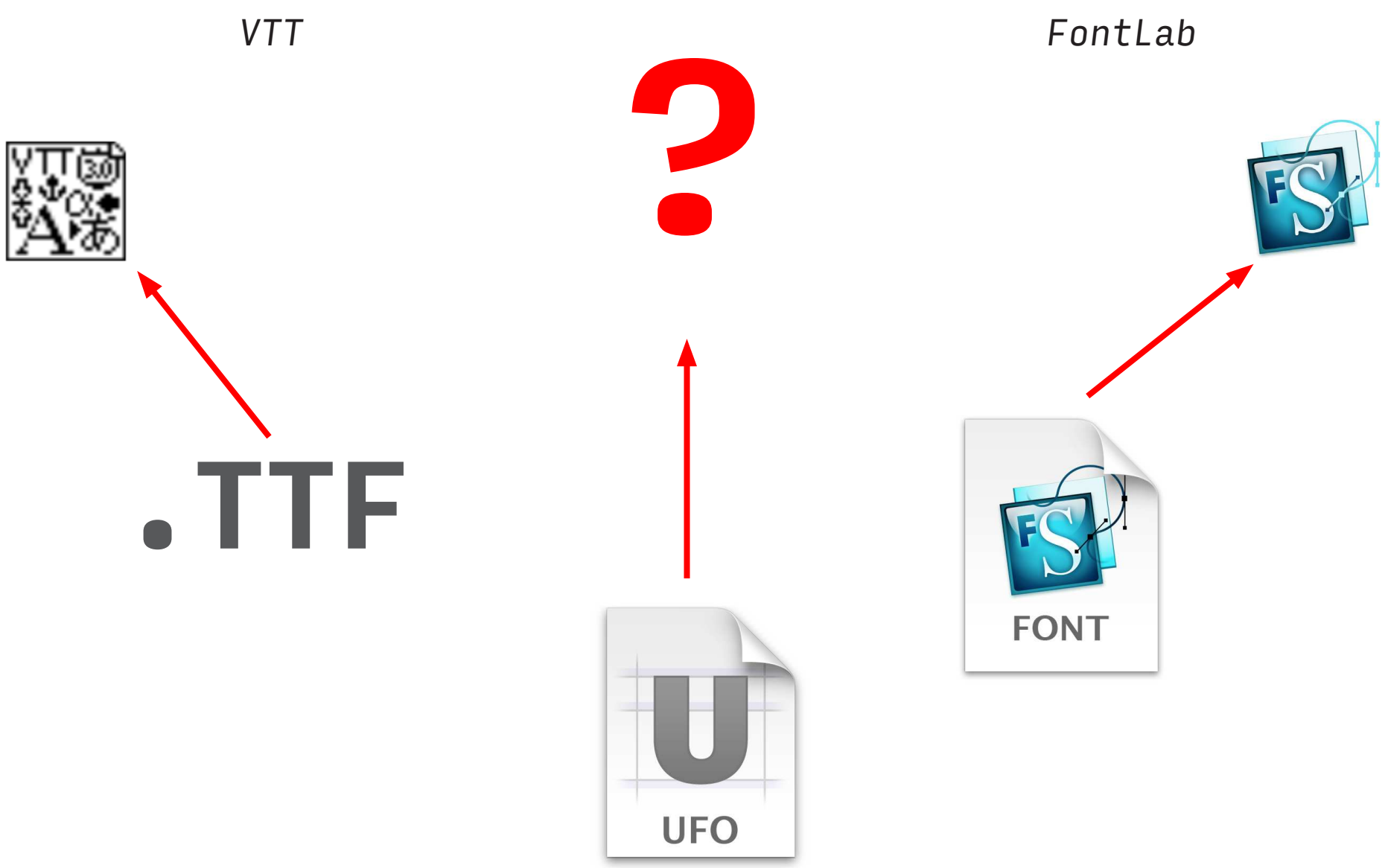
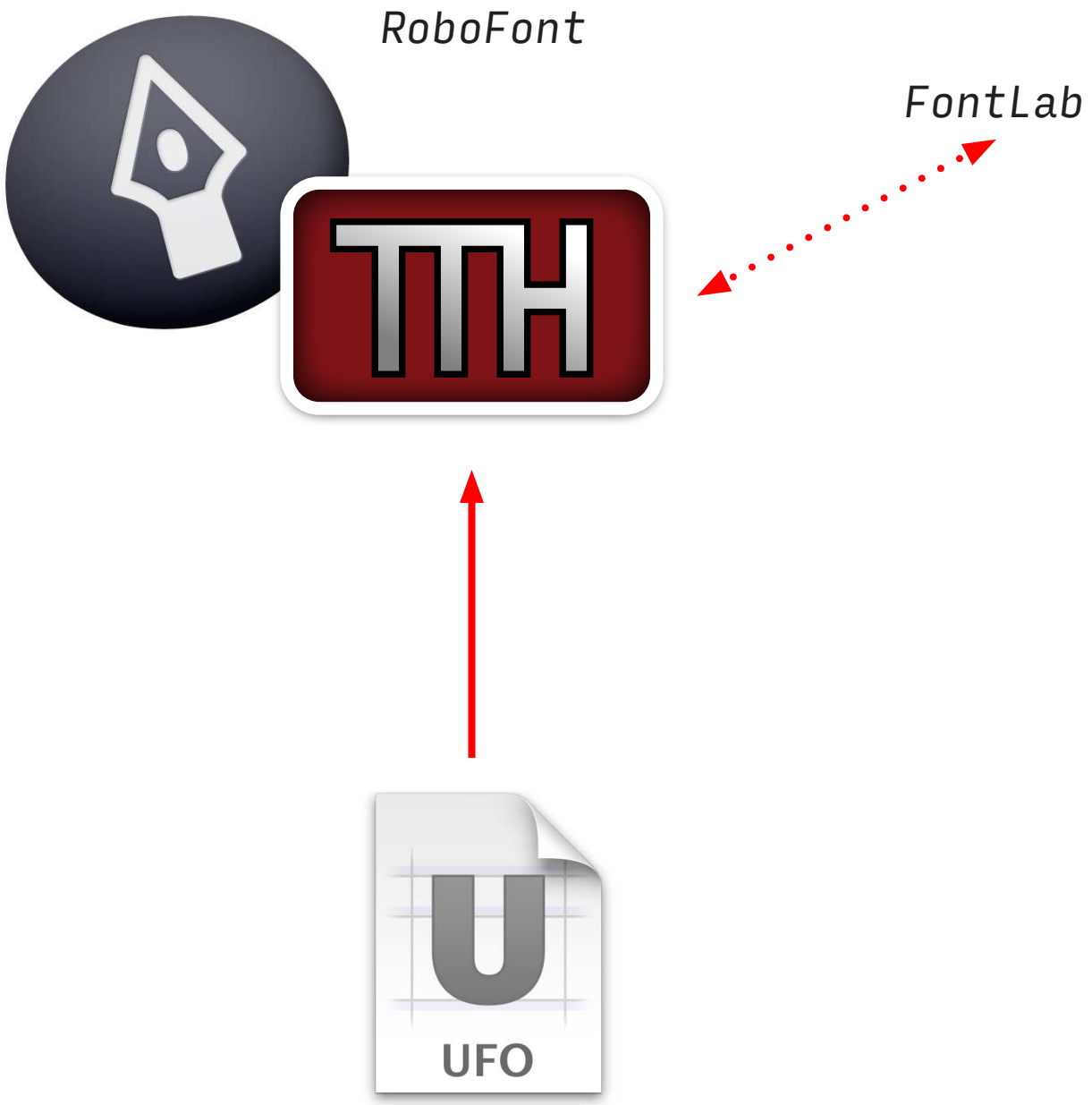
A Visual TrueType Hinting Tool for RoboFont

