A small tutorial on the multilingual features of PatGen2

Yannis Haralambous

I will very briefly discuss and illustrate by an example the features of PatGen2[1], related to the *translation* file.

1 Syntax of the translation file

1.1 What is the problem

The problem is that in our (non-English, non-Latin, non-Indonesian[2]) languages, we have more characters than just the 26 letters of the Latin alphabet. Not to mention that alphabetical orders can be quite different (remember: in Spanish, cucaracha comes before chacal), and that there can be many ways to express these additional characters: for example the French œ (like in ¡¡ Histoire de l'œil ¿¿ by Georges Bataille) can be given to TEX as an 8-bit character, or as `\oe␣`, or as `\oe{}`, or as `^^f7` (since we are *all* now working with DC fonts) and so forth.

Now let's make (or enhance the) hyphenation patterns for our language(s). Fred Liang has not provided PatGen with the extensibility feature: before PatGen2, if your language had less than 26 letters you could hack patterns by substituting letters, if it had more...then the odds were against you.

So the problem is to express additional characters in the patterns, in some understandable way (and if possible to get the results in the appropriate alphabetical order).

1.2 The solution

PatGen2 allows you to include more characters than the 26 letters of the alphabet. You can specify arbitrary many *input forms* for each of them (these

---

[1] the extension to PatGen by Peter Breitenlohner, author of many beautiful and extremely useful TEXware, such as TEX–XET, `DVIcopy` etc.

[2] As far as I know, English, Latin and Indonesian are the only languages without diacritics or special characters...

forms will be identified internally), and you can specify the alphabetical order of your language. The latter feature is only interesting when you want to study the patterns file to correct bugs, it doesn't affect the result.

These informations are transmitted to PatGen2 by means of a *translation* file (usually with the extension `.tra`). The syntax of this file is not very user-friendly, but if you deal with PatGen you are supposed to be a hacker anyway, and hackers just **love** weird-but-efficient-syntaxes [the proof: we all like TeX![3]].

On the first line you specify the `\lefthyphenmin` and `\righthyphenmin` values. The seven first positions of this line are used as follows: two positions for `\lefthyphenmin`, two positions for `\righthyphenmin`, and one position for substitute symbols of each of `.`, `-`, `*` in the output file. This can be useful if you are going to use the dot the asterisk or the hyphen to describe your additional characters (for example one may use the dot to specify the dot accent on a letter).

The next lines concern letters of the alphabet of our language. The first position of each line is the delimiter of our fields. You know from languages like C or `awk` that it is very useful to define our own delimiter (instead of the blank space) if for any reason we want to *include* a blank space into our fields. This will be the case in the example: one usually leaves a blank space after a TeX macro without argument.

Otherwise just leave the first position blank. Next comes the standard representation of our character (preferably lowercase). This is how the character will appear in the patterns and in the hyphenated output file.

Follows again a delimiter, and all the *equivalent* representations of your character: its uppercase form, and as many other input forms we wish, always followed by a delimiter.

The order of lines specifies the alphabetical order of your characters.

When PatGen sees two consecutive delimiters, it stops reading; so we can include comments after that.

2 An example

To illustrate this I have taken some Greek words (people who know *Χάρρυ Κλύνν* and his album Π*ατάτες* will recognize some of these words. . .) and included an α with accent, and a variant representation of π, in form of a macro `\varpi␣`. Also I followed the Greek alphabetical order in the translation file.

Here is my list of words:

```
A-NU-PO-TA-QTOS
A-KA-TA-M'A-QH-TOS
A-EI-MNH-STOS
MA-NA
```

---

[3] . . . just joking

```
TOUR-KO-GU-FTIS-SA
NU-QO-KO-PTHS
LI-ME-NO-FU-LA-KAS
MPRE-LOK
A-GA-\varpi A-EI
PEI-RAI-’AS
```

(file `greek.dic`); and here is my translation file (`greek.tra`):

```
 1 1
 a A
 ’a ’A
 b B
 g G
 d D
 e E
 z Z
 h H
 j J
 i I
 k K
 l L
 m M
 n N
 o O
#p#P#\varpi ##
 r R
 s S
 t T
 u U
 f F
 q Q
 y Y
 w W
```

(yes, yes, that’s not a joke; Greek really uses values 1 and 1 for minimal right and left hyphenations!). As you see, I have chosen `#` as delimiter in the case of π, because `\varpi` is supposed to be followed by a blank space. And then I wrote two delimiters (`##`) to show PatGen2 that I finished talking about π.

