TUG 2021 – one set of patterns for 2 languages: Czech and Slovak

Ondřej & Petr Sojka

Czechoslovak Hyphenation Patterns, Word Lists, and Workflow – Why Hyphenate Czecho-Slovak Simply Syllabically?
TUG 2023 – patterns for nine languages

Take home message from today:
We developed prototype of Universal syllabic Segmentation for Czech, Slovak, Georgian, Greek, Polish, Russian, Turkish, Turkmen, and Ukrainian (cz, sk, ka, el, pl, ru, tr, tk a ua) and showed that

A) developing universal, up-to-date, high-coverage, and highly generalized universal syllabic segmentation patterns is possible, with high impact on virtually all typesetting engines, including web page renderers;

B) bringing wide character support into the hyphenation part of TEx suite of programs is possible by using Judy array data structure.
Contents

Introduction to Hyphenation Patterns

Hyphenation as Syllabification

Pattern Generation

Judy

Conclusions

Bibliography
Section 1

Introduction to Hyphenation Patterns
Patterns? Patterns!

“pattern ORIGIN Middle English patron ‘something serving as a model’, from Old French. The change in a 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.”


One can see the patterns everywhere: rhythm patterns in music or poetry conveying a message, patterns of behavior, letter patterns, …, you name it: *hyphenation patterns*.

Instead of storing the whole ever-growing dictionary of words with hyphenation points, a smaller set of rules called hyphenation patterns are generated from the dictionary, covering most if not all, information from there.
Patterns (of hyphenation) that compete with each other

Frank Liang, DEK’s student at Stanford (Ph.D., 1983), developed a general, language-independent method and algorithm for hyphenation based on the idea of competing patterns of varying length to cope with exceptions. [3].

- pattern is a substring with a piece of information about hyphenation between characters: hy3ph he2n n2at hen5at
- odd numbers permit, even numbers forbid hyphenation
- patterns are as short as possible to be as general as possible (new compound words, etc.)
- pattern compete with each other: instead of one big set of patterns, decomposition into layered sets generated in levels
  \( p_1 \) hyphenating patterns generated in level 1, \( p_2 \) inhibiting patterns—exceptions for \( p_1 \),
  \( p_3 \) hyphenating patterns to cover what has not been covered by “\( p_1 \land \neg p_2 \)”,…
Introduction to Hyphenation Patterns

Hyphenation lookup: an instance of dictionary problem

Given the already hyphenated word list of a language (dictionary), how to generate the patterns? Liang’s task was: less than 5,000 patterns, less than 30,000 bytes per language in format file (RAM during \TeX run).
**Introduction to Hyphenation Patterns**

**hyphen.tex generation by patgen (Liang, 1983) [3]**

<table>
<thead>
<tr>
<th>level</th>
<th>parameters</th>
<th>patterns</th>
<th>good</th>
<th>bad</th>
<th>good</th>
<th>bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 20 (4)</td>
<td>458</td>
<td>67,604</td>
<td>14,156</td>
<td>76.6%</td>
<td>16.0%</td>
</tr>
<tr>
<td>2</td>
<td>2 1 8 (4)</td>
<td>509</td>
<td>7,407</td>
<td>11,942</td>
<td>68.2%</td>
<td>2.5%</td>
</tr>
<tr>
<td>3</td>
<td>1 4 7 (5)</td>
<td>985</td>
<td>13,198</td>
<td>551</td>
<td>83.2%</td>
<td>3.1%</td>
</tr>
<tr>
<td>4</td>
<td>3 2 1 (6)</td>
<td>1647</td>
<td>1,010</td>
<td>2,730</td>
<td>82.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>5</td>
<td>1 ∞ 4 (8)</td>
<td>1320</td>
<td>6,428</td>
<td>0</td>
<td>89.3%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

A total of 4,919 patterns (4,447 unique) were obtained in hyphen.tex (27,860 bytes) from Webster’s Pocket dictionary (30,000+ words only). *Suffix-compressed packed trie* occupying 5,943 locations, with 181 outputs (less than 1% of original word list).

Patterns find 89.3% of the hyphens in the dictionary. 109 passes through the dictionary are needed.

