Handwriting fonts, METAFONT and OpenType

Karel Píška

Institute of Physics, Academy of Sciences
Prague, Czech Republic

4 September 2009

EuroTEX 2009 NLDA: NDC/IDL, Den Haag, 31 Aug. – 4 Sep. 2009

Introduction

In my presentation under **handwriting** we will understand a *writing system* taught in primary schools in the Czech Republic, Armenia and Georgia, approx. in the last 20–40 years.

Two educational aspects in my presentation:

- *writing* is closely connected with primary school education
- my comparative study of "advanced" typography with METAFONT(and TEX) and OpenType may be considered educational

i.e. features of school handwriting and corresponding solutions with computer typography.

Introduction (cont.)

I will demonstrate

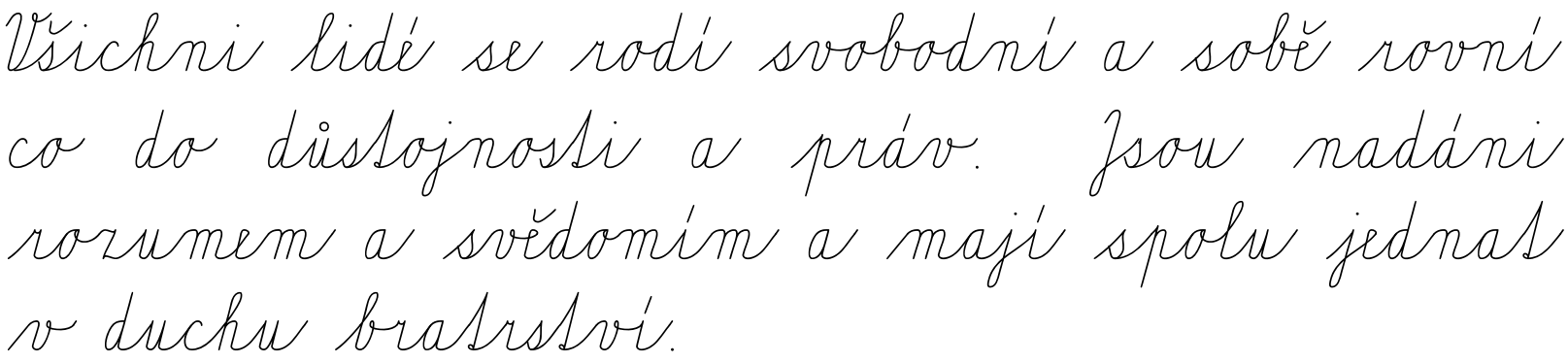
- traditional principles and peculiarities of contemporary Czech, Georgian and Armenian handwriting
- a short general introduction to "advanced" typography with METAFONT and OpenType
- examples of computer realization for handwriting fonts
- summary of my experiences

The Czech handwriting font `slabikar` was designed and created in METAFONT by Petr Olšák.

Contents

Contents (cont.)

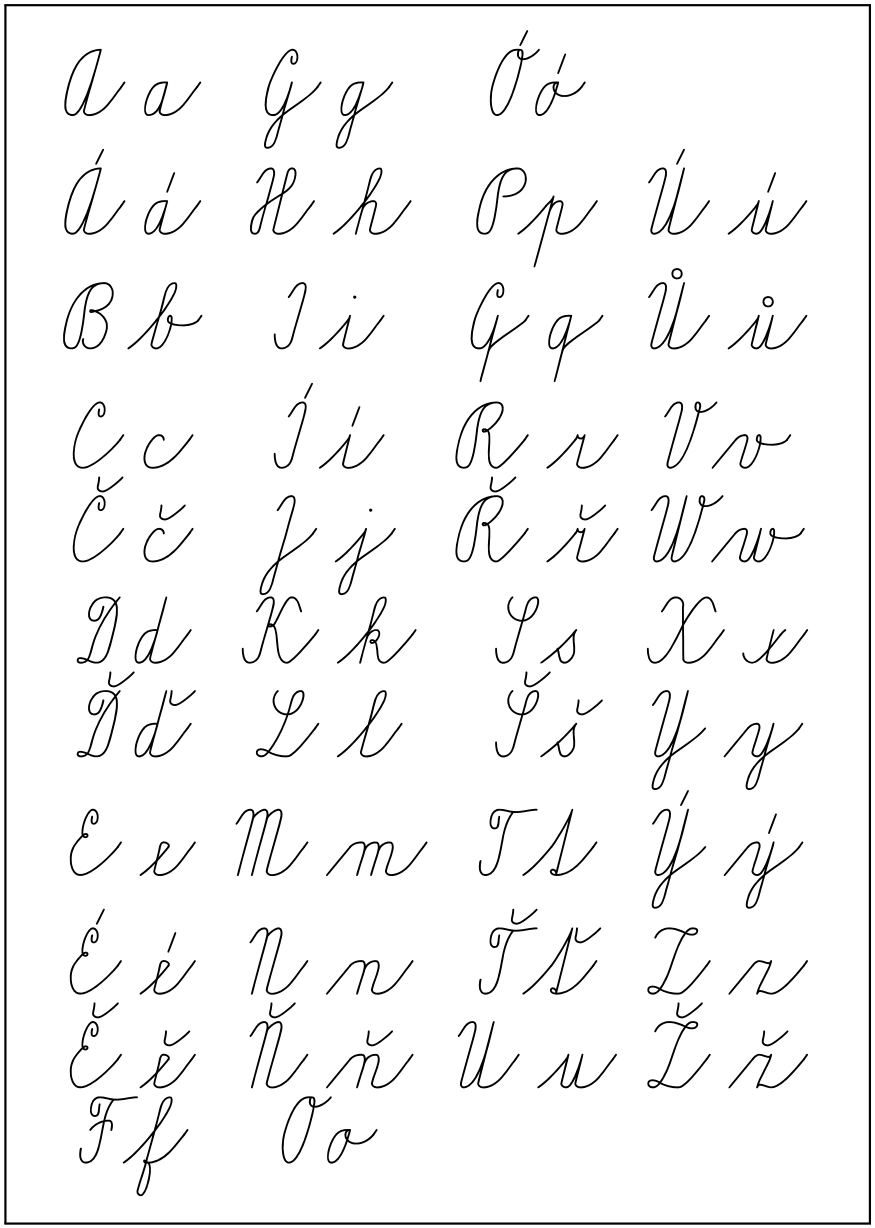
Handwritten scripts – Czech handwriting



[from Universal Declaration of Human rights]