Here is what I got on my console:

```
PatGen -t greek.tra -o greek.out greek.dic
This is PatGen, C Version 2.1 / Macintosh Version 2.0
Copyright (c) 1991-93 by Wilfried Ricken. All rights reserved.
```

```
left_hyphen_min = 1, right_hyphen_min = 1, 24 letters
0 patterns read in
pattern trie has 266 nodes, trie_max = 290, 0 outputs
hyph_start:  1
hyph_finish: 2
pat_start:   2
pat_finish:  4
good weight: 1
bad weight:  1
threshold:   1
processing dictionary with pat_len = 2, pat_dot = 1

0 good, 0 bad, 31 missed
0.00 %, 0.00 %, 100.00 %
69 patterns, 325 nodes in count trie, triec_max = 440
25 good and 42 bad patterns added (more to come)
finding 29 good and 0 bad hyphens, efficiency = 1.16
pattern trie has 333 nodes, trie_max = 349, 2 outputs
processing dictionary with pat_len = 2, pat_dot = 0

29 good, 0 bad, 2 missed
93.55 %, 0.00 %, 6.45 %
45 patterns, 301 nodes in count trie, triec_max = 378
2 good and 43 bad patterns added
finding 2 good and 0 bad hyphens, efficiency = 1.00
pattern trie has 339 nodes, trie_max = 386, 6 outputs
processing dictionary with pat_len = 2, pat_dot = 2

31 good, 0 bad, 0 missed
100.00 %, 0.00 %, 0.00 %
47 patterns, 303 nodes in count trie, triec_max = 369
0 good and 47 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 344 nodes, trie_max = 386, 13 outputs
51 nodes and 11 outputs deleted
total of 27 patterns at hyph_level 1

pat_start:   2
pat_finish:  4
good weight: 1
bad weight:  1
threshold:   1
processing dictionary with pat_len = 2, pat_dot = 1
```

```
31 good, 0 bad, 0 missed
100.00 %, 0.00 %, 0.00 %
27 patterns, 283 nodes in count trie, triec_max = 315
0 good and 27 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 295 nodes, trie_max = 386, 4 outputs
processing dictionary with pat_len = 2, pat_dot = 0

31 good, 0 bad, 0 missed
100.00 %, 0.00 %, 0.00 %
27 patterns, 283 nodes in count trie, triec_max = 303
0 good and 27 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 320 nodes, trie_max = 386, 6 outputs
processing dictionary with pat_len = 2, pat_dot = 2

31 good, 0 bad, 0 missed
100.00 %, 0.00 %, 0.00 %
24 patterns, 280 nodes in count trie, triec_max = 286
0 good and 24 bad patterns added
finding 0 good and 0 bad hyphens
pattern trie has 328 nodes, trie_max = 386, 9 outputs
35 nodes and 7 outputs deleted
total of 0 patterns at hyph_level 2
hyphenate word list? y
writing PatTmp.2

31 good, 0 bad, 0 missed
100.00 %, 0.00 %, 0.00 %
Time elapsed: 0:37:70 minutes.
```

and here are the results: first of all the patterns (file `greek.out`)

```
a1g
a1e
a1k
a1m
a1n
a1p
a1t
a1q
'a1q
e1l
e1n
```

```
h1t
i1'a
i1m
i1r
1ko
o1g
o1p
o1t
o1f
r1k
s1s
1st
u1l
u1p
u1f
u1q
```

As you see, `o1t` comes before `o1f`: (smile) simply because we are talking about $o1\tau$ and $o1\phi$. So the alphabetical order is well respected. Also you see that PatGen2 has read the word `A-GA-\varpi␣A-EI` exactly as if it were `A-GA-PA-EI` and has made out of it the pattern `a1p` (if you look in the input words there is no other reason for this pattern to exist).

And here is our result (file `PatTmp.2`):

```
a*nu*po*ta*qtos
a*ka*ta*m'a*qh*tos
a*ei*mnh*stos
ma*na
tour*ko*gu*ftis*sa
nu*qo*ko*pths
li*me*no*fu*la*kas
mpre*lok
a*ga*pa*ei
pei*rai*'as
```

(Of course, it is correct; you think I would have shown it if it weren't correct?) As you see there is no `\varpi␣` anymore: PatGen2 has really replaced it by `p`, and so `A-GA-\varpi␣A-EI` has become `a*ga*pa*ei`.

3 Where do I find more information?

I voluntarily didn't discussed the various parameters used for pattern generation. For these there is very good litterature:

- the TeXbook, by the Grand Wizard of TeX arcana, appendix H;

- LaTeX Erweiterungsmöglichkeiten, by Helmut Kopka, Addison-Wesley, Pages 482–489;
- Swedish Hyphenation for TeX, by Jan Michael Rynning, [sorry, I don't know where this paper is published];
- Word Hy-phen-a-tion by Com-put-er, Stanford University Report `STAN-CS-83-977`;
- forthcoming paper by Dominik Wujastyk, on British hyphenation;
- Hyphenation Patterns for Ancient Greek and Latin, TUGboat 13 (4), pages 457–469.
- and many others...

Where to get PatGen2? probably everywhere, but certainly in Stuttgart (`IP 129.69.1.12`), `soft/tex/systems/pc/utilities/patgen.zip` (take a look also at `soft/tex/systems/knuth/texware/patgen.version2.1/patgen.README`).

4 Go forth, etc etc

OK, once again Go Forth and make masterpieces of hyphenation patterns... **but** please get in touch with the TWGMLC (Technical Working Group on Multiple Language Coordination) first, since people there are working on many languages: maybe they have already done what you need and are still testing it; or maybe they haven't and in that case you could help us a lot.