Generation required about 1 hour of CPU time on PDP-11.
hyphen.tex used as a default for \language=0 in every \TeX installation

% The Plain TeX hyphenation tables
% [NOT TO BE CHANGED IN ANY WAY!]
% Unlimited copying and redistribution of this file
% are permitted as long as this file is not modified.
% Modifications are permitted, but only if
% the resulting file is not named hyphen.tex.
\patterns{% just type <return> if you’re not using INITEX
.ach4 .ad4der .af1t .al3t .am5at
...
Introduction to Hyphenation Patterns

**patgen program: machine learning from data**

One of the very first approaches that harnessed the power of data: Liang’s program patgen for generation of hyphenation patterns from a word list:

- efficient lossy or lossless *compression* of hyphenated dictionary with several orders of magnitude compression ratio.

- generated patterns have minimal length, e.g., shortest context possible, which results in their *generalization* properties.

- hyphenation of out-of-vocabulary words, too.

For Czech, *exact lossless* pattern generation is feasible [7] (TUG 2019), while reaching 100% coverage and simultaneously no errors, and the same holds for 2 language patterns (Czech+Slovak). [6]

Strict pattern minimality (size) is not an issue nowadays.
Section 2

Hyphenation as Syllabification
Approaches to hyphenation

**etymology-based**  The rule is to cut a word on the border of a compound word or the border of the stem and an affix, prefix, or negation. A typical example is the British hyphenation rules by the Oxford University Press [4].


Clear trend towards phonology-based, e.g. syllabic hyphenation.
Rules and examples of syllabic segmentations

- primarily syllabic according to pronunciation
- morphology only secondary (compound words)
- language per se, its vocabulary and hyphenation rules develop in time:
  roz-um (Haller, 1956, prefix roz, stem um) → ro-zum (2021, just syllables, etymology forgotten)\(^1\)

\(^1\)Similar shift is in other languages and cultures (UK→US)
Lexical Distance Among the Languages of Europe
Why syllabic patterns I?

- All languages share phonology principles, e.g. syllables, only orthography (correct writing) differs.
- Most languages define hyphenation according to pronunciation on syllable borders; morphology is secondary (compound words).
- Language per se, its vocabulary and hyphenation rules develop in time: roz-um (Haller, 1956, prefix roz, stem um) → ro-zum (2021, just syllables, etymology forgotten)
Why syllabic patterns II?

"Typographical prowess lies not in the ostentatious deployment of extravagant lexemes, but rather in the discerning mastery of the elegant harmony that interweaves characters, glyphs, and spaces, where the judicious orchestration of hyphenation serves as an exquisite testament to the printer's art." — not Edward Tufte

"Typographical prowess lies not in the ostentatious deployment of extravagant lexemes, but rather in the discerning mastery of the elegant harmony that interweaves characters, glyphs, and spaces, where the judicious orchestration of hyphenation serves as an exquisite testament to the printer's art." — not Edward Tufte
Why syllabic patterns III?

- Proof of concept for universal hyphenation patterns presented at RASLAN 2019 workshop [8].
- Superb results for Czech+Slovak patterns [6], consistent markup of syllables increased the quality of joint patterns.
- No hyphenation collisions — almost no words that have different hyphenations in different languages with syllabic hyphenation preferences.
- Available data resources to join the ‘syllabic hyphenation club’.

Question 1
Should universal syllabic patterns ever be developed?
# Wordlists collected for 13 languages

Table 1: Language resources and patterns used in pattern development experiments. All data have been converted to UTF-8 encoding and contain lowercase alphabetic characters only. Alphabet size (# chars) counts characters appearing in the language wordlist collected. Languages were chosen by the diversity in the size of patterns, syllables.

