} \cup V \cup A$ , where  $\langle A, \leq \rangle$  is partially ordered system.
- Frank Liang, DEK’s student at Stanford (Ph.D., 1983), developed the method and algorithms for hyphenation based on the idea of competing patterns of varying length.

## The Method

- general, language independent method.
- pattern is a substring with a information about hyphenation between characters: `hy3ph he2n .euro7 7tex.`
- odd numbers allow hyphenation, even numbers forbid hyphenation
- patterns are as short as possible to be as general as possible (new compound words, etc)
- pattern **compete** each other: instead of one big set of patterns, decomposition into several layered approximations (subpatterns)  $p_1$  (positive subpatterns),  $p_2$  (negative subpatterns—exceptions for  $p_1$ ),  $p_3$  (positive subpatterns to cover what has not been covered by “ $p_1 \wedge \neg p_2$ ”), ...

```
h y p h e n a t i o n
p1          1n a
p1          1t i o
p2          n2a t
p2          2i o
p2          h e2n
p3 h y3p h
p4          h e n a4
p5          h e n5a t
h0y3p0h0e2n5a4t2i0o0n
h y-p h e n-a t i o n
```

How to generate  
the patterns

?

## Techniques of Pattern Generation

“An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil’s behaviour.

Alan Turing, *Mind* (59):433-460, 1950

- ➡ machine learning technique for generation of patterns from an already hyphenated wordlist.
- ➡ PATGEN program written by Liang, generated patterns (hyphen . tex) for English from small Webster dictionary of hyphenated words (89% of hyphenation breaks covered).
- ➡ packed trie data structure for pattern (wordlist) storage; DFA with output (packed by suffix compression); **linear** search time with respect to the word length for pattern retrieval [Slide]
- ➡ pattern space minimization NP-complete problem

☞ number of tested candidates for patterns (and time of pattern generation)  
grows quickly with wider context and bigger alphabet

## Techniques of Pattern Generation (cont.)

- ➡ **Stratification technique:** elimination of “not necessary” training examples speeds up learning.
- ➡ **Bootstrapping technique:** Iterative bootstrapping technique for wordlist build and error correction.
- ➡ Final parameters of patterns generation setting—**Fine tuning:** With parameters of learning process we can fine-tune size and quality of patterns.

## Stratified Sampling Technique

“A large body of information can be comprehended reasonably well by studying more or less random portions of the data. The technical term for this approach is stratified sampling.” Knuth, 1991

Example of stratification rules for Czech wordlist generation using Czech morphological analyser:

1. We add only every 7th (actually 17th worked as well) derived word form from the full list to the PATGEN input list, with exceptions that:
2. every word stem must be accompanied by at least one derived form, and
3. every derived form with overlapping prefixes has to be present in the PATGEN input list as well, and

4. only one word with prefixes  $\bar{n}e$  (by which one can create negation to almost every word) and  $\bar{n}e\bar{j}$  (by which one creates superlatives) is included.

## Bootstrapping

“When engineers build a suspension bridge, first they draw a thin cable across the body of water. Then they use that cable to hoist a larger one. Then they use both cables to pull a third, and eventually create a thick cable of intertwined wires that you can drive a truck across (actually hundreds of trucks).”

Dave Winner

- Pattern development cycle starts with a small set of already hyphenated words.
- Patterns build from this set are applied on a bigger set and results are proofread.
- and again and again,...

## Parameters of Pattern Generation

- ➡ A candidate for a pattern (string of  $k$  letters of a given length and alphabet) is added to the list of new patterns in a given level when:  
$$good * good\_weight - bad * bad\_weight \geq threshold$$
- ➡ Counting good and bad behaviour for a given interletter position of a pattern candidate is done on the whole wordlist.
- ➡ Parameters can be fine-tuned to get size or precision optimized pattern (only heuristics, probably NP-complete problem).