Jérémie Hornus & Samuel Hornus – Robothon 2015


VTT
FontLab
?
.TTF
UFO
FONT


RoboFont
FontLab
UFO

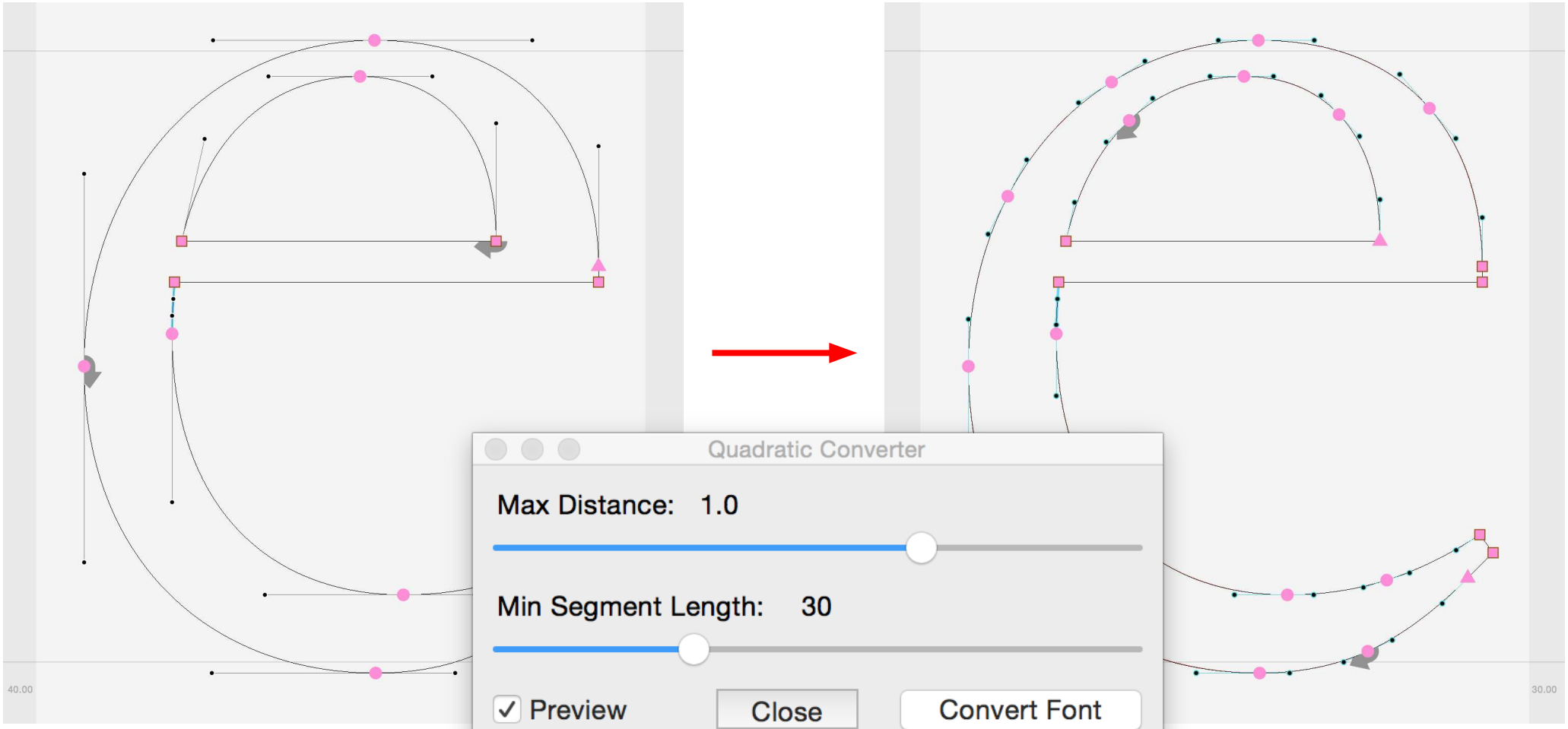
Contents

- Visual Interactive feedback
- Quadratic Converter
- From VGP to TT ASM
- Inside the UFO
- Inside the TTF

- Some tools to help the hinter’s work

DEMO time

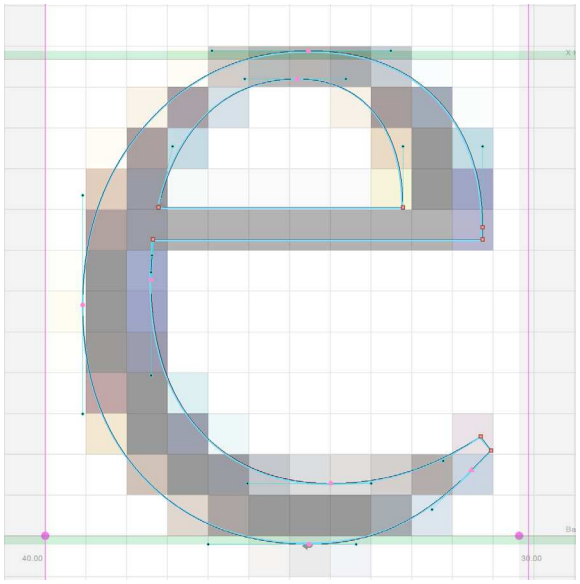
QuadraticConverter



Cubic (PostScript)

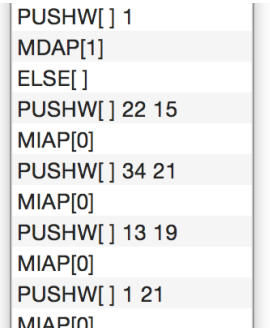
Quadratic (TrueType)