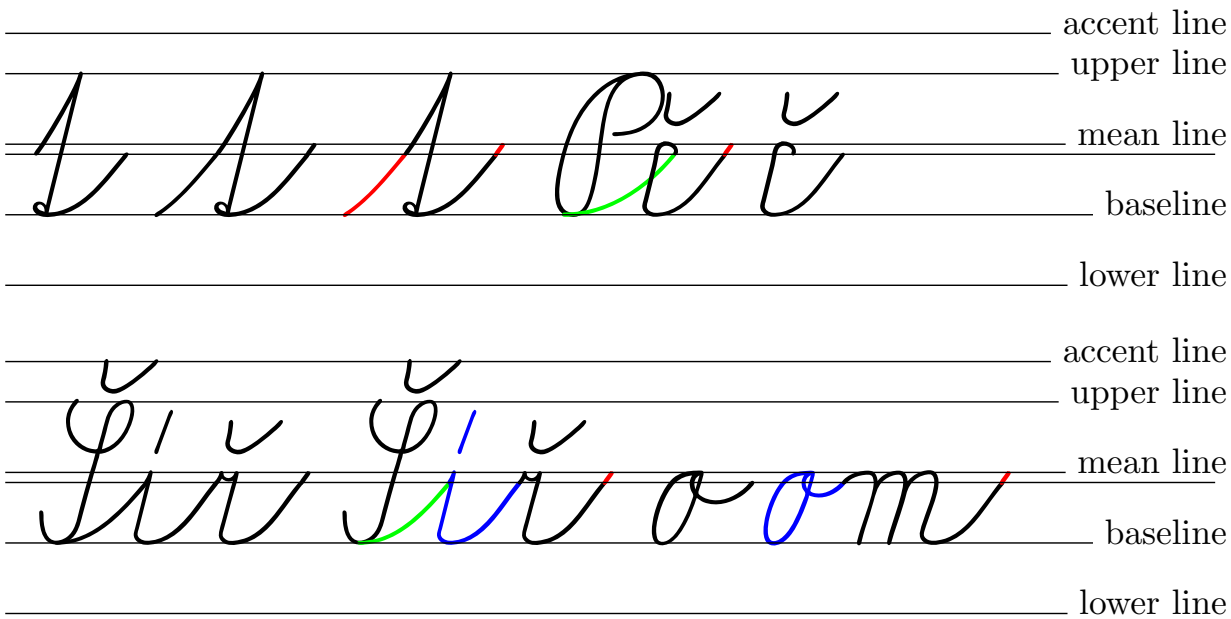
Today's "ordinary" Czech school handwriting has far to *calligraphy* and also to a special school subject called *penmanship* [krasopis] taught earlier and disappearing tens years ago. In the 19th century in Austria-Hungary and before typewriters a nice hand-writing was very important, for example, in state administration, offices, etc. Characteristics of Czech handwriting may differ from Britain, France, etc. using also Latin script (majuscule and minuscule are common): slanted; all letters in the word connected together, written on one go.

Czech handwriting alphabet



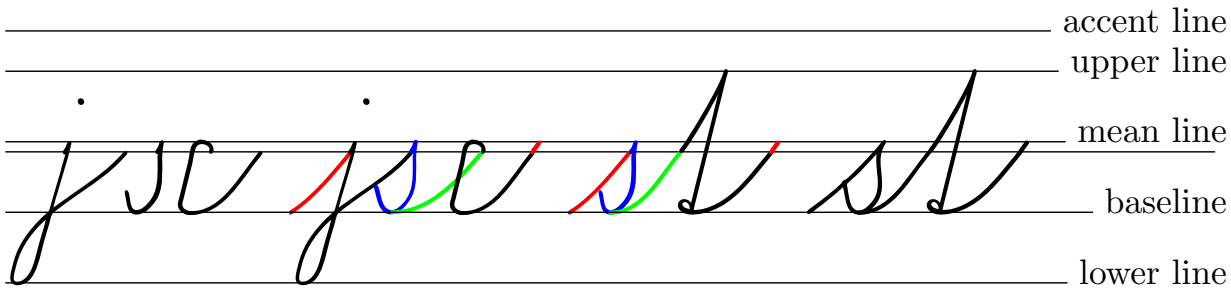
Czech handwriting (cont.)

I will show some “special” conventions in examples.



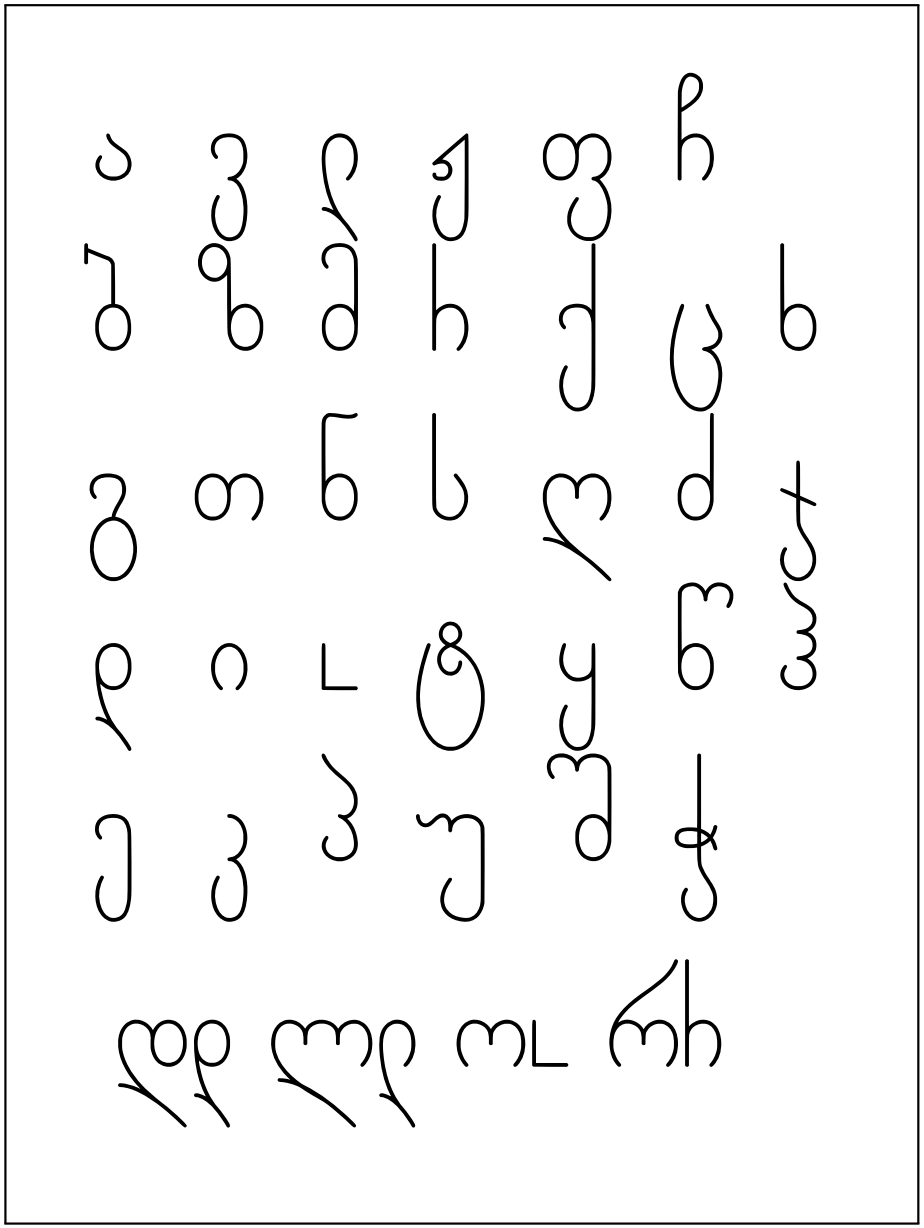
The Czech letters (Latin letters without and with accents) in the fonts corresponding to Unicode numbers are in the *medial* form and then may be contextually modified.

Czech handwriting (cont.)



