Generating Natural Language Numerals with TeX

Ivan Derzhanski
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
iad58g@gmail.com

Milena Veneva
Independent Researcher
milena.p.veneva@gmail.com

Abstract

Sometimes one needs to produce a text in which many numbers have to be written out in words. Writing such a text and ensuring it is error-free can be a burden, especially if the author is not fluent in the language. Such may occur when working on a reference grammar, a research paper or presentation, or a problem on number names for a contest in linguistics. A remedy is to prepare the text with TeX and let some parts be generated automatically. The human effort this takes is to compose a grammar that describes the features of the numeral system. This paper discusses how this is done.

Keywords: number names, number systems, numerals, TeX, typesetting

1. Introduction

Sometimes one needs to produce a text in which many numbers have to be written out in words. Writing such a text and ensuring it is error-free can be a burden, especially if the quantity of numbers is very large, they change a lot during the editing, or the author is not fluent in the language. Such may occur when working on a reference grammar, a research paper or presentation, or a problem on number names for a contest in linguistics (Derzhanski and Veneva, 2018).

This burden can become lighter if the text is prepared with TeX (Knuth, 1986). Some parts can then be generated automatically (Derzhanski, 2013), and number names are a prime candidate. The human effort this takes is to compose a grammar that describes the features of the numeral system, but we leave the bulk of the typing and the proofreading to TeX.

It should be noted that in this day 'producing a document with TeX' tends to mean writing it in LaTeX 2_ε (LaTeX 2_ε, 2018) or another such extension, but in fact the arithmetic operations, conditionals, switch-case statements, and other programming commands which facilitate the process of writing a semi-self-generating grammar pertain to pure TeX, albeit freely used within LaTeX 2_ε.

2. The Problems

The method has been employed for generating numerals in eight languages in the statements and solutions of linguistic problems on number names that have been assigned at different instalments of the International Linguistic Olympiad (IOL) (www.ioling.org/) or national-scale contests in linguistics in Bulgaria (Derzhanski, 2009: Chapters 12 and 13). Here are the sources, the languages, their ISO 639-3 codes, families and countries where spoken:

1. IOL7 (Evgenia Korovina and Ivan Derzhanski): Sulka (sua: isolate, Papua New Guinea);
2. IOL8 (Ksenia Gilyarova): Drehu (dhv: Austronesian, New Caledonia);
3. IOL10 (Ksenia Gilyarova): Umbu-Ungu (ubu: Trans-New Guinea, Papua New Guinea);
4. IOL13 (Milena Veneva): Arammba (stk: South-Central Papuan, Papua New Guinea);
5. IOL13 (Milena Veneva): Classical Nahuatl (nci: Uto-Aztecan, Aztec Empire);
6. IOL15 (Milena Veneva): Birom (bom: Atlantic-Congo, Nigeria);

7. Winter Mathematics Contest 2000 (Ivan Derzhanski): Georgian (kat: Kartvelian, Georgia);
8. National Contest in Linguistics 2001 (Ivan Derzhanski): Yoruba (yor: Atlantic-Congo, Nigeria).

Table 1 summarises the principal features of the number systems of these languages, as well as Bulgarian as a point of comparison; that is, the answers to the following questions:

1. What is the base of the number system, and are there supplementary bases (such are often 5 and/or 10, and then perhaps 15, when the principal base is 20)?
2. Does the base have alternative (suppletive) names?
3. Are there any other numbers that play a base-like part in the number system?
4. Does the language use subtraction, or better, do the numbers just below the base behave – or are they formed – in an unusual way?
5. Does the language use overcounting (Menninger, 1969; Hanke, 2005)?
6. What, if any, arithmetic operations are marked?
7. Is the order of addends ($+$) and multiplicands ($\times$) ascending ($\nearrow$) or descending ($\searrow$)?
8. Are there any (morpho)phonological changes in the derivation of number names?