Interactive Feedback



grid-fitted outline
& bitmap

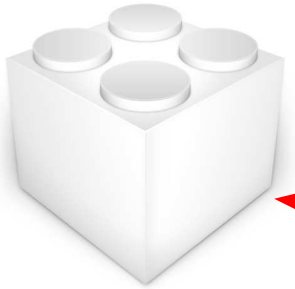
User modifies Visual Glyph Program



Update TrueType Assembly

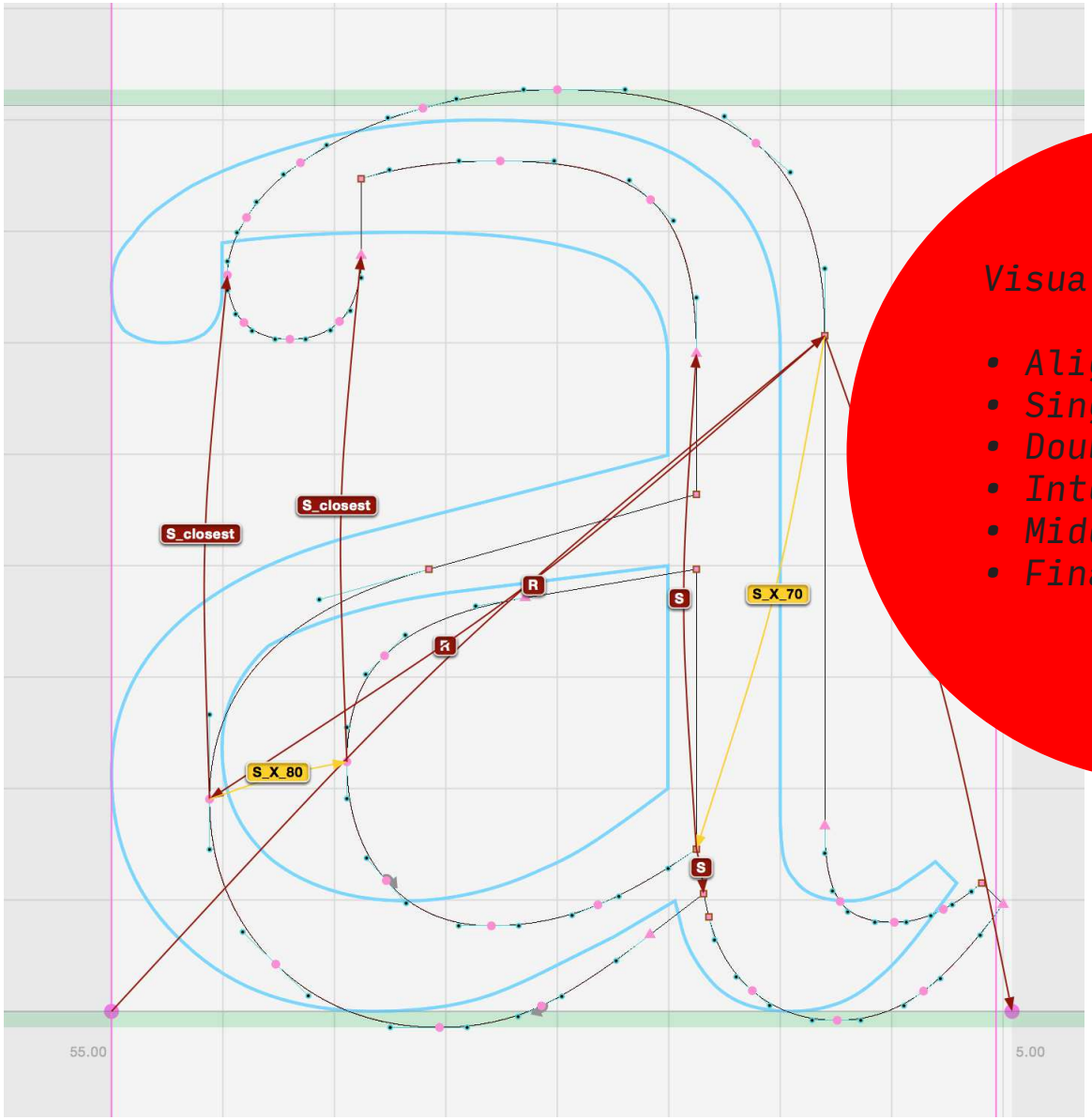
.TTF

Generate Mini Font



FreeType/freetype-py

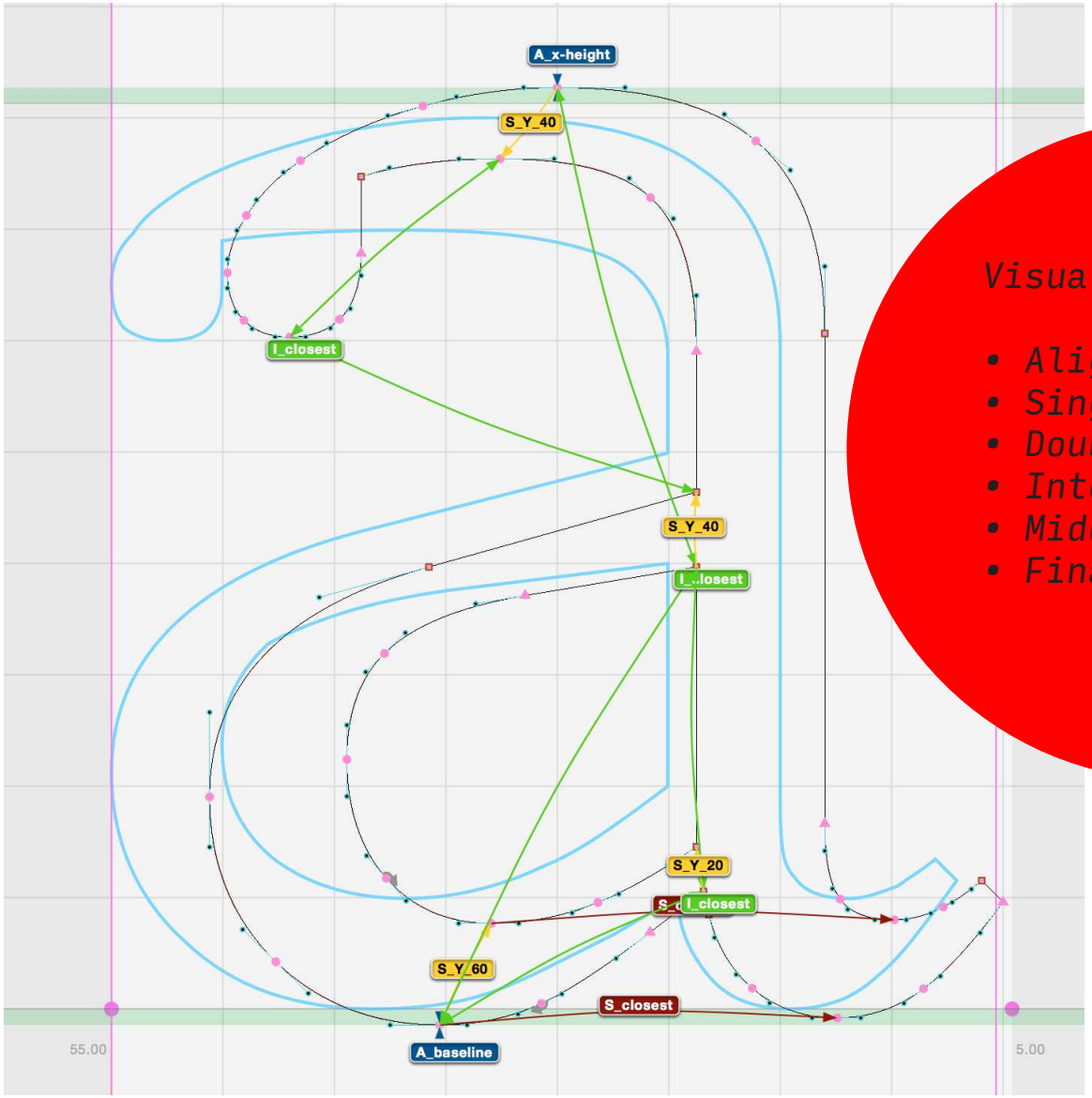
Visual Glyph Program



Visual Commands:

- *Align*
- *Single Link*
- *Double Link*
- *Interpolate*
- *Middle Delta*
- *Final Delta*

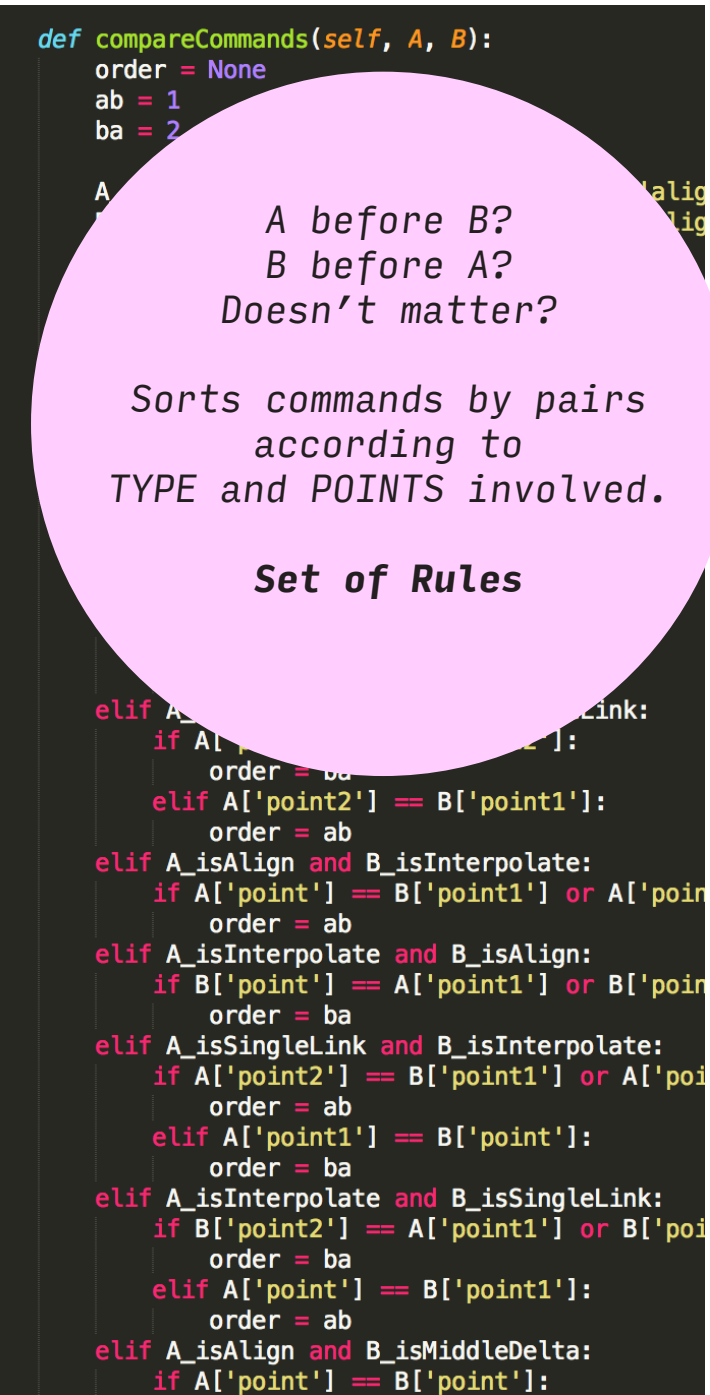
Visual Glyph Program



Visual Commands:

- *Align*
- *Single Link*
- *Double Link*
- *Interpolate*
- *Middle Delta*
- *Final Delta*

Visual Glyph Program



A before B?
B before A?
Doesn't matter?

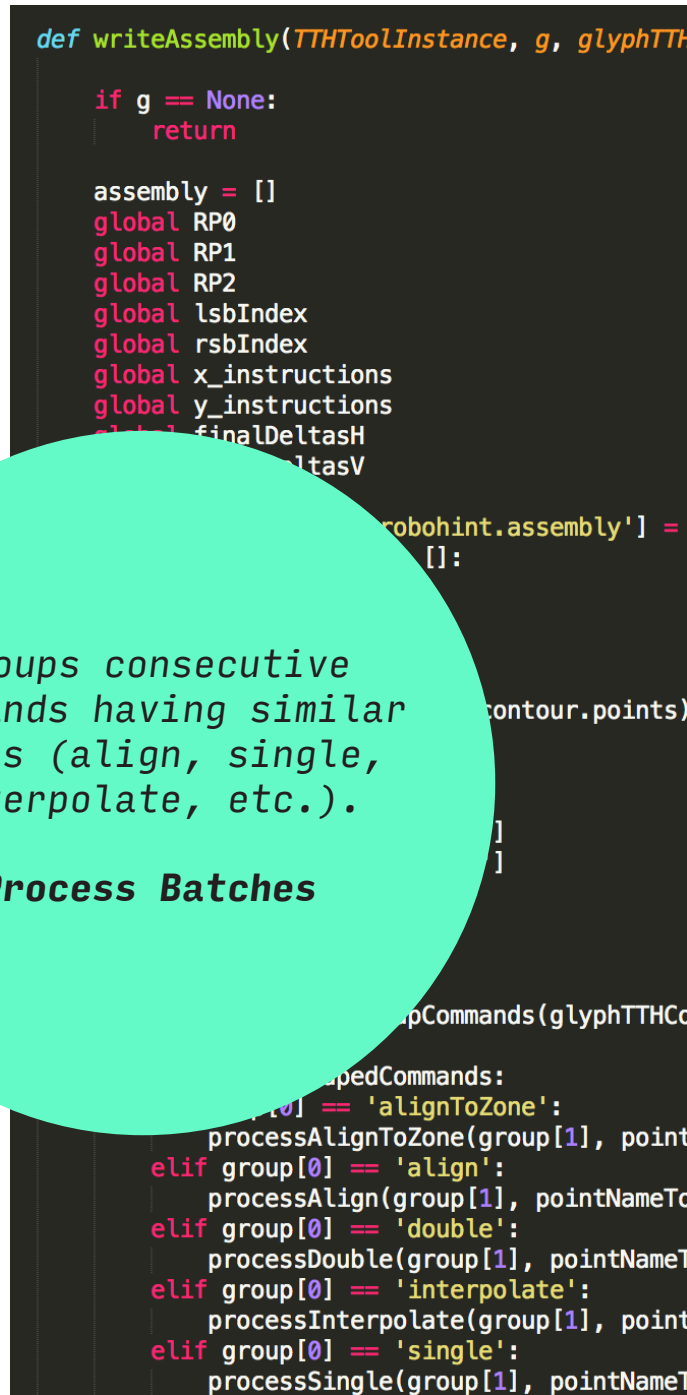
Sorts commands by pairs according to TYPE and POINTS involved.