<table>
<thead>
<tr>
<th>Language</th>
<th># words</th>
<th># chars</th>
<th># patterns</th>
<th># syllables</th>
<th>pattern source, alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech+Slovak (cz+sk)</td>
<td>606,499</td>
<td>47</td>
<td>8,231</td>
<td>2,288,413</td>
<td>[6] correct optimized parameters, Latin</td>
</tr>
<tr>
<td>Greek (el-monoton)</td>
<td>10,432</td>
<td>48</td>
<td>1,208</td>
<td>37,736</td>
<td>[5] tex-hyphen repo, Greek</td>
</tr>
<tr>
<td>Panjabi (pa)</td>
<td>892</td>
<td>52</td>
<td>60</td>
<td>2,579</td>
<td>[5] tex-hyphen repo, Gurmukhi</td>
</tr>
<tr>
<td>Russian (ru)</td>
<td>19,698</td>
<td>33</td>
<td>4,808</td>
<td>75,532</td>
<td>[5] tex-hyphen repo</td>
</tr>
<tr>
<td>Tamil (ta)</td>
<td>46,526</td>
<td>48</td>
<td>71</td>
<td>209,380</td>
<td>[5] tex-hyphen repo, Tamil</td>
</tr>
<tr>
<td>Telugu (te)</td>
<td>28,849</td>
<td>66</td>
<td>72</td>
<td>125,508</td>
<td>[5] tex-hyphen repo, Telugu</td>
</tr>
<tr>
<td>Thai (th)</td>
<td>757</td>
<td>64</td>
<td>4,342</td>
<td>1,185</td>
<td>[5] tex-hyphen repo, Thai</td>
</tr>
<tr>
<td>Turkish (tr)</td>
<td>24,634</td>
<td>32</td>
<td>597</td>
<td>103,989</td>
<td>[5] tex-hyphen repo, Latin</td>
</tr>
<tr>
<td>Turkmen (tk)</td>
<td>9,262</td>
<td>30</td>
<td>2,371</td>
<td>33,080</td>
<td>[5] tex-hyphen repo, Latin</td>
</tr>
<tr>
<td>Ukrainian (ua)</td>
<td>17,007</td>
<td>33</td>
<td>1,990</td>
<td>65,099</td>
<td>[5] tex-hyphen repo, Cyrillic</td>
</tr>
</tbody>
</table>
Language alphabet overlaps

**Table 2**: Cells contain the number of lowercase characters that overlap between languages. In total, 13 languages contain in total 412 different lowercase characters, more than Patgen is capable of digesting.

<table>
<thead>
<tr>
<th>Language</th>
<th>cz+sk</th>
<th>ka</th>
<th>el</th>
<th>pa</th>
<th>pl</th>
<th>ru</th>
<th>ta</th>
<th>te</th>
<th>th</th>
<th>tr</th>
<th>tk</th>
<th>ua</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech+Slovak (cz+sk)</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>28</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Georgian (ka)</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Greek (el-monoton)</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Panjabi (pa)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Polish (pl)</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>22</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Russian (ru)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Tamil (ta)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Telugu (te)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Thai (th)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Turkish (tr)</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>25</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Turkmen (tk)</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>30</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Ukrainian (ua)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td></td>
</tr>
</tbody>
</table>
Section 3

Pattern Generation
Why universal?

laziness
Multilingual patterns hypothesis

1. worst case language combination $\implies \sim$ linear growth in pattern size (!)
2. same words are pronounced the same $\implies$ same patterns can be used
3. therefore, universal patterns are feasible
Why not just concat all the patterns?

- concat of all patterns – 204 kB
- correct optimized new universal patterns – 219 kB
- size optimized new universal patterns – 101 kB
- .ne3č
Data

noise
Pattern Generation

Conflicts in data

and the lack of ground truth
Generation workflow

- goal – show feasibility $\implies$ final workflow straightforward
- hyphenate wordlists, remove and study conflicts, generate patterns
### Generation output

<table>
<thead>
<tr>
<th>Word list</th>
<th>Parameters</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
<th>Size</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czechoslovak</td>
<td>custom</td>
<td>99.87%</td>
<td>0.03%</td>
<td>0.13%</td>
<td>32 kB</td>
<td>5,907</td>
</tr>
<tr>
<td>Czechoslovak</td>
<td>correctopt</td>
<td>99.99%</td>
<td>0.00%</td>
<td>0.01%</td>
<td>45 kB</td>
<td>8,231</td>
</tr>
<tr>
<td>Czechoslovak</td>
<td>sizeopt</td>
<td>99.67%</td>
<td>0.00%</td>
<td>0.33%</td>
<td>40 kB</td>
<td>7,417</td>
</tr>
<tr>
<td>Universal</td>
<td>custom</td>
<td>99.10%</td>
<td>0.28%</td>
<td>0.90%</td>
<td>77 kB</td>
<td>11,238</td>
</tr>
<tr>
<td>Universal</td>
<td>correctopt</td>
<td>99.83%</td>
<td>0.04%</td>
<td>0.17%</td>
<td>219 kB</td>
<td>29,742</td>
</tr>
<tr>
<td>Universal</td>
<td>sizeopt</td>
<td>98.25%</td>
<td>0.01%</td>
<td>1.75%</td>
<td>101 kB</td>
<td>14,321</td>
</tr>
</tbody>
</table>