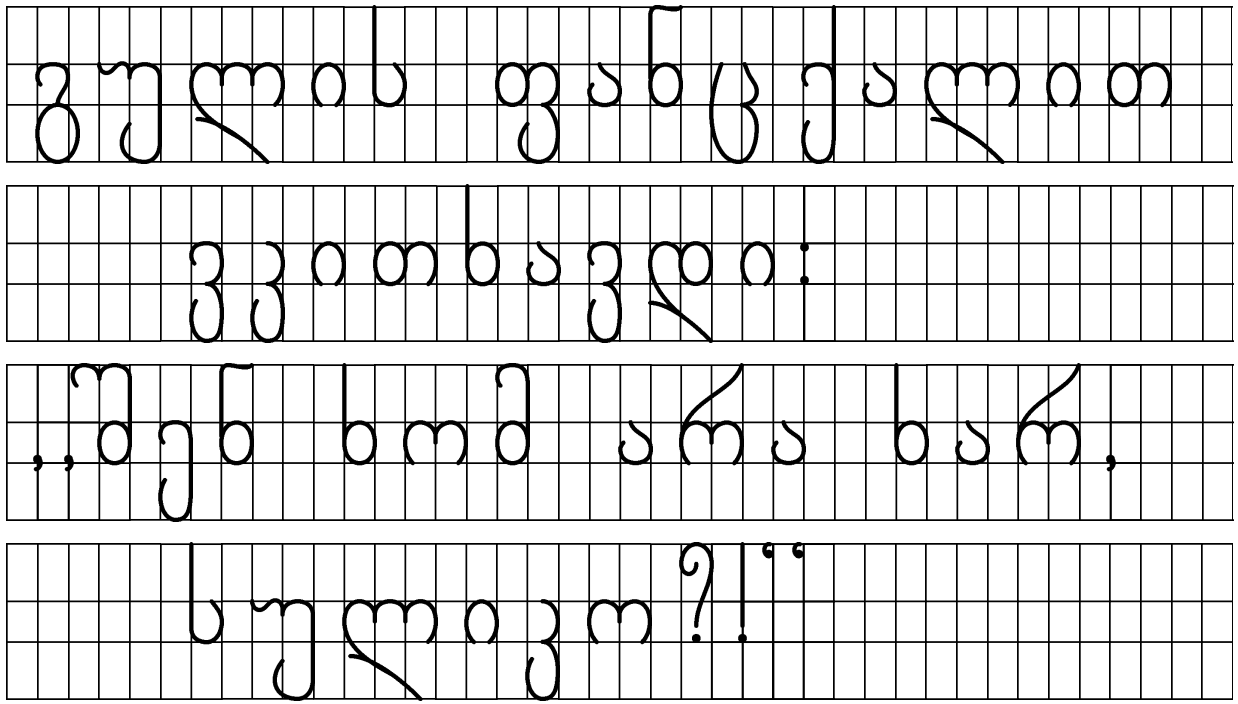
3 different “s”

Georgian handwriting alphabet



Georgian handwriting

Modern Georgian script (contemporary Mkhedruli) does not distinguish capital and small letters; (taught) handwriting is traditionally upright.



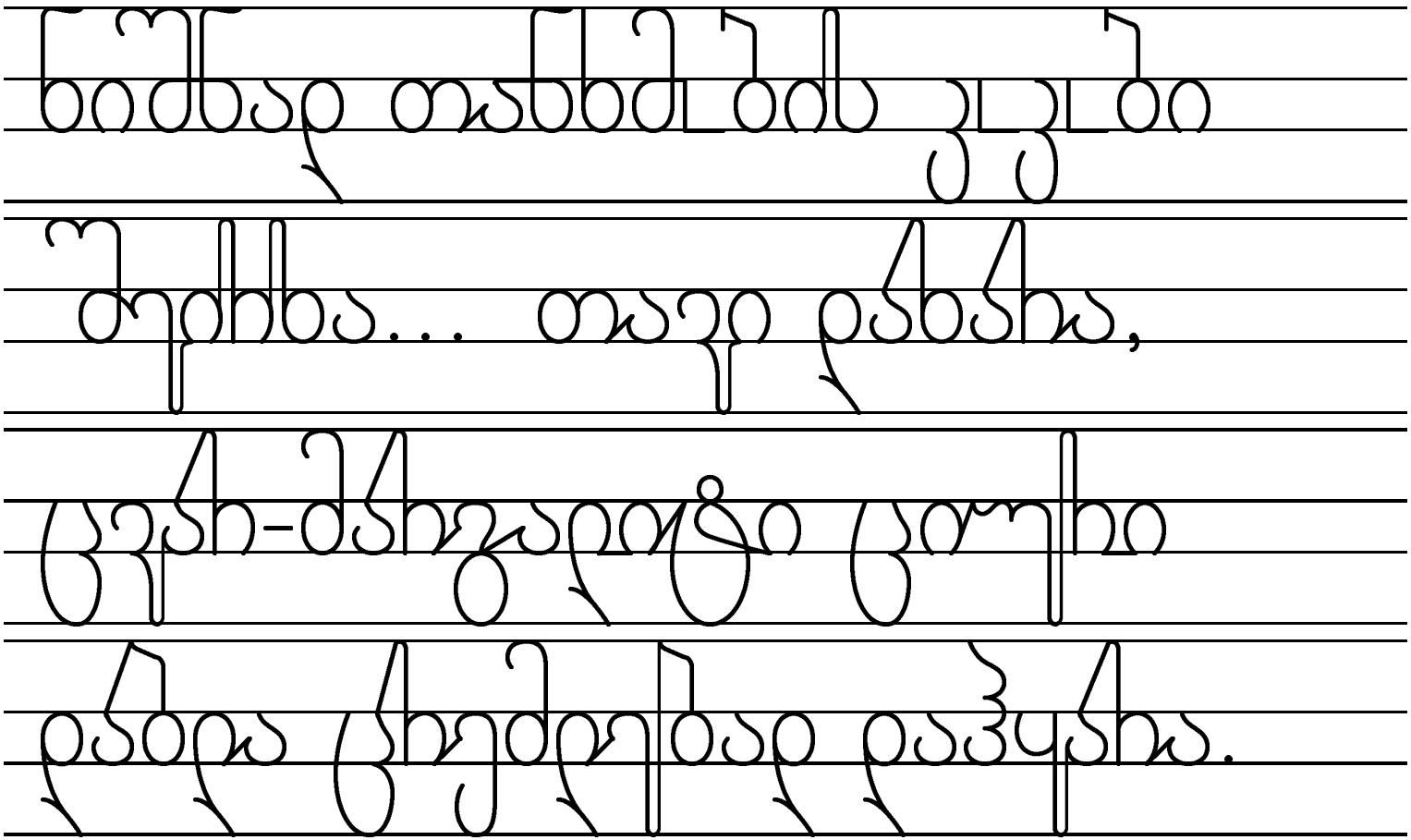
პირველი კლასი

(The first year)

In the font, the primary glyph form is *isolated*.