Set of Rules

Topological Sort

Grouping

TrueType Assembly



Groups consecutive commands having similar types (align, single, interpolate, etc.).

Process Batches

Deactivate stems for grayscale and subpixel

Program

index	active	code	point	point1	point2	align	round	stem	zone	delta	ppm1	ppm2	mono	gray
0	☑	alignh	lsb			round								
1	☑	singleh		lsb	*167058B04		true							
2	☑	singleh		*167058B04	rsb		true							
3	☑	singleh		*167058B04	*166F6B82C			X_80						
4	☑	singleh		*167058B04	*16706304C		true							
5	☑	singleh		*16706304C	*166F6BC3C			X_70						
6	☑	singleh		*166F6BC3C	*1670585BC									
7	☑	singleh		*166F6BC3C	*167062F14									
8	☑	singleh		*16706304C	*16705841C		true							
9	☑	mdeltah	*166F6B82C							5	17	18	true	false
10	☑	mdeltah	*166F6B82C							5	13	14	true	false
11	☑	alignt	*167058894						x-height					
12	☑	alignb	*167058D74						baseline					
13	☑	alignt	*1670584EC						ascenders					
14	☑	alignb	*167062F7C						baseline					
15	☑	singlev		*167058894	*166F6BA9C			Y_60						
16	☑	singlev		*167058D74	*166F6B5BC			Y_30						
17	☑	interpolatev	*1670585BC	*167058894	*166F6B5BC	round								
18	☑	interpolatev	*167062F14	*167058D74	*167058894	round								
19	☑	singlev		*1670585BC	*166F6BBD4			Y_30						
20	☑	singlev		*167062F14	*166F6BC3C			Y_60						
21	☑	singlev		*1670584EC	*167058484		true							
22	☑	singlev		*167058484	*16705841C			Y_20						

BATCH

Assembly

```
PUSHB[ ] 10
CALL[ ]
DUP[ ]
PUSHB[ ] 0
SWAP[ ]
WS[ ]
PUSHB[ ] 1
SWAP[ ]
WS[ ]
SVTCA[1]
PUSHW[ ] 62
MDAP[1]
PUSHW[ ] 62
SRP0[ ]
PUSHW[ ] 28
MDRP[11100]
PUSHW[ ] 28
SRP0[ ]
PUSHW[ ] 63
MDRP[11100]
PUSHW[ ] 28
SRP0[ ]
PUSHB[ ] 0
RS[ ]
IF[ ]
PUSHW[ ] 49
MDRP[10000]
ELSE[ ]
PUSHW[ ] 49 9
MIRP[10100]
EIF[ ]
PUSHB[ ] 1
RS[ ]
PUSHB[ ] 0
EQ[ ]
IF[ ]
PUSHW[ ] 140 49 15...
DELTAP1[ ]
PUSHW[ ] 76 49 92 49 2
DELTAP1[ ]
EIF[ ]
SVTCA[0]
PUSHW[ ] 0
RCVT[ ]
IF[ ]
PUSHW[ ] 22
MDAP[1]
PUSHW[ ] 34
```

lib

lib.plist

lib.plist 〉 No Selection

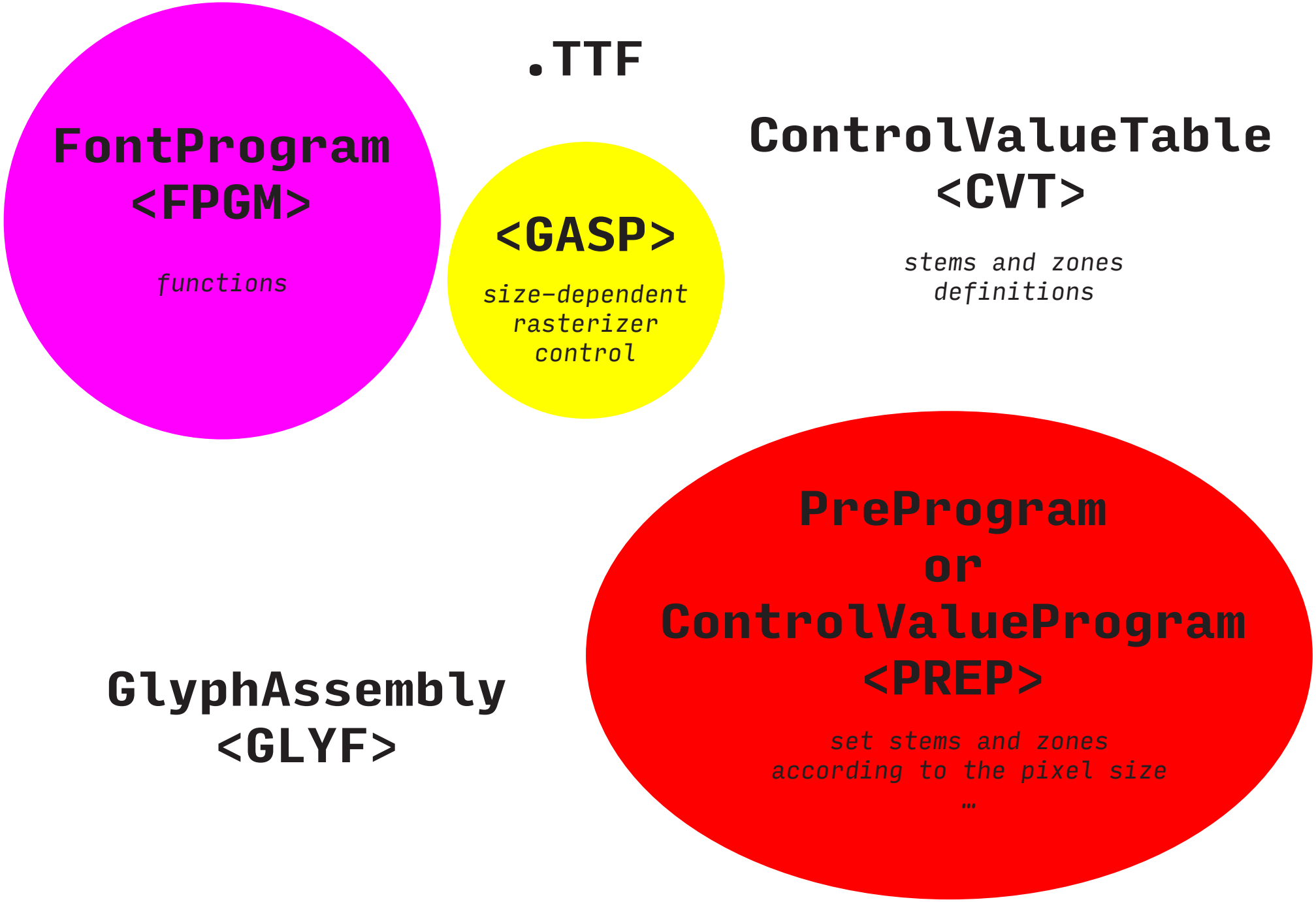
Key	Type	Value
▼ Root	Dictionary	(14 items)
▼ com.fontlab.v2.tth	Dictionary	(5 items)
alignppm	Number	48
codeppm	Number	72
▶ stems	Dictionary	(12 items)
stemsnap	Number	17
▶ zones	Dictionary	(5 items)
▶ com.robofont.robohint.cvt	Array	(23 items)
▶ com.robofont.robohint.fpgm	Array	(334 items)
▶ com.robofont.robohint.gasp	Dictionary	(4 items)
com.robofont.robohint.maxp.maxFun…	Number	12
com.robofont.robohint.maxp.maxStorage	Number	5
▶ com.robofont.robohint.prep	Array	(54 items)
▼ com.sansplomb.tth	Dictionary	(2 items)
bitmapPreviewSelection	String	Monochrome
deactivateStemWhenGrayScale	Number	1
▶ com.typemytype.robofont.Cubic cont…	Array	(4 items)
▶ com.typemytype.robofont.foreground…	Array	(4 items)
▶ com.typemytype.robofont.layerOrder	Array	(1 item)
com.typemytype.robofont.segmentType	String	qcurve
▶ com.typemytype.robofont.sort	Array	(1 item)
▶ public.glyphOrder	Array	(737 items)