Table 1: Liang's patterns for English (hyphen.tex), 4447 patterns, 1 hour (PDP-10)  
 CPU time, total patterns size 27667 B

| level | length | param        | hyphens |       | % correct | % wrong | # patterns |
|-------|--------|--------------|---------|-------|-----------|---------|------------|
| 1     | 2-3    | 1 2 20       | 67604   | 14156 | 76.6      | 16.0    | + 458      |
| 2     | 3-4    | 2 1 8        | 7407    | 11942 | 68.2      | 2.5     | + 509      |
| 3     | 4-5    | 1 4 7        | 13198   | 551   | 83.2      | 3.1     | + 985      |
| 4     | 5-6    | 3 2 1        | 1010    | 2730  | 82.0      | 0.0     | +1647      |
| 5     | 5-8    | 1 $\infty$ 4 | 1320    | 6428  | 89.3      | 0.0     | +1320      |

## PATGEN statistics for Czech hyphenation

Table 2: Standard Czech hyphenation with Liang's parameters for English

| level | length | param  | % correct | % wrong | # patterns | size  |
|-------|--------|--------|-----------|---------|------------|-------|
| 1     | 2–3    | 1 2 20 | 96.95     | 14.97   | + 855      |       |
| 2     | 3–4    | 2 1 8  | 94.33     | 0.47    | +1706      |       |
| 3     | 4–5    | 1 4 7  | 98.28     | 0.56    | +1033      |       |
| 4     | 5–6    | 3 2 1  | 98.22     | 0.01    | +2028      | 32 kB |

**Table 3: Standard Czech hyphenation with improved (size optimized) strategy**

| <b>level</b> | <b>length</b> | <b>param</b>  | <b>% correct</b> | <b>% wrong</b> | <b># patterns</b> | <b>size</b>  |
|--------------|---------------|---------------|------------------|----------------|-------------------|--------------|
| <b>1</b>     | <b>1–3</b>    | <b>1 2 20</b> | <b>97.41</b>     | <b>23.23</b>   | <b>+ 605</b>      |              |
| <b>2</b>     | <b>2–4</b>    | <b>2 1 8</b>  | <b>85.98</b>     | <b>0.31</b>    | <b>+ 904</b>      |              |
| <b>3</b>     | <b>3–5</b>    | <b>1 4 7</b>  | <b>98.40</b>     | <b>0.78</b>    | <b>+1267</b>      |              |
| <b>4</b>     | <b>4–6</b>    | <b>3 2 1</b>  | <b>98.26</b>     | <b>0.01</b>    | <b>+1665</b>      | <b>23 kB</b> |

## Czech/Slovak hyphenation

| # of words | # of hyphenation points |               |                  |
|------------|-------------------------|---------------|------------------|
|            | Correct                 | Wrong         | Missed           |
| Czech      |                         |               |                  |
| 372562     | 1019686<br>(98.26%)     | 39<br>(0.01%) | 18086<br>(1.74%) |
| Slovak     |                         |               |                  |
| 333139     | 1025450<br>(98.53%)     | 34<br>(0.01%) | 15273<br>(1.47%) |

## Optimalization of patterns

“The ultimate goal of mathematics is to eliminate all need for intelligent thought.”  
Graham, Knuth, Patashnik, Concrete Mathematics, 1989

## Czech hyphenation of compounds

**Table 4: Standard Czech hyphenation  
with improved (% of correct optimized) strategy**

| level | length | param | % correct | % wrong | # patterns | size  |
|-------|--------|-------|-----------|---------|------------|-------|
| 1     | 1–3    | 1 5 1 | 95.43     | 6.84    | +2261      |       |
| 2     | 1–3    | 1 5 1 | 95.84     | 1.17    | +1051      |       |
| 3     | 2–5    | 1 3 1 | 99.69     | 1.24    | +3255      |       |
| 4     | 2–5    | 1 3 1 | 99.63     | 0.09    | +1672      | 40 kB |

**Table 5: Czech hyphenation of compound words  
(Liang but allowing 1-length patterns in level 1)**

| <b>level</b> | <b>length</b> | <b>param</b>  | <b>% correct</b> | <b>% wrong</b> | <b># patterns</b> | <b>size</b>  |
|--------------|---------------|---------------|------------------|----------------|-------------------|--------------|
| <b>1</b>     | <b>1–3</b>    | <b>1 2 20</b> | <b>72.97</b>     | <b>14.32</b>   | <b>+ 300</b>      |              |
| <b>2</b>     | <b>2–4</b>    | <b>2 1 8</b>  | <b>69.32</b>     | <b>3.09</b>    | <b>+ 450</b>      |              |
| <b>3</b>     | <b>3–5</b>    | <b>1 4 7</b>  | <b>84.09</b>     | <b>4.02</b>    | <b>+ 870</b>      |              |
| <b>4</b>     | <b>4–6</b>    | <b>3 2 1</b>  | <b>82.61</b>     | <b>0.33</b>    | <b>+2625</b>      | <b>25 kB</b> |