language	sua			dhv	ubu	stk	nci	bom	kat	yor	bul
1: base	20	20	20+	20+	24	6	20+	12	20+	20+	10
2: other names	no			yes	no	yes	yes	yes	no	no	no
3: other bases	3	4	no	no	4	no	no	no	no	no	no
4: subtraction (‘$-$’)	no			no	no	no	no	yes	no	yes	no
5: overcounting (‘$\neg$’)	no			no	yes	no	no	no	no	no	no
6: operations	$+, \times 2$			$+$	$\neg$	no	$+$	$+, \times$	$+, \times$	$+, -$	$+$
7(a): word order $+$	$\nearrow$			$\nearrow, \searrow$	$\searrow$	$\searrow$	$\searrow$	$\searrow$	$\searrow$	$\nearrow$	$\nearrow$
7(b): word order $\times$	$\searrow$			$\nearrow$	$\searrow$	$\nearrow$	$\nearrow$	$\searrow$	$\nearrow$	$\searrow$	$\nearrow$
8: (morpho)phonology	no			yes	no	no	yes	yes	yes	no	yes

Table 1: Linguistic phenomena in several number name systems.

3. The Idea

The idea of writing a computer program to convert a number to words is not original. It can be found under the form of a popular programming exercise on applying conditional and switch-case operators and manipulating strings of characters. For instance, problem #5.6 in (Dreyfus and Gangloff, 1975) concerns composing a program in Fortran IV to write out a given one- or two-digit number in French. Likewise, problem #31 in (Todorova et al., 2008) shows one of the ways to convert a two-digit number input from the keyboard to its Bulgarian name in C++.

4. TEX Definitions

Typesetting with TEX (Knuth, 1986) is akin to writing a program in several ways.

One is that frequently used constructions can be formulated as macro definitions—control sequences that can be evoked every time we need them. They can be mathematical formulae, words, sentences or even whole text passages. This reduces the number of keystrokes, typing errors and inconsistencies.

Another is that information of various types can be stored in variables (registers), which can be assigned values and performed operations on (in particular, integer arithmetics).

Finally, there are flow of control constructions of the if-then-else kind (depending on the outcome of a numeric comparison or another boolean condition) and the switch type (depending on the non-negative integer value of a variable).

5. Implementation

5.1. Bulgarian

Bulgarian has a decimal number system; up to 99 multiplication is expressed by juxtaposition and addition by the preposition **na** ‘on, over’ and the conjunction **i** ‘and’. The number names in this range are formed as follows:

Rule no.		Lines
1	**edno** 1, **dve** 2, **tri** 3, **chetiri** 4, **pet** 5, **shest** 6, **sedem** 7, **osem** 8, **devet** 9	#20
2	**deset** 10	#18
3	α·**na**·**deset** = $10+\alpha$ $(1 \le \alpha \le 9)$ (if $\alpha = 1$, the stem is **edi**; if $\alpha = 2$, the stem is **dva**)	##17–18
4	β·**deset** $(2 \le \beta \le 9)$ (if $\beta = 2$, the stem is **dva**)	#13
5	β·**deset i** $\alpha = \beta \times 10 + \alpha$ $(1 \le \alpha \le 9, 2 \le \beta \le 9)$ (if $\beta = 2$, the stem is **dva**)	##13–14, #20

Table 2: Rules for Bulgarian

Figure 1 shows the TEX macros for generating number names in the range $[1;99]$. The top macro is `\blg` (e. g., `\blg{42}` produces **chetirideset i dve** $4 \times 10 + 2$), and it turns to the auxiliary `\bln`, which yields numerals in the range $[1;9]$, in the general case for both the `\tens` and the `\ones`, into which the argument is split in lines 9–10. Since by default counting is done in the neuter gender but the number 2 as a multiplier in 20 and both 1 and 2 as addends in the second decade are in the masculine, the flag `\ifneutrum` is set to indicate that a neuter form is required.

```
1  \newcommand \bln[1]{\ifcase #1
2      \or ed\ifneutrum no\else i\fi \or dv\ifneutrum e\else a\fi
3      \or tri\or chetiri\or pet\or shest\or sedem\or osem\or devet\fi
4  }
5  \newcount \tens \newcount \ones
6  \newif \ifneutrum
7  \newcommand \blg[1]{%
8      \ifnum #1<100
9          \tens=#1\divide \tens by 10
10         \ones=-\tens \multiply \ones by 10\advance \ones by #1
11         \neutrumfalse
12         \ifnum 1<\tens
13             \bln{\tens}deset%
14             \ifnum 0<\ones \space i \fi
15         \fi
16         \ifnum 1=\tens
17             \ifnum 0<\ones \bln{\ones}na\fi
18             deset%
19         \else
20             \ifnum 0<\ones \neutrumtrue \bln{\ones}\fi
21         \fi
22     \fi
23 }
```

Figure 1: Macro definitions for generating Bulgarian numerals up to 99.