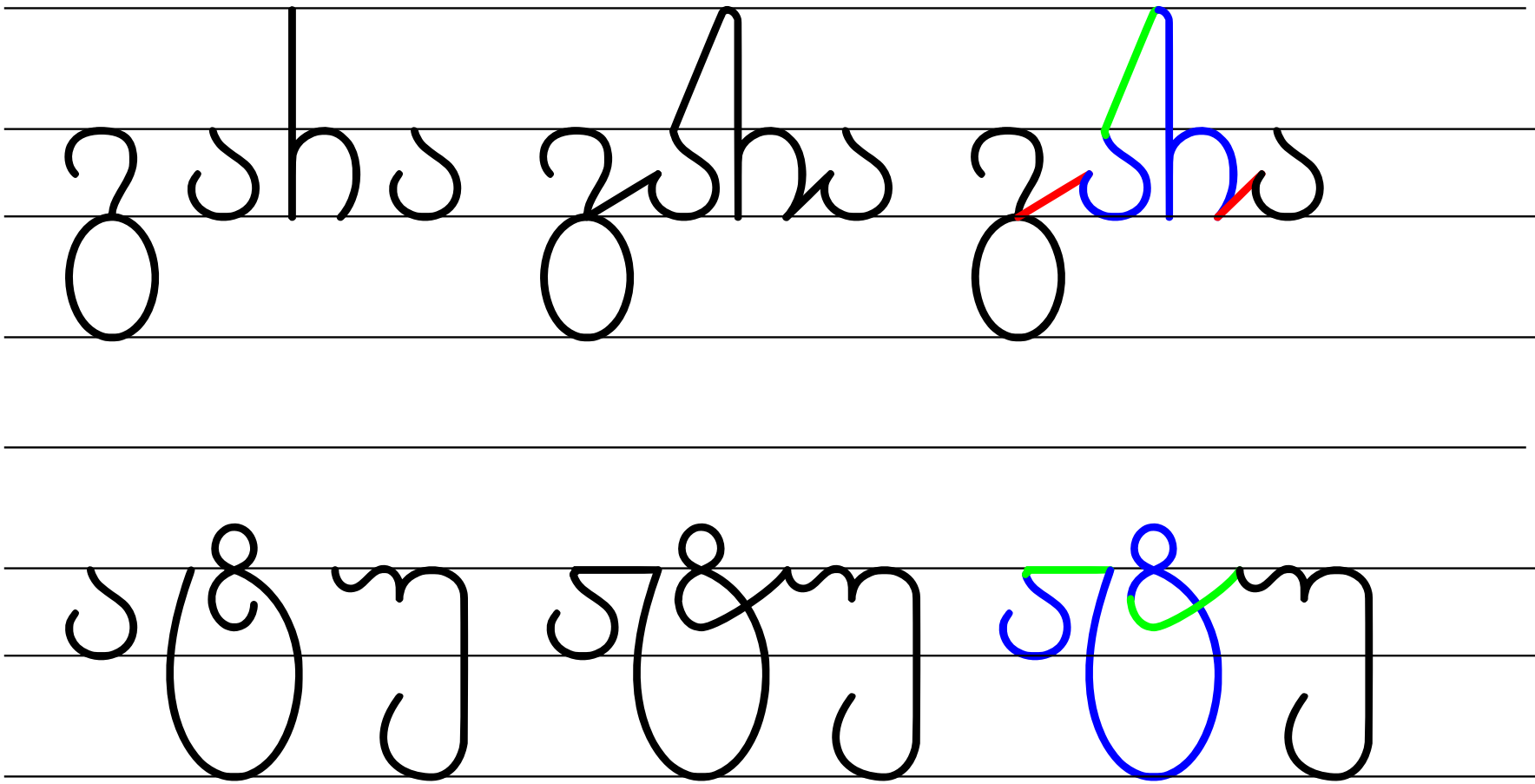
Georgian handwriting (cont.)

In advanced Georgian handwriting are letters in the word joined together, but not all the pairs of the adjacent letters.

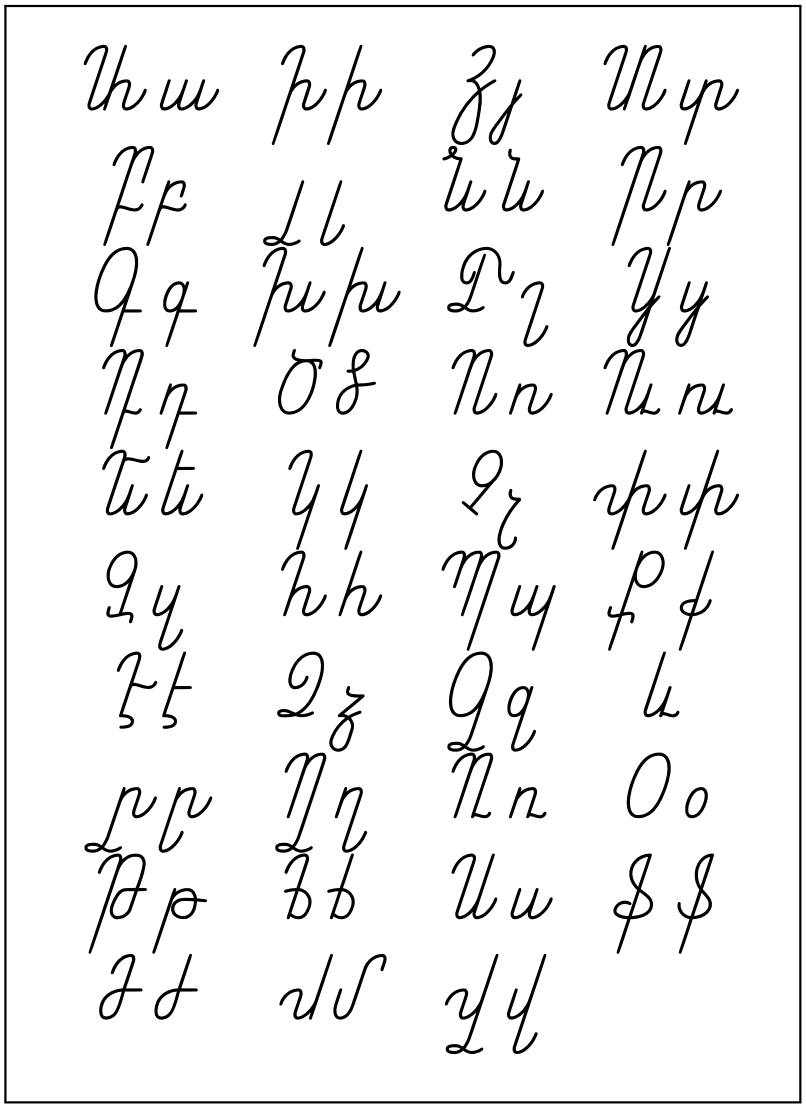


Georgian handwriting (cont.)

Depending of context, some letters may be joined be special strokes, some letters may be additionally modified.

