A Roadmap for Universal Syllabic Segmentation

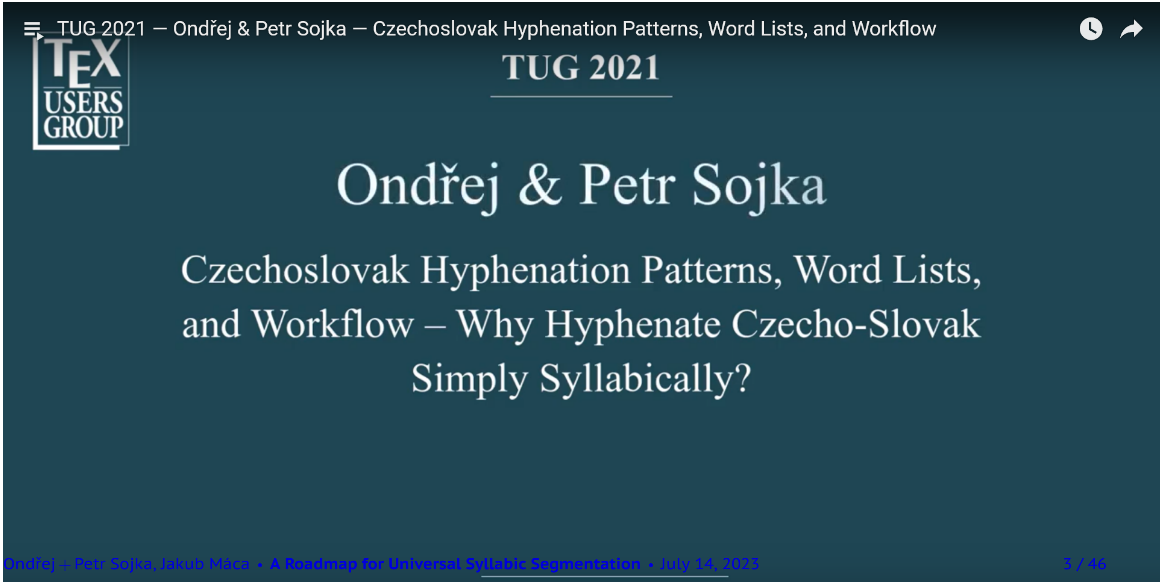
Ondřej Sojka, Petr Sojka, Jakub Máca

Faculty of Informatics, Masaryk University

July 14, 2023

EXIT
MAKE TEX GREAT AGAIN
REGISTERED TRADEMARK
DENIM BRAND
SHERATON

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



TUG 2021 — Ondřej & Petr Sojka — Czechoslovak Hyphenation Patterns, Word Lists, and Workflow

TUG 2021

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

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.”
– *New Oxford Dictionary of English, 1998 edition*

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 "$p_1 \wedge \neg p_2$"),…

Hyphenation lookup: an instance of dictionary problem

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

hy-phen-ation $\rightarrow$ 2 6

$\ldots \rightarrow \ldots$

$\ldots \rightarrow \ldots$

key $\rightarrow$ data

The solution to the dictionary problem:
For the key part (the word) to store
the data part (its division)

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).