.glif

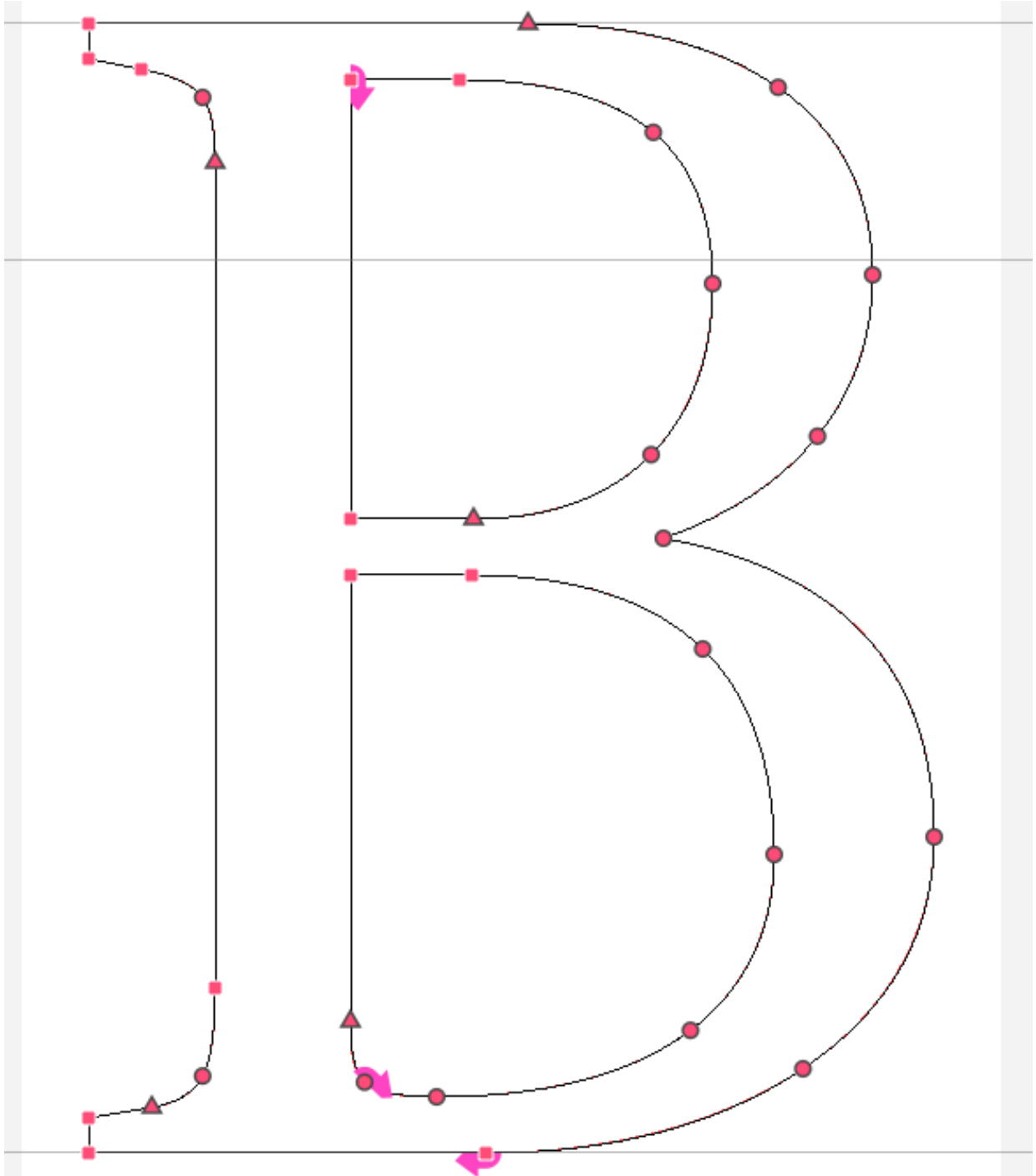
```
</outline>
<lib>
  <dict>
    <key>com.fontlab.ttprogram</key>
    <data>PHR0UHJvZ3JhbT48dHRjIGFjdGl2ZT0idHJ1ZSIgYWxpZ249InJvdW5kIiBjb2RlPSJh
    bGlnbmgiIHBvaW50PSJsc2IiC8+PHR0YyBhY3RpdmU9InRydWUiIGNvZGU9InNpbmds
    ZWgiIHBvaW50MT0ibHNiIiBwb2ludDI9IioxNjcwNThCMDQiIHJvdW5kPSJ0cnVlIiAv
    Pjx0dGMgYWN0aXZlPSJ0cnVlIiBjb2RlPSJzaW5nbGVoIiBwb2ludDE9IioxNjcwNThC
    MDQiIHBvaW50Mj0icnNiIiByb3VuZD0idHJ1ZSIgLz48dHRjIGFjdGl2ZT0idHJ1ZSIg
    Y29kZT0ic2luZ2xlaCIgcG9pbnQxPSIqMTY3MDU4QjA0IiBwb2ludDI9IioxNjZGNkI4
    MkMiIHN0ZW09IlhfODAiIC8+PHR0YyBhY3RpdmU9InRydWUiIGNvZGU9InNpbmdsZWgi
    IHBvaW50MT0iKjE2NzA1OEIwNCIgcG9pbnQyPSIqMTY3MDYzMDRDIiByb3VuZD0idHJ1
    ZSIgLz48dHRjIGFjdGl2ZT0idHJ1ZSIgY29kZT0ic2luZ2xlaCIgcG9pbnQxPSIqMTY3
    MDYzMDRDIiBwb2ludDI9IioxNjZGNkJCRDQiIHN0ZW09IlhfNzAiIC8+PHR0YyBhY3Rp
    dmU9InRydWUiIGFsaWduPSJyb3VuZCIgY29kZT0ic2luZ2xlaCIgcG9pbnQxPSIqMTY2
    RjZCQkQ0IiBwb2ludDI9IioxNjcwNTg1QkMiIC8+PHR0YyBhY3RpdmU9InRydWUiIGNv
    ZGU9InNpbmdsZWgiIHBvaW50MT0iKjE2NzA2MzA0QyIgcG9pbnQyPSIqMTY3MDU4NDFD
    IiByb3VuZD0idHJ1ZSIgLz48dHRjIGFjdGl2ZT0idHJ1ZSIgY29kZT0iYWxpZ250IiBw
    b2ludD0iKjE2NzA1ODRFQyIgem9uZT0iYXNjZW5kZXJzIiAvPjx0dGMgYWN0aXZlPSJ0
    cnVlIiBjb2RlPSJhbGlnbnQiIHBvaW50PSIqMTY3MDU4ODk0IiB6b25lPSJ4LWhlaWdo
    dCIgLz48dHRjIGFjdGl2ZT0idHJ1ZSIgY29kZT0iYWxpZ25iIiBwb2ludD0iKjE2NzA1
    OEQ3NCIgem9uZT0iYmFzZWxpbmUiC8+PHR0YyBhY3RpdmU9InRydWUiIGNvZGU9ImFs
    aWduYiIgcG9pbnQ9IioxNjcwNjJGN0MiIHpvbmU9ImJhc2VsaW5lIiAvPjx0dGMgYWN0
    aXZlPSJ0cnVlIiBjb2RlPSJzaW5nbGV2IiBwb2ludDE9IioxNjcwNTg0RUMiIHBvaW50
    Mj0iKjE2NzA1ODQ4NCIgcm91bmQ9InRydWUiC8+PHR0YyBhY3RpdmU9InRydWUiIGNv
    ZGU9InNpbmdsZXYiIHBvaW50MT0iKjE2NzA1ODg5NCIgcG9pbnQyPSIqMTY2RjZCQTlD
    IiBzdGVtPSJZXzYwIiAvPjx0dGMgYWN0aXZlPSJ0cnVlIiBjb2RlPSJzaW5nbGV2IiBw
    b2ludDE9IioxNjcwNThENzQiIHBvaW50Mj0iKjE2NkY2QjVCQyIgc3RlbT0iWV8zMCIg
    Lz48dHRjIGFjdGl2ZT0idHJ1ZSIgY29kZT0ic2luZ2xldiIgcG9pbnQxPSIqMTY3MDU4
    RDc0IiBwb2ludDI9IioxNjcwNjJGMTQiIHBvaW50Mj0iMCIgLz48dHRjIGFjdGl2ZT0i
    PSJ0cnVlIiBjb2RlPSJzaW5nbGV2IiBwb2ludDE9IioxNjcwNjJGMTQiIHBvaW50Mj0i
    KjE2NkY2QkMzQyIgc3RlbT0iWV82MCIgLz48dHRjIGFjdGl2ZT0idHJ1ZSIgY29kZT0i
    <key>com.robofont.robohint.assembly</key>
    <array>
      <string>PUSHB[ ] 10</string>
      <string>CALL[ ]</string>
      <string>DUP[ ]</string>
      <string>PUSHB[ ] 0</string>
      <string>SWAP[ ]</string>
      <string>WS[ ]</string>
      <string>PUSHB[ ] 1</string>
      <string>SWAP[ ]</string>
      <string>WS[ ]</string>
      <string>SVTCA[1]</string>
      <string>PUSHW[ ] 62</string>
      <string>MDAP[1]</string>
      <string>PUSHW[ ] 62</string>
      <string>SRP0[ ]</string>
      <string>PUSHW[ ] 28</string>
      <string>MDRP[11100]</string>
      <string>PUSHW[ ] 28</string>
      <string>SRP0[ ]</string>
      <string>PUSHW[ ] 63</string>
      <string>MDRP[11100]</string>
      <string>PUSHW[ ] 28</string>
      <string>SRP0[ ]</string>
      <string>PUSHB[ ] 0</string>
      <string>RS[ ]</string>
      <string>IF[ ]</string>
      <string>PUSHW[ ] 49</string>
      <string>MDRP[10000]</string>
      <string>ELSE[ ]</string>
      <string>PUSHW[ ] 49 9</string>
      <string>MIRP[10100]</string>
      <string>EIF[ ]</string>
      <string>PUSHW[ ] 28</string>
      <string>SRP0[ ]</string>
      <string>PUSHW[ ] 3</string>
      <string>MDRP[11100]</string>
      <string>PUSHW[ ] 3</string>
      <string>SRP0[ ]</string>
      <string>PUSHB[ ] 0</string>
      <string>RS[ ]</string>
      <string>IF[ ]</string>
      <string>PUSHW[ ] 58</string>
      <string>MDRP[10000]</string>
      <string>ELSE[ ]</string>
      <string>PUSHW[ ] 58 8</string>
      <string>MIRP[10100]</string>
      <string>EIF[ ]</string>
      <string>PUSHW[ ] 58</string>
      <string>SRP0[ ]</string>
```

