METAFONT, METAPOST and a Malayalam font

C.V. Radhakrishnan  K.H. Hussain  K.V. Rajeesh

July 15, 2023
TUG’23, Bonn

Rachana Institute of Typography
Malayalam script
900+, *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- \(\sim 120\) base characters encoded in Unicode
- \(\sim 800\) conjuncts characters, formed from sequence of base characters
900+, *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- \( \sim 120 \) base characters encoded in Unicode
- \( \sim 800 \) conjuncts characters, formed from sequence of base characters
- Shape shifting
900+, \textit{shape-shifting} characters

- Malayalam script evolved from Brahmi, Grantha
- $\sim 120$ base characters encoded in Unicode
- $\sim 800$ conjuncts characters, formed from sequence of base characters
- Shape shifting
  
  \[
  k+r+u: \text{\ä+ö+ı} \rightarrow \text{ë}
  \]
900+, \textit{shape-shifting} characters

- Malayalam script evolved from Brahmi, Grantha
- \(\sim 120\) base characters encoded in Unicode
- \(\sim 800\) conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[ k+r+u: \text{ക+ര+ു} \rightarrow \text{ക്രു} \]
- Vowel sign forms
900+, *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- \( \sim 120 \) base characters encoded in Unicode
- \( \sim 800 \) conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[
  k+r+u: \text{\small \&} + \text{\small \&} + \text{\small \&} \rightarrow \text{\small \&}
  \]
- Vowel sign forms
  \[
  k+u: \text{\small \&} + \text{\small \&} \rightarrow \text{\small \&}
  \]
900+, *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- \(\sim 120\) base characters encoded in Unicode
- \(\sim 800\) conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[k+r+u:  \omega+\alpha+\varpi \rightarrow \epsilon\]
- Vowel sign forms
  \[k+u: \alpha+\varsigma \rightarrow \omicron\]
  \[c+u: \upsilon+\varsigma \rightarrow \upsilon\]
900+, \textit{shape-shifting} characters

- Malayalam script evolved from Brahmi, Grantha
- \(\sim 120\) base characters encoded in Unicode
- \(\sim 800\) conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[ k+r+u : \text{क+र+ു} \rightarrow \text{क्रു} \]
- Vowel sign forms
  \[ k+u : \text{क+ു} \rightarrow \text{कു} \]
  \[ c+u : \text{ച+ു} \rightarrow \text{ചു} \]
  \[ th+u : \text{ത+ു} \rightarrow \text{തു} \]
  \[ n+u : \text{ന+ു} \rightarrow \text{നു} \]
900+, *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- $\sim 120$ base characters encoded in Unicode
- $\sim 800$ conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[k+r+u: \text{കേ+ര+ു} \rightarrow \text{ക്രു}\]
- Vowel sign forms
  \[k+u: \text{ക+ു} \rightarrow \text{കു}\]
  \[c+u: \text{ച+ു} \rightarrow \text{ചു}\]
  \[th+u: \text{ത+ു} \rightarrow \text{തു}\]
  \[n+u: \text{ന+ു} \rightarrow \text{നു}\]
- Requires complex-text shaping support
900+ *shape-shifting* characters

- Malayalam script evolved from Brahmi, Grantha
- \( \sim 120 \) base characters encoded in Unicode
- \( \sim 800 \) conjuncts characters, formed from sequence of base characters
- Shape shifting
  \[ k+r+u : \text{ക+ര+ു} \rightarrow \text{ക്രു} \]
- Vowel sign forms
  \[ k+u : \text{ക+ു} \rightarrow \text{കു} \]
  \[ c+u : \text{ച+ു} \rightarrow \text{ചു} \]
  \[ th+u : \text{ത+ു} \rightarrow \text{തു} \]
  \[ n+u : \text{ന+ു} \rightarrow \text{നു} \]
- Requires complex-text shaping support
METAFONT, METAPOST
Why METAFONT

We are mostly programmers, save a designer.¹

What can we do if we can’t draw but program typefaces?

¹Gerard Unger “...but I don’t want to become a programmer — let alone a parameterizer” [Visible Language, 16(4) (1982) 353–356]
Why METAFONT

We are mostly programmers, save a designer.¹

What can we do if we can’t draw but program typefaces?

METAFONT, of course!

¹Gerard Unger “...but I don’t want to become a programmer — let alone a parameterizer” [Visible Language, 16(4) (1982) 353–356]
Why METAFONT

We are mostly programmers, save a designer.¹

What can we do if we can’t draw but program typefaces?

METAFONT, of course!

• Reusable shape components to create character shapes
• Parametric design: entire family (regular, bold, thin, slanted) from a single source
• Fine adjustments are easy thanks to programmability
• Distributed, version managed & collaborative development
• METAPOST → vector output (SVG)

¹Gerard Unger “…but I don’t want to become a programmer — let alone a parameterizer” [Visible Language, 16(4) (1982) 353–356]
Components & shape library
Components

Reference font: RIT Rachana

…and about a 100 more components.
Components

Reference font: RIT Rachana

…and about a 100 more components.
... and about a 100 more components.
%library of shape components

input ml-shape-lib;

beginfig(34);

% a component

% a component
coor_c_llobe (1) (0,0);
pstroke_c_llobe (1);
%library of shape components
input ml-shape-lib;

beginfig(34);
% a component
% a component
coor_c_llobe (1) (0,0);
pstroke_c_llobe (1);

% another component
% another component
coor_c_rlobe (2) (x1f.r-.5w2d2b,0);
pstroke_c_rlobe (2);
endfig;
Altering pre-defined shapes

Some character/component shapes need to be altered to be used in another character; such as moving a coordinate or changing width of penstroke at a coordinate.