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

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

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
...
```

`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].

phonology-based Hyphenation respects the pronunciation of syllables and allows for much more fluent reading. American publishers [2] and the Chicago Manual of Style [1] users prefer this pragmatic approach.

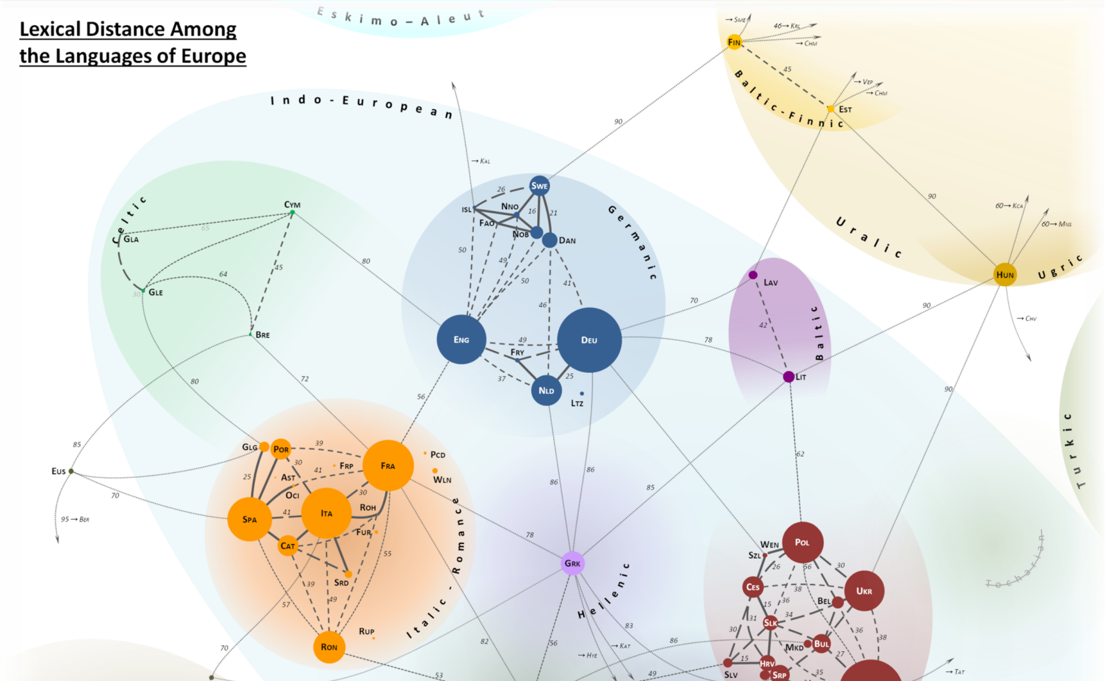
Clear trend towards phonology-based, e.g. syllabic hyphenation.

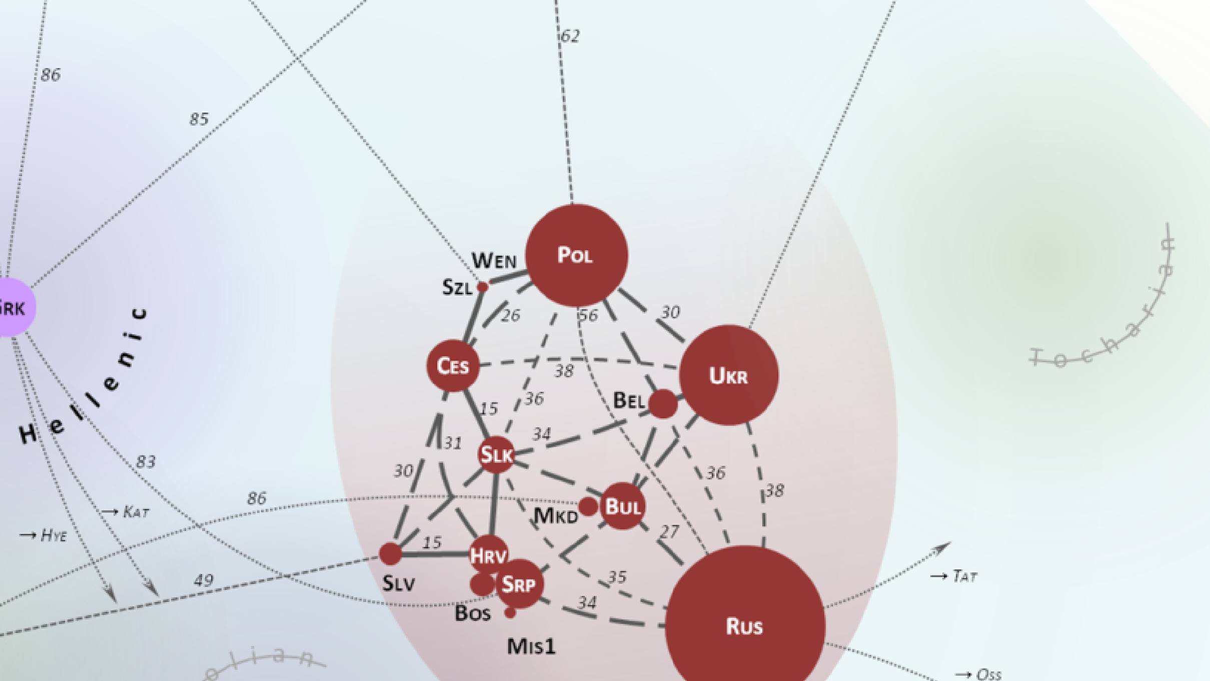
Rules and examples of syllabic segmentations

- 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) $\rightarrow$
  ro-zum (2021, just syllables, etymology forgotten)[1]

[1] Similar shift is in other languages and cultures (UK$\rightarrow$US)

Lexical Distance Among the Languages of Europe



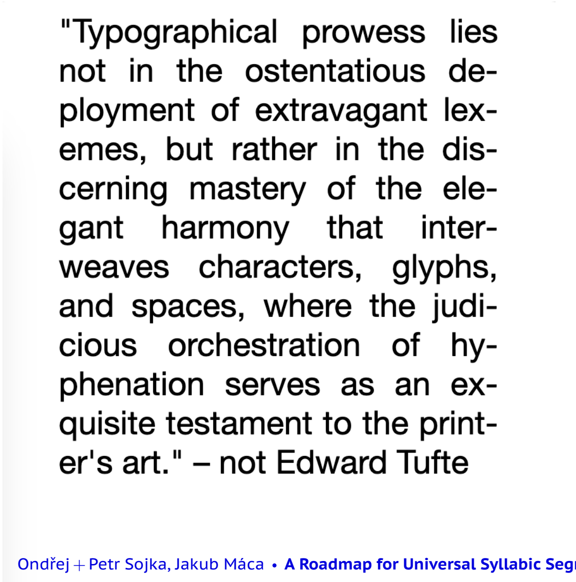


Pol
Wen
Szl
Ces
Ukr
Bel
Slk
Bul
Mkd
Hrv
Srp
Slv
Bos
Mis1
Rus
Hellenic
Tocharian
→ Kat
→ Hye
→ Tat
→ Oss
62
86
85
26
56
30
38
36
15
34
31
30
36
38
83
86
27
15
35
49
34

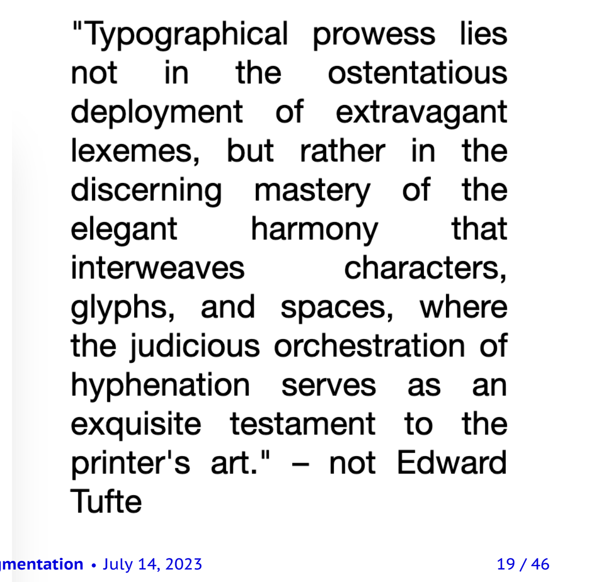
Why syllabic patterns I?

- 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) $\rightarrow$
  ro-zum (2021, just syllables, etymology forgotten)

Why syllabic patterns II?



"Typographical prowess lies
not in the ostentatious de-
ployment of extravagant lex-
emes, but rather in the dis-
cerning mastery of the ele-
gant harmony that inter-
weaves characters, glyphs,
and spaces, where the judi-
cious orchestration of hy-
phenation serves as an ex-
quisite testament to the print-
er'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

Language	# words	# chars	# patterns	# syllables	pattern source, alphabet