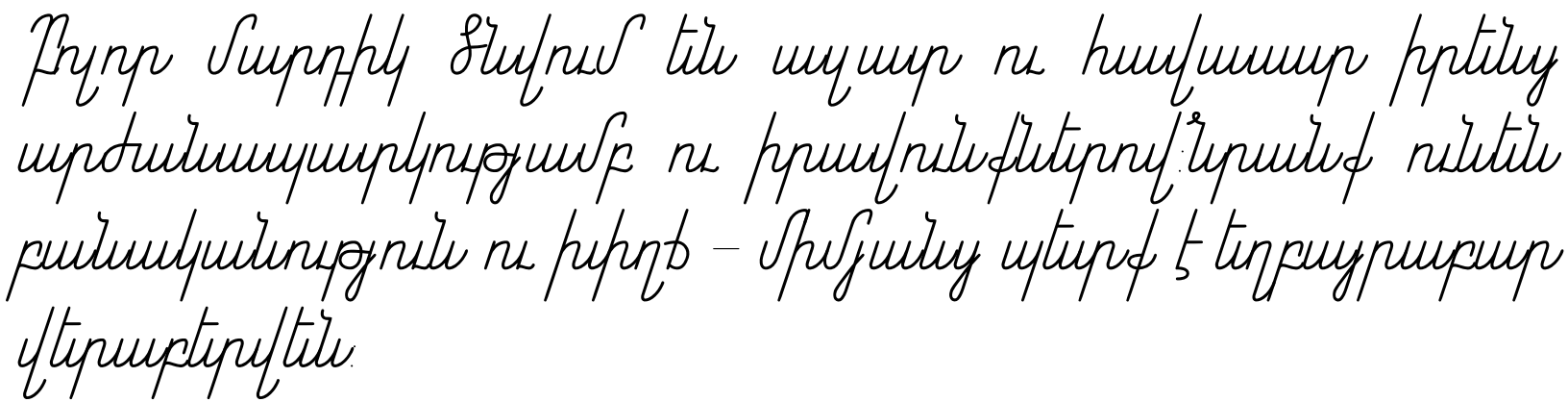


Armenian handwriting alphabet



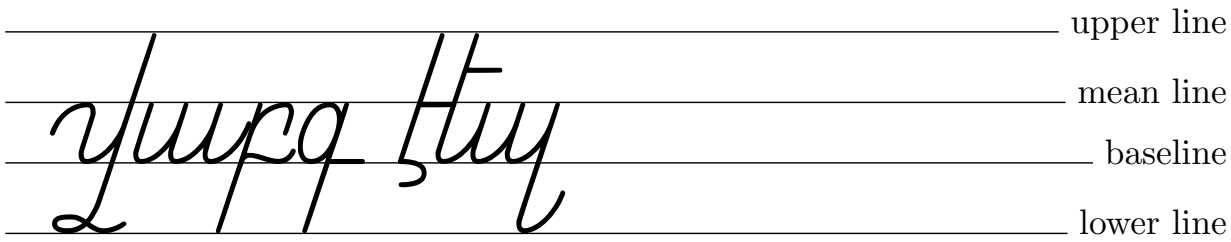
Armenian handwriting

Armenian (like a Latin-based) script distinguish capital and small letters; and handwriting is slanted. There are no special sophisticated joiners between letters, but we have another problem — to split words into letters — a solution could be gaps between letters.



(The Armenian part of my fonts has bugs and has not been finished.)

Armenian handwriting (cont.)



Advanced typography with METAFONT and TEX

METAFONT contains powerful tools, exploited in METAFONT fonts rarely.
"Advanced" (generalized) ligatures in METAFONT

```
ligtable          % produces
 a: b  |=:|    c; % acb
 a: b  |=:|>   c; % acb
 a: b  |=:|>>  c; % acb
 a: b   =:|    c; % cb
 a: b   =:|>   c; % cb
 a: b  |=:     c; % ac
 a: b  |=:>    c; % ac
 a: b   =:     c; % c
```

Advanced typography with METAFONT and TEX(cont.)

"Advanced" (generalized) ligatures in METAFONT

```
.mf          .tfm/.pl
---------------
```

`|=:|` `/LIG/` % retains both *a* and *b*, inserts *c* between: *acb*

`|=:|>` `/LIG/>` % retains both *a* and *b*, inserts *c* between; the processing continues after *a*: *acb*

`|=:|>>` `/LIG/>>` % retains both *a* and *b*, inserts *c* between; the processing continues after *c*: *acb*

`=:|` `LIG/` % retains *b*, inserts *c* before *b*: *cb*

`=:|>` `LIG/>` % retains *b*, inserts *c* before *b*; the processing continues after *c*: *cb*

`|=:` `/LIG` % retains *a*, inserts *c* after *a*: *ac*

`|=:>` `/LIG>` % retains *a*, inserts *c* after *a*; the processing continues after *a*: *ac*

`=:` `LIG` % substitutes both *a* and *b* by *c*.

Advanced typography with METAFONT and TEX(cont.)

Boundary characters

The METAFONT and TEX concept of the "word boundary" (the left and right boundary characters) allows to "implicit" processing of the beginning and the end of the word, i.e. a substitution or adjustment of the letters in the "initial" and the "final" position of the word. In METAFONT sources the left boundary characters is denoted by `"||:"`, the right boundary character must be introduced as the "real" character using the `"boundarychar` *code*`";"` assignment.

Advanced typography with OpenType

"Old TrueType fonts" can be "enriched" by adding "Advanced OpenType Typographic Tables" to produce fonts in OpenType format. (Now we will not discuss two different format versions: "new" TTF and OTF, because the "Advanced . . . tables" are common.)
OpenType introduces substitution (GSUB), positioning (GPOS), and several other tables.
These tables define the set of rules of several types specifying [from OpenType specification]:
Glyph substitution (GSUB) rules
Single substitution, Multiple substitution, Alternate substitution, Ligature substitution, Contextual substitution, Chaining contextual substitution (with specifying a Chain Sub rule and marking sub-runs and specifying exceptions to the Chain Sub rule), Extension substitution, Reverse Chaining Single Substitution;

Advanced typography with OpenType (cont.)

Glyph positioning (GPOS) rules

Single adjustment positioning, Pair adjustment positioning (with Specific and class pair kerning, Enumerating pairs, and Subtable breaks), Cursive attachment positioning, Mark-to-Base attachment positioning, Mark-to-Ligature attachment positioning, Mark-to-Mark attachment positioning, Contextual positioning, Chaining contextual positioning (with Specifying a Chain Pos rule and marking sub-runs, Specifying Contextual Positioning with explicit lookup references, Specifying Contextual Positioning with with in-line single positioning rules, Specifying Contextual Positioning with with in-line cursive positioning rules, Specifying Contextual Positioning with with in-line in-line mark attachment positioning rules, and Specifying exceptions to the Chain Pos rule), Extension positioning.

Advanced typography with OpenType (cont.)

Each *feature* is defined as a system of subsystems called *lookups*. Any *lookup* is described as a subsystem consisted of substitution and positioning rules. Depending on *script* and *language*, a *feature* may be enabled or disabled. If the *feature* is enabled and some *lookup*, contained in this *feature*, fulfil the given conditions, then the execution of corresponding operations should be invoked. It is a signal and the real application must be executed by an application program or operating system, e.g. by means of a special library. At first, we have to create an OpenType font properly, using some suitable tools. And, at second, the font must be in agreement with corresponding software to execute adequate operations according the rules (instruction) defined in the font.

Tools for OpenType tables

- **VOLT** [Microsoft]
  Visual OpenType Layout Tool
  VOLT adds OpenType tables and proofs the features and lookups; accepts only the fonts with OpenType tables produced by VOLT, other OpenType tables deletes. Moreover, before the tests in the proofing window we must run always (re)compilation even for opened fonts from VOLT.
- **FontForge** [G. Williams]
- **AFDKO** [Adobe]
  Adobe Font Development Kit for OpenType
- (**FontLab Studio** [FontLab] — commercial, I do not have it)

Comment
I did not investigate **AAT tables** (Apple Advanced Typography).