com.fontlab.ttprogram
Visual Glyph Program
encoded in base 64

com.robofont.robohint.assembly
Plain TrueType Assembly

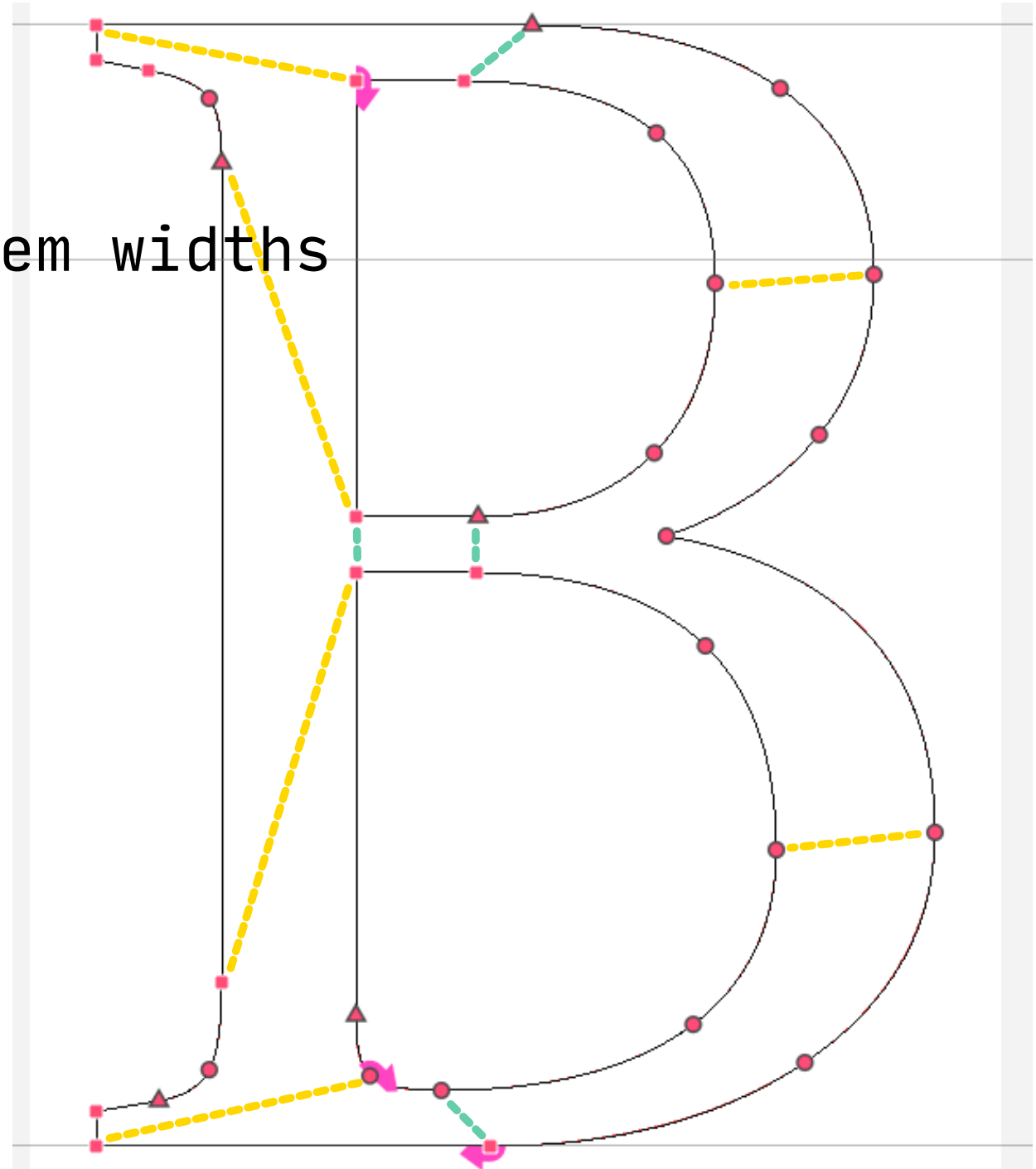

.TTF
FontProgram
<FPGM>
functions
<GASP>
size-dependent
rasterizer
control
ControlValueTable
<CVT>
stems and zones
definitions
PreProgram
or
ControlValueProgram
<PREP>
set stems and zones
according to the pixel size
…
GlyphAssembly
<GLYF>