The hypothesis that for as long as the same words are written the same, they are pronounced and thus also hyphenated the same, they can be hyphenated by one set of hyphenation patterns, was proven.
### Generalization of patterns: 10-fold cross validation check

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Good</th>
<th>Bad</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom</td>
<td>97.99%</td>
<td>1.06%</td>
<td>0.95%</td>
</tr>
<tr>
<td>correctopt</td>
<td>98.10%</td>
<td>1.28%</td>
<td>0.62%</td>
</tr>
<tr>
<td>sizeopt</td>
<td>97.50%</td>
<td>0.94%</td>
<td>1.56%</td>
</tr>
</tbody>
</table>

Generalization abilities are checked by 10-fold cross validation: patterns are generated from 9/10 of word list, and performance is measured on yet unseen 1/10 of “new” words. This is repeated 10 times and the performance results are averaged.
unihyphen.pat (correct optimized)

% Universal hyphenation patterns generated with
% correct optimized parameters (cs-sojka-correctoptimized.par)
% MIT License
.a4ak .a4al .a4am .a2b .a3b .aban3t .aba3t .abi3m .abi3n
.ab3r .ab4rad .a4c .ace3m .ace3t .a2d .a4damy .ada3n .adi3m
.a4dv .ae4re .ae3rá .a2f .af3r .a2g .ag4ni .a4h .a4has .a4hin
.a4his .ahi3t .a4hl .a4hm .a4ho .a4há
...

“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
The need for ‘big’ Patgen

We used only 9 languages in our pattern generation, because for more than 245 different lowercase characters current Patgen will fail.

Question 2
Should patterns be generated for CJK languages or for all syllabic languages and allow wide char representations?
Section 4
Judy array

- Dynamic data structure
- Implemented as a 256-ary tree
- Tries to effectively use cache memory and avoid main memory
- Fast and memory efficient
## Comparison of Judy and Trie

<table>
<thead>
<tr>
<th></th>
<th>Judy</th>
<th>Trie</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>30.407 $\mu$s</td>
<td>94.655 $\mu$s</td>
<td>3.11</td>
</tr>
<tr>
<td>Search</td>
<td>10.752 $\mu$s</td>
<td>16.195 $\mu$s</td>
<td>1.51</td>
</tr>
<tr>
<td>Number of allocations</td>
<td>52,443</td>
<td>617,042</td>
<td>11.77</td>
</tr>
<tr>
<td>Number of allocated bytes</td>
<td>6,546,120</td>
<td>13,426,975</td>
<td>2.05</td>
</tr>
</tbody>
</table>
Judy in Patgen

- Judy is fast when working with patterns
- Judy could be used in Patgen
- Problem of node addressing
  - Trie is implemented as an array
  - Judy can be accessed only on a global level

Question 3

Should the Unicode internal support be included in \TeX{} suite of programs like Patgen, or should it be handled by external segmenters?
UniPatgen

- This version would:
  - Enable generation for arbitrary alphabet
  - Be 4× slower
  - Dynamically allocate memory
- Unicode in I/O
- Unicode internally as a wide char

Question 4

Should UniPatgen add dependence on a Judy library, or should a more conservative solution be sought and implemented?

Question 5

If UniPatgen would be developed, should it be added to the distribution, together with Unicode patterns?
Section 5

Conclusions
Conclusions

Creating universal syllabic patterns is possible:
- 97%+ coverage
- Very good generalization (performance on unseen words)
- No errors (eventual errors after Patgen run could be added as whole word patterns).

Wide character support in Patgen (and \TeX) is possible
- Using Judy
- Allows generating and deploying Unicode patterns
That’s it, folks!

Comments, suggestions, questions, and *answers to posed questions* are welcome!

Let’s repeat the questions:
Questions to answer before doing the next steps

1. Should the universal syllabic patterns ever be developed?
2. If so, should patterns be generated for CJK languages or for all syllabic languages and allow wide char representations?
3. If so, should the Unicode internal support be included in \TeX suite of programs like Patgen, or should it be handled by external segmenters?
4. If UniPatgen would be developed, should it be added to the distribution, together with Unicode patterns?
5. If so, should UniPatgen add dependence on a Judy library, or should a more conservative solution be sought and implemented?
Section 6

Bibliography