Czech+Slovak (cz+sk)	606,499	47	8,231	2,288,413	[6] correct optimized parameters, Latin
Georgian (ka)	50,644	33	2,110	224,799	[5] tex-hyphen repo, Georgian
Greek (el-monoton)	10,432	48	1,208	37,736	[5] tex-hyphen repo, Greek
Panjabi (pa)	892	52	60	2,579	[5] tex-hyphen repo, Gurmukhi
Polish (pl)	20,490	34	4,053	65,510	[5] tex-hyphen repo, Latin
Russian (ru)	19,698	33	4,808	75,532	[5] tex-hyphen repo
Tamil (ta)	46,526	48	71	209,380	[5] tex-hyphen repo, Tamil
Telugu (te)	28,849	66	72	125,508	[5] tex-hyphen repo, Telugu
Thai (th)	757	64	4,342	1,185	[5] tex-hyphen repo, Thai
Turkish (tr)	24,634	32	597	103,989	[5] tex-hyphen repo, Latin
Turkmen (tk)	9,262	30	2,371	33,080	[5] tex-hyphen repo, Latin
Ukrainian (ua)	17,007	33	1,990	65,099	[5] tex-hyphen repo, Cyrillic

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.

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

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

Conflicts in data

and the lack of ground truth

Generation workflow

- goal – show feasibility $\Longrightarrow$ final worflow straightforward
- hyphenate wordlists, remove and study conflicts, generate patterns

Generation output

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

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

Parameters	Good	Bad	Missed
custom	97.99%	1.06%	0.95%
correctopt	98.10%	1.28%	0.62%
sizeopt	97.50%	0.94%	1.56%

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

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

	Judy	Trie	Ratio
Insert	30.407 μs	94.655 μs	3.11
Search	10.752 μs	16.195 μs	1.51

Number of allocations	52,443	617,042	11.77
Number of allocated bytes	6,546,120	13,426,975	2.05