beginfig(32);
% dx1b=-2u;
coor_g_da (1) (0,0);
pstroke_g_da (1);
endfig;
Altering pre-defined shapes

Some character/component shapes need to be altered to be used in another character; such as moving a coordinate or changing width of penstroke at a coordinate.

\begin{verbatim}
beginfig(32);
% dx1b=-2u;
coor_g_da (1) (0,0);
pstroke_g_da (1);
endfig;
\end{verbatim}

\begin{verbatim}
beginfig(32);
dx1b=-2u;
coor_g_da (1) (0,0);
pstroke_g_da (1);
endfig;
\end{verbatim}
Change width and/or angle of penstroke.

beginfig(32);
dx1b=-2u; % change x-pos
coor_g_da (1) (0,0);
ang1a=a_sw; % change angle
wd1d=wd1h=w_ea+1t; % change width
pstroke_g_da (1);
endfig;
Stacked conjuncts.

beginfig(00);
% first char with a prefix
\g1:=image(gl_pa(p));
vconj:=true; width_angle(wa_n);
% same char with different prefix
\g2:=image(gl_pa(pp));
\g3:=\g2 \times\text{scaled} .6 \times\text{scaled} .6;
currentpicture:=\g1;
addto currentpicture also \g3
  shifted (xpart (lrcorner \g1)
  -xpart(urcorner \g3),
  -(12u+5));
endfig;
The vconj usage.
A bit of woodwork

When the left edge of a horizontal stem joins with the curved stroke of the character.

%% definition:
def pstroke_edge (expr ll,ul,ur,lr) =
  filldraw ll -- ul -- ur -- lr --
cycle withcolor gcolor;
enddef;

%% usage example:
pstroke_edge (  
  ( x.prx.1f.l, y.prx.3a ),
  z.prx.1f.l ,
  z.prx.3b ,
  z.prx.3a
  );
Reusing parts of a glyph

```plaintext
%% glyph def of t1 %%
def gl_Ta =
coor_g_ta (t1.1) (0,0);
% set start of subpath
reset_cut; start:=xstt;
% strokes from 'start'
pstroke_g_ta (t1.1);
reset_xst;
enddef;

%% glyph def of t1r1 %%
def gl_TR =
% draw curve from 4th point
xstt:=3; gl_Ta;
% attach bottom component
ang_cor:=-5;
coor_c_prkar (t1.5) (x.t1.1f,3u);
start:=0;
wd.t1.5a=wd.t1.1f;
x.t1.5f=x.t1.1g;
x.t1.5a=x.t1.1d;
y.t1.5a=y.t1.1d;
wd.t1.5f:=wd.t1.1g;
pstroke_c_prkar (t1.5);
reset_cut;
enddef;
```
Vowel signs ു, ൂ, ൃ, ൄ conjoin with consonants to form conjuncts. The signs ു, ൂ (u, ū) themselves have 4 different forms depending on the base character.

<table>
<thead>
<tr>
<th>Tamil</th>
<th></th>
<th>Tamil</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ു, ൂ</td>
<td>ku, ru</td>
<td>ൄ, ൅</td>
<td>jū, rū</td>
</tr>
<tr>
<td>െ, േ, ൈ</td>
<td>gu, ju, thu</td>
<td>൅, െ, േ</td>
<td>kū, gū, thū</td>
</tr>
<tr>
<td>െ, െ, ൈ</td>
<td>nu, ṇu, nnu</td>
<td>൅, െ, േ</td>
<td>nū, ṇū, nnuū</td>
</tr>
<tr>
<td>ൈ, ṇ, ṇ, ṇ</td>
<td>du, pu, bu, mu</td>
<td>െ, െ, െ, േ</td>
<td>dū, pū, bū, mū</td>
</tr>
</tbody>
</table>
### Vowel signs

The vowel sign ‘incises’ into the base character. `unfil` doesn’t work with SVG, an ‘overlap removal’ is necessary.

```plaintext
%% α %%
def gl_pu (suffix prx) =
outln:=true;
   gl_pa(prx);
make_stem_u (21) (x.prx.3d−.5wd21c, y.prx.3c−.75wd21b);
enddef;
%% ـ-sign ـ%%
def make_stem_u (suffix $) (expr xsh,ysh) =
   coor_vl_round_u_alt ($) (xsh,ysh);
pstroke_vl_round_u_alt ($);
   if not noreverse: stem:= reverse stem; fi
   find_outlines(rmpath,stem)(P);
   for i=1 upto P.num: ypenstroke P[i]; endfor
   enddef;
```
OpenType font building
Binary font formats (OTF, TTF, WOFF2) are generated by a FontForge script, driven by config file.

**METAPOST → SVG → FontForge + scripts → OTF/TTF/WOFF2**

**Config file:**

```plaintext
#Metadata
[font]
family=Sayahna
name=Sayahna-Regular
version=0.9.1
ascent=820
descent=180
copyright=Copyright 2021–2023 Rachana Institute of Typography
<info@rachana.org.in>

#SVG, Opentype feature file, Unicode mapping
[source]
glyphdir=svgs-regular/
featurefile=features/sayahna-feature.fea
uglyphmapfile=tools/rit-ml-uc_glyph.map
scaleglyphs=True

#Width of specific glyphs
[width]
space=300

#Default and overridden left/right bearings
[bearing]
default=30,40
i1=-74,30  #negative left bearing
i2=-80,30
r1=-112,30
xx=-57,30
y2=-70,30
y2u1=-70,30
y2u2=-70,30
v2=-40,30
```
Thanks

Source code under free software licenses, LPPL and OFL:
https://gitlab.com/rit-fonts/Sayahna-font/

Acknowledgements

The METAFONT book
METATYPE1, plain_ex: JNS team
Jeroen Hellingman
Khaled Hosny

Feedback: rajeesh@rachana.org.in