5.2. Birom

The use of macros to avoid typos is most opportune when the number names contain many diacritics, which happens to be the case in Birom. The number names up to 120 (the range featured in the problem; in fact the same rules hold for $[121;131]$ as well, and only fail to do so at 132) obey the rules in Table 3.

The TEX macros for generating these numerals in Birom are shown in Figure 2. The main macro

Rule no.		Lines
1	gwīnìŋ 1, bà 2, tàt 3, nààs 4, tùŋūn 5, tìīmìn 6, tàāmà 7, rwīīt 8	#30
2	ʃāā-$\alpha = 12 - \alpha$ ($1 \le \alpha \le 3$): ʃāātàt 9, ʃāābà 10, ʃāāgwīnìŋ 11	#30
3	kūrū 12	#21
4	bā-kūrū bī-$\bar{\gamma}$ = $\gamma \cdot 12$ ($2 \le \gamma \le 8$), bā-kūrū ʃāā-bī-$\bar{\gamma}$ = $(12 - \gamma) \cdot 12$ ($1 \le \gamma \le 2$) (the tone in the first syllable of γ becomes middle)	##22–23
5	β ná $\left\{\begin{matrix}\text{gwɛ̄ } (\delta = 1) \\ \text{vɛ̀ } (2 \le \delta \le 11)\end{matrix}\right\} \delta = \beta + \delta \qquad (\beta = k \cdot 12)$	##21–28

Table 3: Rules for Birom

```
1  \newcommand \biron [3]{%
2      \newcount \numm \numm=#3%
3      \ifnum 8<\numm \textesh\={a}\={a}%
4          \advance \numm by-12\numm =-\numm
5      \fi
6      #1\ifcase \numm \or gw\={\i}n\`{\i}\ng%
7          \or b#2{a}\or t#2{a}t\or n#2{a}#2{a}s%
8          \or t#2{u}\ng\={u}n\or t#2{\i}\={\i}m\`{\i}n%
9          \or t#2{a}\={a}m\`{a}\or rw\={\i}\={\i}t\fi
10 }
11 \newcount \biggestBM
12 \newcount \biggestBA
13 \newcommand \birom [1]{%
14     \newcount \numb \numb=#1
15     \ifnum 121>\numb
16         \ifnum 11<\numb
17             \biggestBM=\numb
18             \divide \biggestBM by 12
19             \biggestBA=\biggestBM
20             \multiply \biggestBA by 12
21             \ifnum 1=\biggestBM k\={u}r\={u}%
22             \else b\={a}k\={u}r\={u}
23                 \biron{b\={\i}}{\=}{\biggestBM}\fi
24             \advance \numb by -\biggestBA
25             \ifnum 0<\numb \space n\'{a}
26                 \ifnum 1=\numb gw\={\textepsilon}
27                 \else v\`{\textepsilon} \fi
28             \fi
29         \fi
30         \ifnum 0<\numb \biron {}{\`}{\numb}\fi
31     \fi
32 }
```

Figure 2: Macro definitions for generating Birom numerals.

is `\birom` (e. g., `\birom{117}` produces *bākūrū ʃāābītāt ná vɛ̀ ʃāātàt* $(12 - 3) \times 12 + (12 - 3)$). The auxiliary `\biron` is somewhat more complex than its counterpart for Bulgarian: it implements the expression of 9–11 by subtraction and the prefixing of *bī-* and the tone change in coefficients in the names of dozens.