Helper Tools



Helper Tools

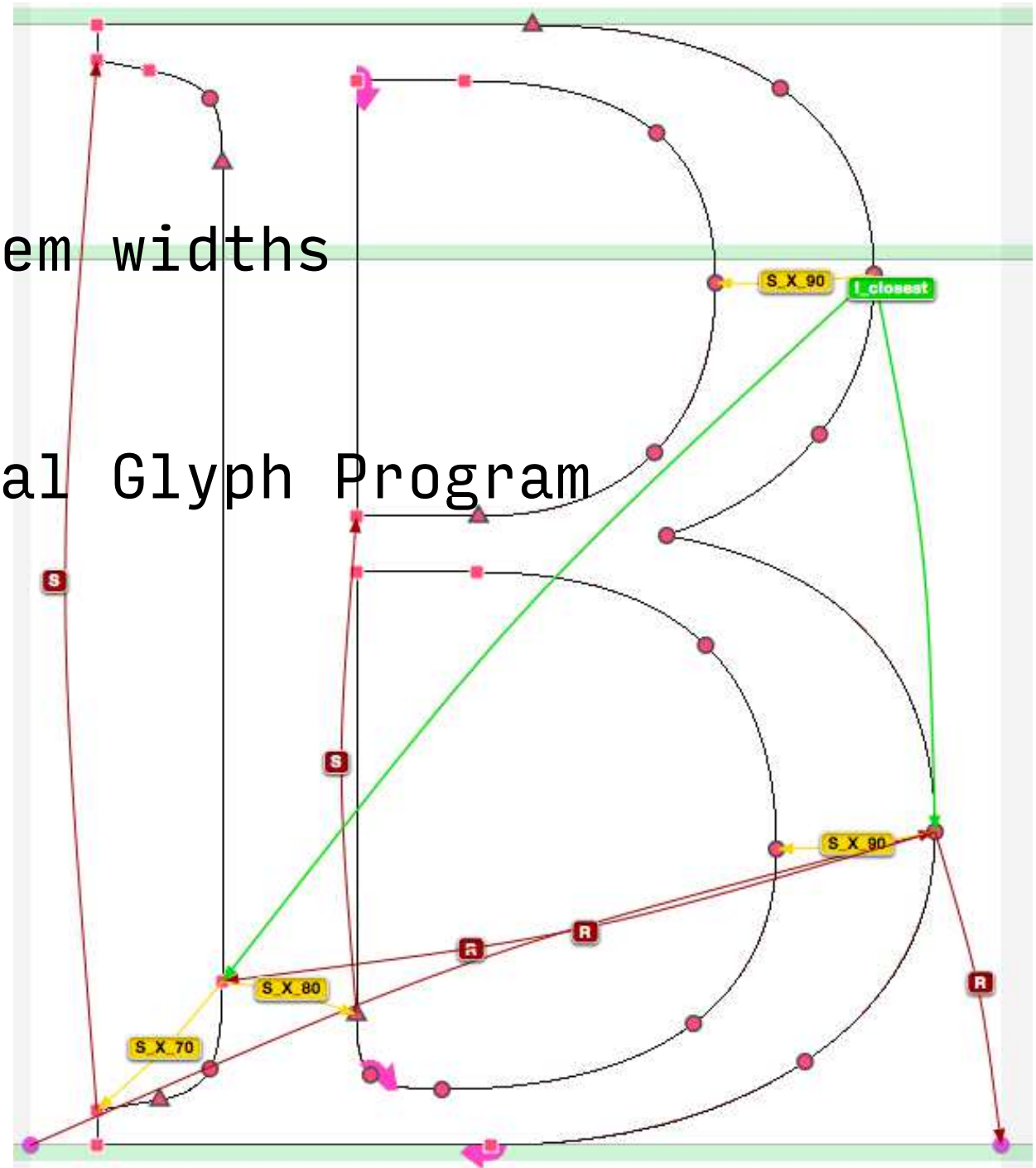
Auto-stem
Finds typical stem widths



Helper Tools

Auto-stem
Finds typical stem widths

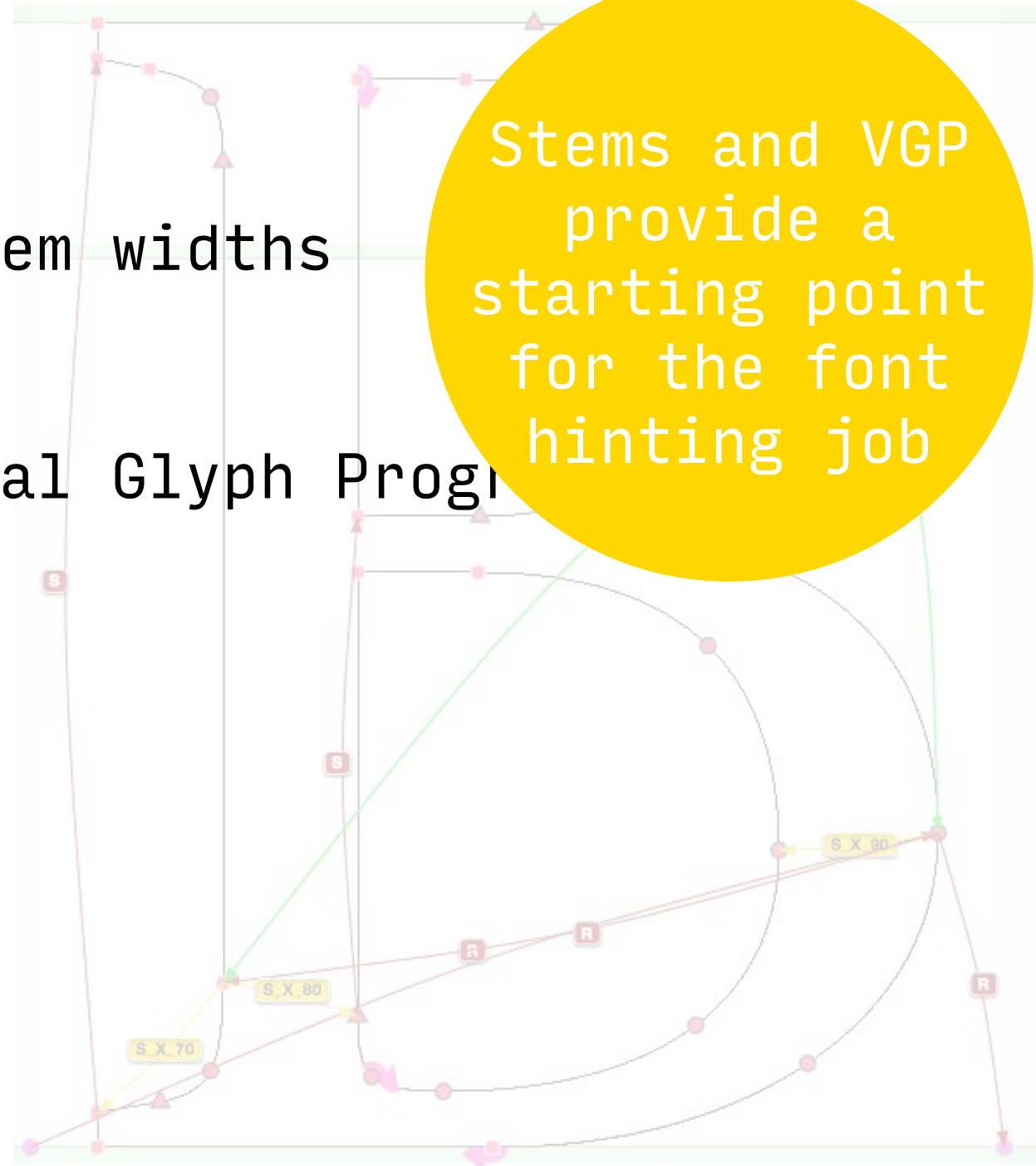
Auto-hint
Generates a Visual Glyph Program



Helper Tools

Auto-stem
Finds typical stem widths

Auto-hint
Generates a Visual Glyph Progr



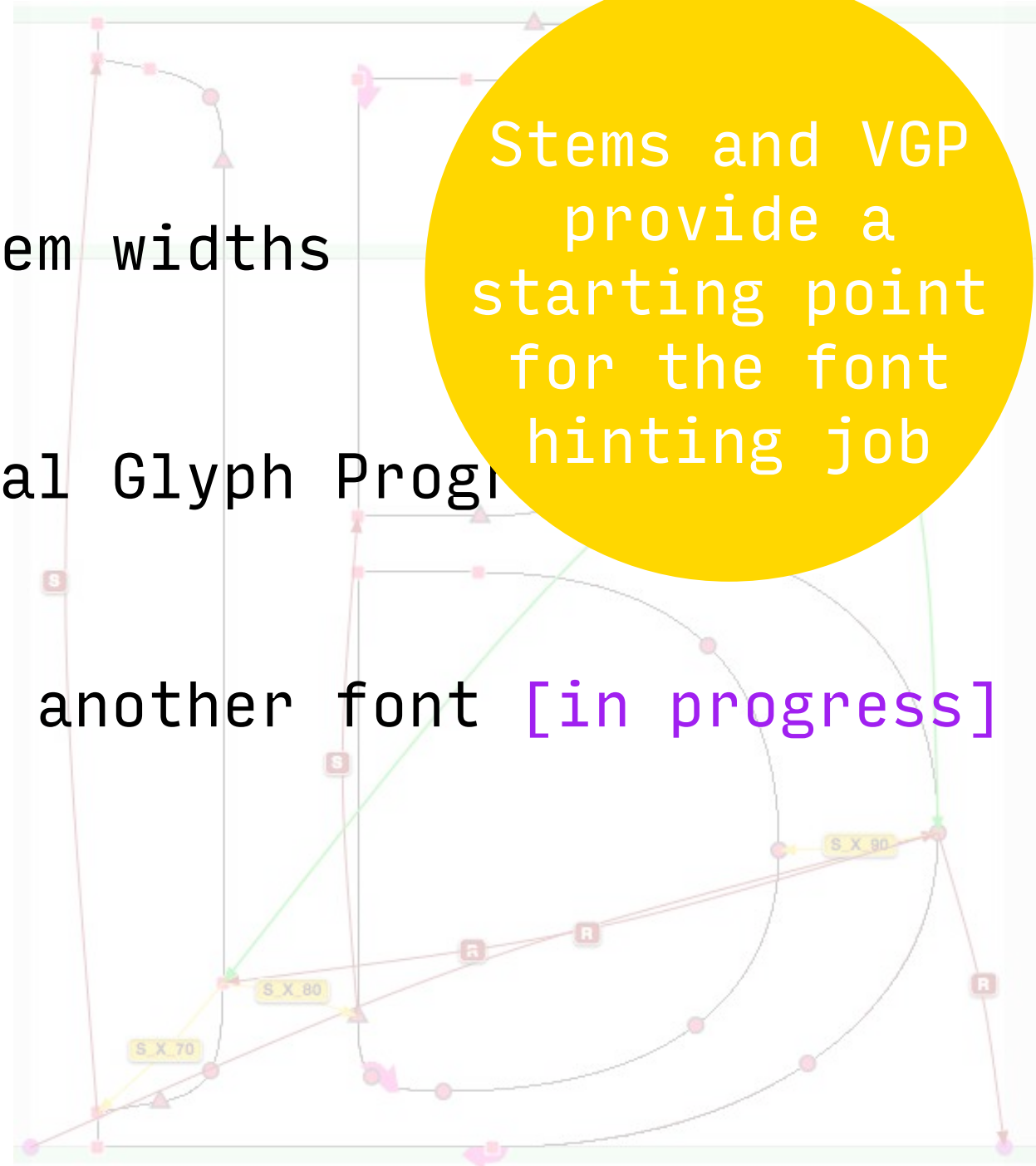
Stems and VGP provide a starting point for the font hinting job