**Table 6: Czech hyphenation of compound words**  
 (% of correct slightly optimized)

| <b>level</b> | <b>length</b> | <b>param</b>  | <b>% correct</b> | <b>% wrong</b> | <b># patterns</b> | <b>size</b>  |
|--------------|---------------|---------------|------------------|----------------|-------------------|--------------|
| <b>1</b>     | <b>1–3</b>    | <b>1 2 20</b> | <b>72.97</b>     | <b>14.32</b>   | <b>+ 300</b>      |              |
| <b>2</b>     | <b>2–4</b>    | <b>2 1 8</b>  | <b>69.32</b>     | <b>3.09</b>    | <b>+ 450</b>      |              |
| <b>3</b>     | <b>3–5</b>    | <b>1 4 3</b>  | <b>90.82</b>     | <b>4.24</b>    | <b>+3014</b>      |              |
| <b>4</b>     | <b>4–6</b>    | <b>3 2 1</b>  | <b>89.07</b>     | <b>0.36</b>    | <b>+2770</b>      | <b>40 kB</b> |

**Table 7: Czech hyphenation of composed words with parameters**  
 (% of correct optimized, but % of wrong and size increases)

| <b>level</b> | <b>length</b> | <b>param</b> | <b>% correct</b> | <b>% wrong</b> | <b># patterns</b> | <b>size</b>  |
|--------------|---------------|--------------|------------------|----------------|-------------------|--------------|
| <b>1</b>     | <b>1–3</b>    | <b>1 5 1</b> | <b>64.35</b>     | <b>5.34</b>    | <b>+1415</b>      |              |
| <b>2</b>     | <b>2–4</b>    | <b>1 5 1</b> | <b>67.10</b>     | <b>1.88</b>    | <b>+1261</b>      |              |
| <b>3</b>     | <b>3–5</b>    | <b>1 3 1</b> | <b>97.94</b>     | <b>5.39</b>    | <b>+8239</b>      |              |
| <b>4</b>     | <b>4–6</b>    | <b>1 3 1</b> | <b>97.91</b>     | <b>1.14</b>    | <b>+2882</b>      | <b>84 kB</b> |

## Exercises

- Downsize (get rid of) your hyphenation exception list.
- Bootstrap and develop wordlist and patterns for your native language.
- Downsize your hyphenation wordlist by the stratification technique.
- Bootstrap and develop wordlist and patterns for several hyphenation classes your native language. [Slide]
- Develop patterns for compound words of your native language.
- Develop patterns for hyphenation of long chemical formulae.
- Develop joint patterns for your two favourite languages (you have plenty of texts to typeset without tags for language switching).
- Develop patterns for phonetic hyphenation. [Slide]
- Develop patterns for a language with more than 255 characters (etymological dictionary). [Slide]

- Use T<sub>E</sub>X's primitives to implement spell checking in T<sub>E</sub>X (coloring of unknown words).

## Another applications

- PATGEN is not frequently used for another tasks sofar; pity, as the methods are problem independent (it is usually just encoding and formulation problem): (Thai) segmentation, Arabic letter hamza, sentence segmentation/tagging, . . . .
- We now have program OPATGEN for UNICODE handling and for bigger applications.

## Summary, suggested reading

- ☞ Mastering T<sub>E</sub>X's hyphenation parameters may improve quality of documents dramatically.
- ☞ New applications possible, new possibilities even after 30 years of usage!
- ☞ New tools available for wider range of applications: OPATGEN.
- ☞ Further reading: Appendix H of the T<sub>E</sub>Xbook, Yannis's patgen tutorial in CTAN/info, documentation of OPATGEN, there are papers in tugboat.

Go forth now and create masterpieces of the publishing art  
and of the perfect hyphenation patterns!