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 $4\times$ 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

Bibliography I

[1] Anonymous. *The Chicago Manual of Style*. 17th ed. Chicago: University of Chicago Press, Sept. 2017, p. 1146. isbn: 9780226287058.

[2] Philip Babcock Gove and Merriam Webster. *Webster's Third New International Dictionary of the English language Unabridged*. Springfield, Massachusetts, U.S.A: Merriam-Webster Inc., Jan. 2002.

[3] Franklin M. Liang. "Word Hy-phen-a-tion by Com-put-er". PhD thesis. Stanford University, Aug. 1983, p. 44. url: `https://www.tug.org/docs/liang/liang-thesis.pdf`.

[4] R. E. Allen, ed. *The Oxford Spelling Dictionary*. Vol. II. The Oxford Library of English Usage. Oxford University Press, 1990, p. 299.

Bibliography II

[5] Arthur Rosendahl and Mojca Miklavec. *TEX hyphenation patterns*. eng. Accessed 2023-07-05. 2023. url: `http://hyphenation.org/tex`.

[6] Petr Sojka and Ondřej Sojka. “New Czechoslovak Hyphenation Patterns, Word Lists, and Workflow”. eng. In: *TUGboat* 42.2 (2021). issn: 0896-3207. url: `https://doi.org/10.47397/tb/42-2/tb131sojka-czech`.

[7] Petr Sojka and Ondřej Sojka. “The Unreasonable Effectiveness of Pattern Generation”. In: *TUGboat* 40.2 (2019), pp. 187–193. url: `https://tug.org/TUGboat/tb40-2/tb125sojka-patgen.pdf`.

Bibliography III

[8] Petr Sojka and Ondřej Sojka. “Towards Universal Hyphenation Patterns”. In: *Proceedings of Recent Advances in Slavonic Natural Language Processing—RASLAN 2019*. Ed. by Aleš Horák, Pavel Rychlý, and Adam Rambousek. `https://is.muni.cz/publication/1585259/?lang=en`. Karlova Studánka, Czech Republic: Tribun EU, 2019, pp. 63–68. url: `https://nlp.fi.muni.cz/raslan/2019/paper13-sojka.pdf`.