Substitutions – Simple substitutions

We can skip such non-complex substitutions; only several small examples. . .
Non-contextual substitutions
In METAFONT&TEX/LATEX usually solved by macros (to switch fonts, typeset math, . . .)
Single non-contextual substitutions
LATEX macro OpenType feature

`\textsc{}`	`smcp`
`\oldstylenums{}`	`onum`

SmallCaps
0123456789
Multiple non-contextual substitutions
TEX and its children have powerful facilities to typeset math and can exist without delegations of similar actions on fonts
`\frac{}{}` $\frac{a}{b}$; the OpenType feature is also `frac`

Complex substitutions - contextual in METAFONT

```
ligtable "b":
     "m" =:| bnarrow,
     "n" =:| bnarrow,  ncaron =:| bnarrow,
     "v" =:| bnarrow,  "w" =:| bnarrow,
     "y" =:| bnarrow,  yacute =:| bnarrow,
     rightboundaries;
ligtable "o":
     "m" =:| onarrow,
     "n" =:| onarrow,  ncaron =:| onarrow,
     "v" =:| onarrow,  "w" =:| onarrow,
     "y" =:| onarrow,  yacute =:| onarrow,
     rightboundaries;
ligtable oacute:
     "m" =:| oacutenarrow,
     "n" =:| oacutenarrow,  ncaron =:| oacutenarrow,
     "v" =:| oacutenarrow,  "w" =:| oacutenarrow,
     "y" =:| oacutenarrow,  yacute =:| oacutenarrow,
     rightboundaries;
ligtable "v":
     "m" =:| vnarrow,
     "n" =:| vnarrow,  ncaron =:| vnarrow,
     "v" =:| vnarrow,  "w" =:| vnarrow,
     "y" =:| vnarrow,  yacute =:| vnarrow,
     rightboundaries;
ligtable "w":
     "m" =:| wnarrow,
     "n" =:| wnarrow,  ncaron =:| wnarrow,
     "v" =:| wnarrow,  "w" =:| wnarrow,
     "y" =:| wnarrow,  yacute =:| wnarrow,
     rightboundaries;
```

Complex substitutions - contextual in VOLT

METAFONT–VOLT—FEA

The example with narrower b,o,v,w

Rules structured and ordered in significantly different ways.

```
DEF_LOOKUP "CZEbmnvwy" PROCESS_BASE PROCESS_MARKS ALL DIRECTION LTR
IN_CONTEXT
 RIGHT ENUM GLYPH "m" GLYPH "n" GLYPH "ncaron" GLYPH "v" GLYPH "w"
 GLYPH "y" GLYPH "yacute" END_ENUM
END_CONTEXT
AS_SUBSTITUTION
SUB GLYPH "b" WITH GLYPH "bnarrow" END_SUB
SUB GLYPH "o" WITH GLYPH "onarrow" END_SUB
SUB GLYPH "oacute" WITH GLYPH "oacutenarrow" END_SUB
SUB GLYPH "v" WITH GLYPH "vnarrow" END_SUB
SUB GLYPH "w" WITH GLYPH "wnarrow" END_SUB
END_SUBSTITUTION
```

Complex substitutions - contextual in VOLT

```
DEF_SCRIPT NAME "Latin" TAG "latn"

DEF_LANGSYS TAG "CZE "

DEF_FEATURE NAME "Standard Ligature Set 1" TAG "liga"
 LOOKUP "CZEliga"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 1" TAG "ss01"
 LOOKUP "CZEjoinc" LOOKUP "CZEjoinl" LOOKUP "CZEjoinc_s" LOOKUP "CZEjoins_s" LOOKUP "CZEbmnvwy"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 2" TAG "ss02"
 LOOKUP "CZEbmnvwy"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 3" TAG "ss03"
 LOOKUP "CZEgjqy"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 4" TAG "ss04"
 LOOKUP "CZErbound"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 5" TAG "ss05"
 LOOKUP "CZEbeg"
END_FEATURE
DEF_FEATURE NAME "Stylistic Set 6" TAG "ss06"
 LOOKUP "CZEbegpos"
END_FEATURE
END_LANGSYS
DEF_LANGSYS NAME "Default" TAG "dflt"

END_LANGSYS
END_SCRIPT
```

Complex substitutions - contextual in FEA

```
@CZEbmnvwy = [ m n ncaron v w y yacute ];
lookup CZEbmnvwy {
 sub b’ @CZEbmnvwy by bnarrow;
 sub o’ @CZEbmnvwy by onarrow;
 sub oacute’ @CZEbmnvwy by oacutenarrow;
 sub v’ @CZEbmnvwy by vnarrow;
 sub w’ @CZEbmnvwy by wnarrow;
} CZEbmnvwy;
feature ss02 { # "Stylistic Set 2"
 script latn;
 language dflt;
 lookup CZEjoincini;
 lookup CZEjoinP;
 lookup CZEjoinc_sc;
 lookup CZEjoins_ss;
 lookup CZEbmnvwy;
} ss02;

feature ss03 { # "Stylistic Set 3"
 script latn;
 language dflt;
 lookup CZEjoinc;
 lookup CZEjoinl;
 lookup CZEjoinc_s;
 lookup CZEjoins_s;
 lookup CZEbmnvwy;
```

Complex substitutions – insertion with METAFONT

```
ligtable  ||: clqq: dash: slash: "(": "[": "&": "+": leftboundary::
      "a" kern kk#,  aacute kern kk#,
      "b" |=:|> 3,
      "c" kern kk#, ccaron kern kk#,
      "d" kern kk#, dcaron kern kk#,
      "e" |=:|> 3, eacute |=:|> 3, ecaron |=:|> 3,
      "f" |=:|> 3,
      "g" kern kk#,
      "h" |=:|> 3,
      "i" |=:|> 3, iacute |=:|> 3,
      "j" |=:|> 3,
      "k" |=:|> 3,
      "l" |=:|> 3,
      "m" |=:|> 2,
      "n" |=:|> 2,  ncaron |=:|> 2,
      "o" kern kk#,  oacute kern kk#,
      "p" |=:|> 3,  "q" kern kk#,
      "r" |=:|> 3,  rcaron |=:|> 3,
      "s" |=:> sleft,  scaron |=:> scaronleft,
      "u" |=:|> 3,  uring |=:|> 3, uacute |=:|> 3,
      "t" |=:|> 3,  tcaron |=:|> 3,
      "v" |=:|> 2,  "w" |=:|> 2,
      "x" |=:|> 7,
      "y" |=:|> 2,  yacute |=:|> 2,
      "z" |=:|> 2,  zcaron |=:|> 2;
```