A part of the text of the problem on Birom numerals prepared with the use of this method (as LaTeX source and typeset) is in Figure 3.

```
\newcommand \numB [1]{\mbox {\bomfont {\birom {#1}}}}

\begin{enumerate}
 \item $\numB{5}^2 + \numB{3} + \numB{4} = \numB{32}$
 \item $\numB{3}\totheB{4} = \numB{81}$
 \item $\numB{7}^2 + \numB{9} + \numB{1} = \numB{59}$
 \item $\numB{9}\totheB{1} = \numB{9}$
 \item $\numB{8}^2 + \numB{2} + \numB{5} = \numB{71}$
 \item $\numB{2}\totheB{5} = \numB{32}$
 \item $\numB{9}^2 + \numB{4} + \numB{3} = \numB{88}$
 \item $\numB{4}\totheB{3} = \numB{64}$
 \item $\numB{16} + \numB{21} = \numB{18} + \numB{2} + \numB{17}$
\end{enumerate}
...
\item Write the numbers \numB{36}, \numB{11}, \numB{12}
and the equalities (A) and (B) in numerals.
%
\begin{enumerate}
 \item[A.] $\numB{108} - \numB{3} - \numB{13} = \numB{92}$
 \item[B.] $\numB{49} - \numB{14} - \numB{15} = \numB{20}$
\end{enumerate}
```

1. **tùŋūn**^{2} + **tàt** + **nààs** = **bākūrū bībā ná vɛ̀ rwīīt**
2. **tàt**$^{\text{nààs}}$ = **bākūrū bītīīmìn ná vɛ̀ ʃāātàt**
3. **tàāmà**^{2} + **ʃāātàt** + **gwīnìŋ** = **bākūrū bīnāās ná vɛ̀ ʃāāgwīnìŋ**
4. **ʃāātàt**$^{\text{gwīnìŋ}}$ = **ʃāātàt**
5. **rwīīt**^{2} + **bà** + **tùŋūn** = **bākūrū bītūŋūn ná vɛ̀ ʃāāgwīnìŋ**
6. **bà**$^{\text{tùŋūn}}$ = **bākūrū bībā ná vɛ̀ rwīīt**
7. **ʃāātàt**^{2} + **nààs** + **tàt** = **bākūrū bītāāmà ná vɛ̀ nààs**
8. **nààs**$^{\text{tàt}}$ = **bākūrū bītūŋūn ná vɛ̀ nààs**
9. **kūrū ná vɛ̀ nààs** + **kūrū ná vɛ̀ ʃāātàt** = **kūrū ná vɛ̀ tìīmìn** + **bà** + **kūrū ná vɛ̀ tùŋūn**

(b) Write the numbers **bākūrū bītāt**, **ʃāāgwīnìŋ**, **kūrū** and the equalities (A) and (B) in numerals.

A. **bākūrū ʃāābītāt** – **tàt** – **kūrū ná gwɛ̄ gwīnìŋ** = **bākūrū bītāāmà ná vɛ̀ rwīīt**
B. **bākūrū bīnāās ná gwɛ̄ gwīnìŋ** – **kūrū ná vɛ̀ bà** – **kūrū ná vɛ̀ tàt** = **kūrū ná vɛ̀ rwīīt**

Figure 3: An excerpt from the statement of the Birom problem.

5.3. Yoruba

Yoruba operates a decimal–vigesimal system; its most peculiar feature is that subtraction (of 10 from a whole twenty and of 1 to 5 from a whole ten) is liberally used where most other languages use addition. The rules produce the numbers up to 184 except for the range [25; 34], because 30 has a suppletive name, which was not featured in the problem for which the macros shown here were made.[1]

Rule no.		Lines
1	*ọkan* 1, *eji* 2, *ẹta* 3, *ẹrin* 4, *arun* 5, *ẹfa* 6, *eje* 7, *ẹjọ* 8, *ẹsan* 9	#6
2	*ẹwa* 10	#19
3	*ogun* 20	#24
4	*ogun* $\beta = \beta \times 20$ $(2 \leq \beta \leq 9)$	##24–25
5	*ẹwa din ogun* $\beta = \beta \times 20 - 10$ $(3 \leq \beta \leq 9)$	##23–25
6	α *din* $\gamma = \gamma - \alpha$ $(1 \leq \alpha \leq 5; \gamma = k \cdot 10, 4 \leq k \leq 19)$	##16–17
7	α *l-*$\gamma = \gamma + \alpha$ $(1 \leq \alpha \leq 4; \gamma = k \cdot 10, 1 \leq k \leq 19)$	##16–17