Complex substitutions – insertion with VOLT

```
DEF_GROUP "acccap"
 ENUM GLYPH "Aacute" GLYPH "Ccaron" GLYPH "Dcaron" GLYPH "Eacute" G
 GLYPH "Iacute" GLYPH "Ncaron" GLYPH "Oacute" GLYPH "Rcaron" GLYPH
 GLYPH "Tcaron" GLYPH "Uacute" GLYPH "Uring" GLYPH "Yacute" GLYPH '
END_GROUP
DEF_GROUP "accver"
 ENUM GLYPH "aacute" GLYPH "ccaron" GLYPH "dcaron" GLYPH "eacute" G
 GLYPH "iacute" GLYPH "ncaron" GLYPH "oacute" GLYPH "rcaron" GLYPH
 GLYPH "tcaron" GLYPH "uacute" GLYPH "uring" GLYPH "yacute" GLYPH '
END_GROUP
DEF_GROUP "czebeg"
 ENUM GROUP "czelet" GROUP "czemod" GLYPH "joinc" GLYPH "joinl" GLY
END_GROUP
DEF_GROUP "czelet"
 ENUM RANGE "A" TO "Z" GROUP "acccap" GROUP "czever" END_ENUM
END_GROUP
DEF_GROUP "czemid"
 ENUM GROUP "czever" GROUP "czemod" END_ENUM
END_GROUP
DEF_GROUP "czemod"
 ENUM GLYPH "bnarrow" GLYPH "vnarrow" GLYPH "wnarrow"
 GLYPH "onarrow" GLYPH "oacutenarrow" END_ENUM
END_GROUP
DEF_GROUP "czever"
 ENUM RANGE "a" TO "z" GROUP "accver" END_ENUM
END_GROUP
```

Complex substitutions – insertion with VOLT

```
DEF_LOOKUP "CZEbeg" PROCESS_BASE PROCESS_MARKS ALL DIRECTION LTR
EXCEPT_CONTEXT
 LEFT GROUP "czebeg"
END_CONTEXT
AS_SUBSTITUTION
SUB GLYPH "b" WITH GLYPH "lbounda" GLYPH "b" END_SUB
SUB GLYPH "bnarrow" WITH GLYPH "lbounda" GLYPH "bnarrow" END_SUB
SUB GLYPH "e" WITH GLYPH "lbounda" GLYPH "e" END_SUB
SUB GLYPH "eacute" WITH GLYPH "lbounda" GLYPH "eacute" END_SUB
SUB GLYPH "ecaron" WITH GLYPH "lbounda" GLYPH "ecaron" END_SUB
SUB GLYPH "f" WITH GLYPH "lbounda" GLYPH "f" END_SUB
SUB GLYPH "h" WITH GLYPH "lbounda" GLYPH "h" END_SUB
SUB GLYPH "i" WITH GLYPH "lbounda" GLYPH "i" END_SUB
SUB GLYPH "iacute" WITH GLYPH "lbounda" GLYPH "iacute" END_SUB
SUB GLYPH "j" WITH GLYPH "lbounda" GLYPH "j" END_SUB
SUB GLYPH "k" WITH GLYPH "lbounda" GLYPH "k" END_SUB
SUB GLYPH "l" WITH GLYPH "lbounda" GLYPH "l" END_SUB
SUB GLYPH "m" WITH GLYPH "lbound" GLYPH "m" END_SUB
SUB GLYPH "n" WITH GLYPH "lbound" GLYPH "n" END_SUB
SUB GLYPH "ncaron" WITH GLYPH "lbound" GLYPH "ncaron" END_SUB
SUB GLYPH "p" WITH GLYPH "lbounda" GLYPH "p" END_SUB
SUB GLYPH "r" WITH GLYPH "lbounda" GLYPH "r" END_SUB
SUB GLYPH "rcaron" WITH GLYPH "lbounda" GLYPH "rcaron" END_SUB
SUB GLYPH "s" WITH GLYPH "sleft" END_SUB
SUB GLYPH "scaron" WITH GLYPH "scaronleft" END_SUB
SUB GLYPH "t" WITH GLYPH "lbounda" GLYPH "t" END_SUB
SUB GLYPH "tcaron" WITH GLYPH "lbounda" GLYPH "tcaron" END_SUB
SUB GLYPH "u" WITH GLYPH "lbounda" GLYPH "u" END_SUB
SUB GLYPH "uring" WITH GLYPH "lbounda" GLYPH "uring" END_SUB
SUB GLYPH "uacute" WITH GLYPH "lbounda" GLYPH "uacute" END_SUB
SUB GLYPH "v" WITH GLYPH "lbound" GLYPH "v" END_SUB
SUB GLYPH "vnarrow" WITH GLYPH "lbound" GLYPH "vnarrow" END_SUB
SUB GLYPH "w" WITH GLYPH "lbound" GLYPH "w" END_SUB
SUB GLYPH "wnarrow" WITH GLYPH "lbound" GLYPH "wnarrow" END_SUB
SUB GLYPH "x" WITH GLYPH "joinx" GLYPH "x" END_SUB
SUB GLYPH "y" WITH GLYPH "lbound" GLYPH "y" END_SUB
SUB GLYPH "yacute" WITH GLYPH "lbound" GLYPH "yacute" END_SUB
SUB GLYPH "z" WITH GLYPH "lbound" GLYPH "z" END_SUB
SUB GLYPH "zcaron" WITH GLYPH "lbound" GLYPH "zcaron" END_SUB
END_SUBSTITUTION
```

Complex substitutions – insertion with FEA

```
@acccap = [ Aacute Ccaron Dcaron Eacute Ecaron Iacute Ncaron Oacute Rcaron Scaron Tcaron
 Uacute Uring Yacute Zcaron ];
@accver = [ aacute ccaron dcaron eacute ecaron iacute ncaron oacute rcaron scaron tcaron
 uacute uring yacute zcaron ];
@czever = [ a - z @accver ];
@czelet = [ A - Z @acccap @czever ];
@czemod = [ bnarrow vnarrow wnarrow onarrow oacutenarrow ];
@czemid = [ @czever @czemod ];
@czefin = [ a aacute A Aacute b bnarrow c ccaron C Ccaron d dcaron e eacute ecaron
 E Eacute Ecaron f g G h H i iacute j J k K l L m M n ncaron N Ncaron o oacute
 onarrow oacutenarrow p q Q r rcaron R Rcaron t tcaron u uring uacute U Uacute Uring
 v vnarrow w wnarrow x X y yacute z zcaron Z Zcaron ];
@czeini = [ b bnarrow e eacute ecaron f h i iacute j k l m n ncaron p r rcaron s scaron
 t tcaron u uring uacute v vnarrow w wnarrow x y yacute z zcaron ];
@czefintmp = [ a.fin aacute.fin A.fin Aacute.fin b.fin bnarrow.fin c.fin ccaron.fin
 C.fin Ccaron.fin d.fin dcaron.fin e.fin eacute.fin ecaron.fin E.fin Eacute.fin Ecaron.fin
 f.fin g.fin G.fin h.fin H.fin i.fin iacute.fin j.fin J.fin k.fin K.fin l.fin L.fin
 m.fin M.fin n.fin ncaron.fin N.fin Ncaron.fin o.fin oacute.fin onarrow.fin oacutenarrow.fin
 p.fin q.fin Q.fin  r.fin rcaron.fin R.fin Rcaron.fin t.fin tcaron.fin u.fin uring.fin uacute.fin
 U.fin Uacute.fin Uring.fin v.fin vnarrow.fin w.fin wnarrow.fin x.fin X.fin y.fin yacute.fin
 z.fin zcaron.fin Z.fin Zcaron.fin ];
@czeinitmp = [ b.ini bnarrow.ini e.ini eacute.ini ecaron.ini f.ini h.ini i.ini iacute.ini j.ini
 k.ini l.ini m.ini n.ini ncaron.ini p.ini r.ini rcaron.ini s.ini scaron.ini t.ini tcaron.ini u.ini
 uring.ini uacute.ini v.ini vnarrow.ini w.ini wnarrow.ini x.ini y.ini yacute.ini z.ini zcaron.ini ];
@czebeg = [ @czelet @czemod @czeinitmp joinc joinl joins ];
@CZEjoin = [ a aacute b c ccaron d dcaron e eacute ecaron f g h i iacute j k l m n ncaron
 o oacute p q r rcaron s scaron u uring uacute t tcaron v w x y yacute z zcaron ];
@CZEjoinc = [ a aacute b c ccaron d dcaron e eacute ecaron f g h i iacute j k l o oacute p q
 r rcaron s scaron u uring uacute x ];
@CZEjoins = [ m n ncaron t tcaron v w y yacute z zcaron ];
@CZEbmnvwy = [ m n ncaron v w y yacute ];
@CZEgjqy = [ g G j J q Q y yacute Y Yacute ];
```