Table 4: Rules for Yoruba

The TeX macros for generating these numerals in Yoruba and an excerpt from the text of the problem produced with their use (as LaTeX source and typeset) are shown in Figures 4 and 5, respectively.

```
1  \newcommand \yorn [1]{%
2      \ifcase #1\or\d okan\or eji\or\d eta\or\d erin\or arun \or\d efa
3      \or eje\or\d ej\d o\or\d esan\fi}
4  \newcount \scor \newcount \tens \newcount \ones \newcount \absones
5  \newcommand \yorr [1]{%
6      \ifnum #1<10 \yorn #1%
7      \else
8          \tens =#1\divide \tens by 10
9          \ones =-\tens \multiply \ones by 10\advance \ones by #1
10         \ifnum 4<\ones
11             \advance \tens by 1\advance \ones by-10%
12             \absones =0\advance \absones by-\ones
13         \else \absones =\ones
14         \fi
15         \ifnum 0=\ones
16         \else \yorn \absones \space
17             \ifnum 0<\ones l-\else din \fi
18         \fi
19         \ifnum 1=\tens \d ewa%
20         \else
21             \scor =\tens \divide \scor by 2%
22             \ones =-\scor \multiply \ones by 2\advance \ones by \tens
23             \ifnum 1=\ones \advance \scor by 1\d ewa din\fi
24             ogun%
25             \ifnum 1<\scor \space \yorn \scor \fi
26         \fi
27     \fi
28 }
```

Figure 4: Macro definitions for generating Yoruba numerals.

[1]The forms given here are in fact reconstructions which reveal the internal structure of the numerals but conceal the complex morphophonological processes which produce the surface forms of the contemporary living language.

```
\newcommand \yorl [1]{$#1$ & \textit {}\yorr{#1}}}

\begin{tabular}{rl}
\yorl 3 \\
\yorl{11} \\
\yorl{22} \\
\yorl{37} \\
\yorl{66} \\
\yorl{93} \\
\yorl{135}
\end{tabular}
...
\item[(a)]Identify the numbers: \yorr{144}; \yorr{45}.
```

3	*ẹta*
11	*ọkan l-ẹwa*
22	*eji l-ogun*
37	*ẹta din ogun eji*
66	*ẹrin din ẹwa din ogun ẹrin*
93	*ẹta l-ẹwa din ogun arun*
135	*arun din ogun eje*

(a) Identify the numbers: *ẹrin l-ogun eje*; *arun din ẹwa din ogun ẹta.*

Figure 5: An excerpt from the statement of the Yoruba problem.

5.4. Some other noteworthy issues

The other occasions in which the method has been used for writing number names in linguistic problems will not be considered in detail here, for want of space, but a few notes on various interesting issues that come up are in order.

Generating large numerals in a non-decimal system is error-prone. The Arammba number system is base-6 and goes up to $6^7 = 279\,936$, so there is much to be gained by leaving the number crunching to the computer. This passage encodes the fact that a number greater than or equal to $6^5 = 7776$ (and presumed less than $2 \times 6^5 = 15552$ because of the parameters of the linguistic problem for whose typesetting the macros were composed) is named *weremeke* '6^5' followed by the difference:

```
\ifnum 7775<\numb
    \ifstarted \space \fi
    weremeke\advance \numb by -7776
    \startedtrue
\fi
```

Likewise for the lower degrees of 6, with coefficients where necessary. The same technique is applied, in the text of the same linguistic problem, for the base-20 system of Nahuatl (if the number is greater than 7999, then 8000 is subtracted, etc.).

Umbu-Ungu is base-24, with 4 as a secondary base, but has special (unanalysable) names for all multiples of 4 up to 32, which means that, although 48 is expressed as 24×2 *tokapu talu*, the following two fours are $52 = 24 + 28$ *tokapu alapu* and $56 = 24 + 32$ *tokapu polangipu*, and 24×2 only comes up in $60 = 24 \times 2 + 12$ *tokapu talu rurepo*. The macro `\tuu{`k`}` (where $3 \leq k \leq 26$; k is not divisible by 6) generates the name of the kth multiple of 4.

```
\newcommand \tuu [1]{\uux=#1\relax
  \ifnum 20<\uux tokapu yepoko \advance\uux by-18
  \else \ifnum 14<\uux tokapu talu \advance\uux by-12
  \else \ifnum 8<\uux tokapu \advance\uux by-6
  \fi\fi\fi
  \advance\uux by-2
  \ifcase \uux \or rurepo\or malapu\or
    supu\or tokapu\or alapu\or polangipu\fi}
```