Complex substitutions – insertion with FEA

```
lookup CZEjoincini {
 sub B’ @CZEjoin by B.ini;
 sub D’ @CZEjoin by D.ini;
 sub Dcaron’ @CZEjoin by Dcaron.ini;
 sub F’ @CZEjoin by F.ini;
 sub I’ @CZEjoin by I.ini;
 sub Iacute’ @CZEjoin by Iacute.ini;
 sub O’ @CZEjoin by O.ini;
 sub Oacute’ @CZEjoin by Oacute.ini;
 sub S’ @CZEjoin by S.ini;
 sub Scaron’ @CZEjoin by Scaron.ini;
 sub T’ @CZEjoin by T.ini;
 sub Tcaron’ @CZEjoin by Tcaron.ini;
 sub V’ @CZEjoin by V.ini;
 sub W’ @CZEjoin by W.ini;
} CZEjoincini;
lookup CZEjoinc {
 sub B.ini by B joinc;
 sub D.ini by D joinc;
 sub Dcaron.ini by Dcaron joinc;
 sub F.ini by F joinc;
 sub I.ini by I joinc;
 sub Iacute.ini by Iacute joinc;
 sub O.ini by O joinc;
 sub Oacute.ini by Oacute joinc;
 sub S.ini by S joinc;
 sub Scaron.ini by Scaron joinc;
 sub T.ini by T joinc;
 sub Tcaron.ini by Tcaron joinc;
 sub V.ini by V joinc;
 sub W.ini by W joinc;
} CZEjoinc;
lookup CZEjoinP {
 sub P’ @CZEjoin by P.ini;
} CZEjoinP;
lookup CZEjoinl {
 sub P.ini by P joinl;
} CZEjoinl;
```

Complex substitutions (cont.)

```
lookup CZEbegtmp {
 ignore sub @czebeg @czeini';
 sub @czeini' by @czeinitmp;
} CZEbegtmp;
lookup CZEbeg {
# sub @czeinitmp by lbounda @czeini;
 sub b.ini by lbounda b;
 sub bnarrow.ini by lbounda bnarrow;
 sub e.ini by lbounda e;
 sub eacute.ini by lbounda eacute;
 sub ecaron.ini by lbounda ecaron;
 sub f.ini by lbounda f;
 sub h.ini by lbounda h;
 sub i.ini by lbounda i;
 sub iacute.ini by lbounda iacute;
 sub j.ini by lbounda j;
 sub k.ini by lbounda k;
 sub l.ini by lbounda l;
 sub m.ini by lbounda m;
 sub n.ini by lbounda n;
 sub ncaron.ini by lbounda ncaron;
 sub p.ini by lbounda p;
 sub r.ini by lbounda r;
 sub rcaron.ini by lbounda rcaron;
 sub s.ini by lbounda s;
 sub scaron.ini by lbounda scaron;
 sub t.ini by lbounda t;
 sub tcaron.ini by lbounda tcaron;
 sub u.ini by lbounda u;
 sub uring.ini by lbounda uring;
 sub uacute.ini by lbounda uacute;
 sub v.ini by lbounda v;
 sub vnarrow.ini by lbounda vnarrow;
 sub w.ini by lbounda w;
 sub wnarrow.ini by lbounda wnarrow;
 sub x.ini by lbounda x;
 sub y.ini by lbounda y;
 sub yacute.ini by lbounda yacute;
 sub z.ini by lbounda z;
 sub zcaron.ini by lbounda zcaron;
} CZEbeg;
```

Solutions

Structure of features and lookups

METAFONT

[Czech ligtables]

[Georgian ligtables]

[Georgian scripts, TUGboat (1998 – 19:3)]

OpenType

[VOLT project] (interchange textual form)

[Feature file] (textual form)

Usage

METAFONT

The Czech and Georgian METAFONT fonts with the special effects work properly, of course only with the TEX engine and TFM.

OpenType

XETEX

I have still errors in OpenType, different of VOLT and Feature language, and I was not able to find my errors to fix them. Moreover, I cannot still confirm that easy solutions are possible. Anyway, processing of OpenType features with XETEX works properly and corresponds to my expectations.

ConTEXt/LuaTEX

The use of ConTEXt/ LuaTEX was not successful (probably, there are some other, maybe, technical problems). And I only plan to continue with tracing of OpenType fonts demonstrated by Hans Hagen — because I spent several weeks with rewriting METAFONT ligtables into VOLT project language and testing, and then transforming into feature language and tests with FontForge.

Solutions – Comparisons – Conclusions

We have 3 versions of substitution and positioning tables (all in source and production representations):

	external interchange	internal binary	
1.	METAFONT source	TFM (TEX font metrics)	
2.	MS VOLT project file	TTF flavored OpenType	MS VOLT font
3.	Adobe feature file	TTF or OTF	Adobe font

different syntax, different structure, different errors (in my font instances)

METAFONT supports /.mf	MS VOLT /.vtp	Adobe AFDKO,FF/.fea
ab -> *acb*	error	error
(leftboundary) *a* -> *ca*	allowed	unsupported
a (rightboundary) -> *ad*	allowed	unsupported

.fea: more paths and features, 2 steps and additional *real* glyphs, not efficient (or even not effective)

In some aspects, like substitution with insertion: OpenType is *not* more wider, powerful and reliable than METAFONT&TEX.

Solutions – Comparisons – Conclusions (cont.)

METAFONT one path processing with ambiguity,
limitations: only pair of adjacent chars,impossible look-ahead 3 chars, max. 256 glyphs in one font.
OpenType (VOLT or Adobe, TTF or OTF): generally more rich set of rules, but still errors in my fonts, difficult to find them and then to fix; possible interferences and collisions between features and lookups.
MS VOLT: works only on MS Windows; read and writes only TTF; has templates for GSUB and GPOS, no tools for MATH tables; VOLT can test only VOLT fonts.
Adobe AFDKO works only on MS Windows, FontForge (also Linux) cannot read .vtp, only .fea, sometimes crashes on VOLT fonts.
OpenType fonts (their correct rules) can be processed with XeTeX and ConTeXt/LuaTeX— now on Hans computer, not yet on my Linux (upgrade of context is needed).

TODO

glyph repository: continue the development
OpenType tables: debug, corrections
LuaTEX: testing and tracing tools
Support: spacing, ...
(Demo of FontForge ?)