Also the language uses overcounting, so 57 is *tokapu talu rurepo·nga telu* $24 \times 2 + 12 \neg 1 = 60 \neg 1$ ('1 from the 4 that completes 60'). This is implemented by checking if the number of ones is zero, and if not, adding 1 to the number of fours before generating their name.

```
\ones=#1
\uuy=#1\divide \uuy by4\fours=\uuy
\multiply \uuy by4\advance \ones by-\uuy
\ifnum 0=\ones \tuu {\fours}%
\else \advance \fours by1\tuu {\fours}nga
    \ifcase \ones \or telu\or talu\or yepoko\fi%
\fi
```

Drehu, which has a vigesimal system but uses 5, 10 and 15 as supplementary bases, calls these three numbers β*-pi*, where β is the quantity of fives, but has a suffix for each of them when a number of the range $[1; 4]$ is to be added: 15 is *köni·pi* 3×5 but 18 is *köni·qaihano* $3 + 15$.

```
\ifnum 0=\ones
    \Drehun \fems pi%
\else \ifcase \fems \Drehun \ones
    \or \Drehun \ones ngömen%
    \or \ifcase \ones \or caa\or lua\or köni\or eka\fi ko%
    \or \Drehun \ones qaihano\fi
```

The macro `\Drehun` produces the numbers from 1 to 4; the forms they assume before the suffix *-ko* '$+ 10$' are simply listed because of various opaque morphophonological changes.

Generating whole noun phrases containing numerals as quantifiers can present additional challenges. The Sulka language has three number systems (for counting coconuts, breadfruit, and everything else). Some of the nouns have suppletive singular and plural forms (e. g., sg. *tu*, pl. *sngu* 'yam'). There is also a dual number (marked by *lo* preposed to the singular), although it does not preclude the use of a numeral. So generating an expression combining a noun and a number involves choosing the appropriate system as well as putting the noun in the appropriate grammatical form (*a tu a tgiang* '1 yam', *a lo tu a lomin* '2 yams', *o sngu a korlotge* '3 yams').

6. Conclusions

It is hoped that this brief exposition has sufficed to demonstrate both the advantages of leaving the construction of complex number names to the computer whilst creating a text – in essence, a minor exercise in automatic natural language generation – and the difficulties one may encounter when doing so. The last example that was mentioned here touched upon the possibility of expanding the method beyond the numeral, which hints at the great potential of the approach.

References

Derzhanski, I. and Veneva, M. (2018). Linguistic Problems on Number Names. In *Proceedings of the Third International Conference Computational Linguistics in Bulgaria*, pages 169–176. `https://dcl.bas.bg/clib/wp-content/uploads/2018/07/CLIB_2018_Proceedings_v2_final.pdf`.

Derzhanski, I. (2009). *Linguistic Magic and Mystery*. Sofia: Union of Bulgarian Mathematics.

Derzhanski, I. (2013). Multilingual Editing of Linguistic Problems. In *Proceedings of the Fourth Workshop on Teaching NLP and CL*, pages 27–34.

Dreyfus, M. and Gangloff, C. (1975). *La pratique du Fortran: Exercises commentés*. Paris: Dunod.

Hanke, T. (2005). *Sum: Overcounting in Numerals*. *LINGUIST List* **16**.2448. `https://linguistlist.org/issues/16/16-2448.html`.

Knuth, D. E. (1986). *The TeXbook*. Addison-Wesley Professional.

LaTeX 2_ε. (2018). *LaTeX2e Unofficial Reference Manual*. `http://tug.org/texinfohtml/latex2e.html`.

Menninger, K. (1969). *Number Words and Number Symbols: A Cultural History of Numbers*. Cambridge: MIT Press.

Todorova, M., Armyanov, P., Petkova, D., and Georgiev, K. (2008). *Sbornik ot zadachi po programirane na C++: Chast 1*. Sofia: Tehnologika.