Helper Tools

Auto-stem
Finds typical stem widths

Auto-hint
Generates a Visual Glyph Progr

Auto-match
Transfers VGP to another font [in progress]



Stems and VGP provide a starting point for the font hinting job

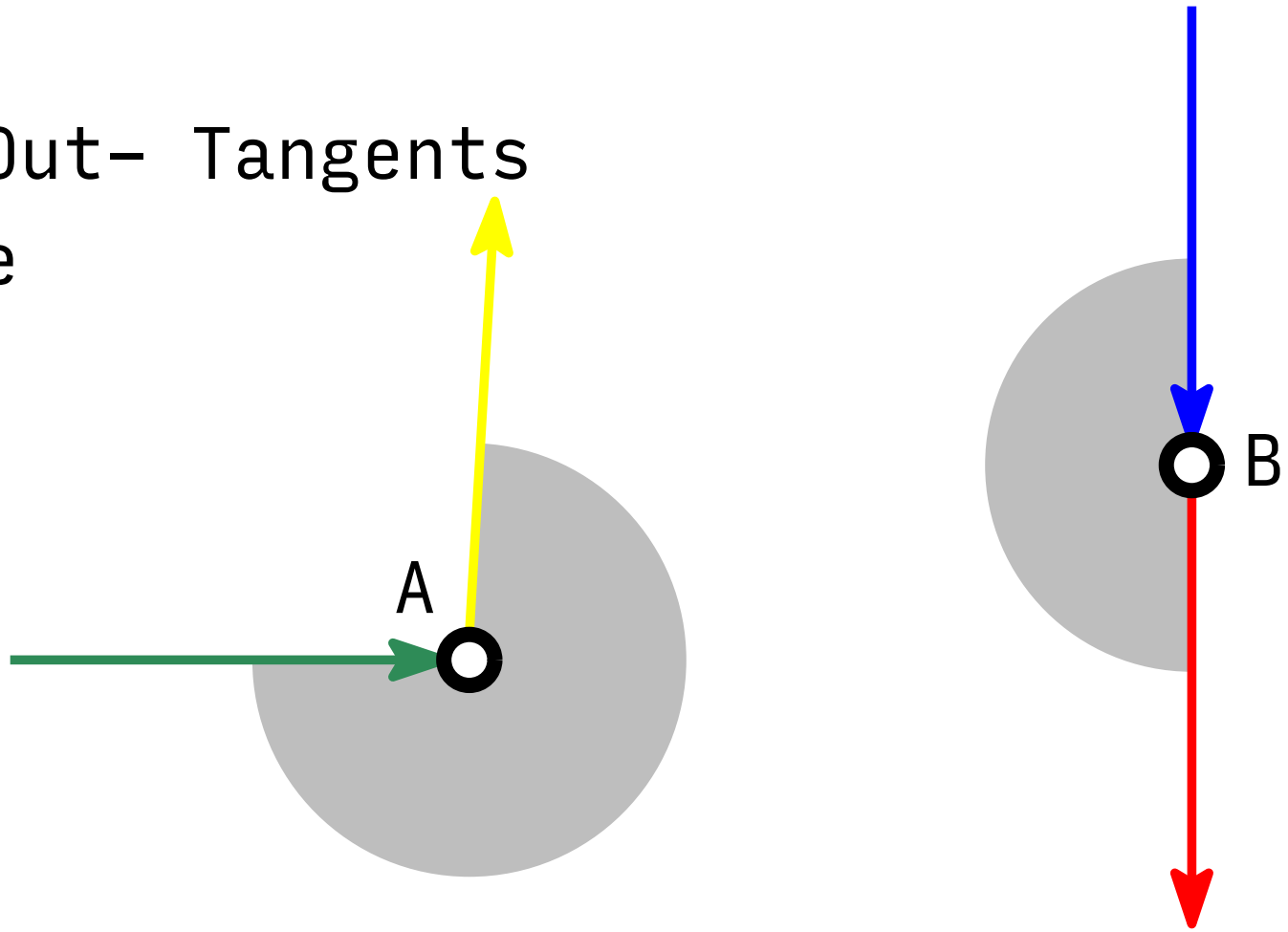
Defining a ‘stem’

‘stem’ = ‘special pair of control points’

Defining a ‘stem’

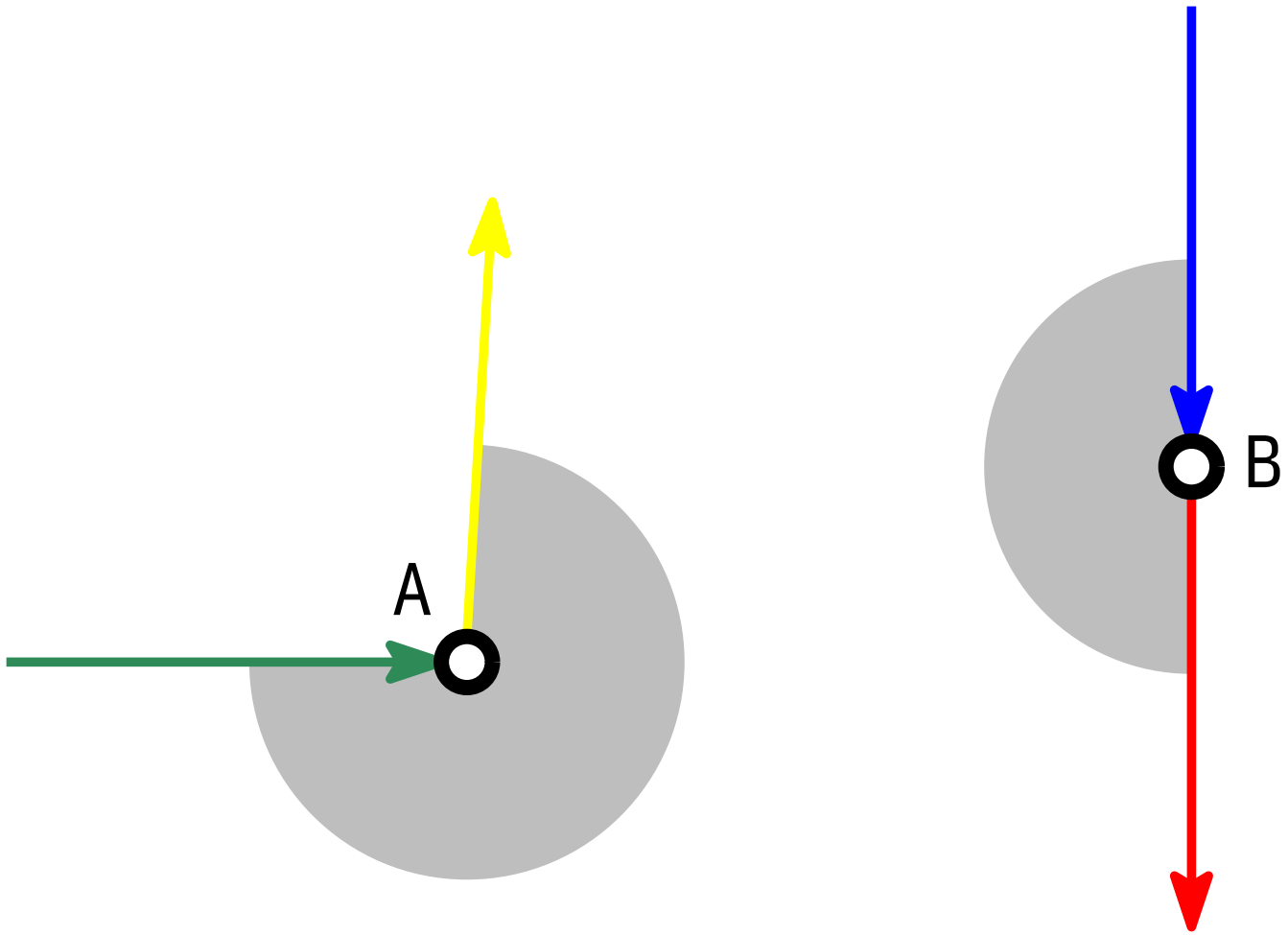
Configuration of two points A and B:

- Position
- In- and Out- Tangents
- Clockwise



Defining a ‘stem’

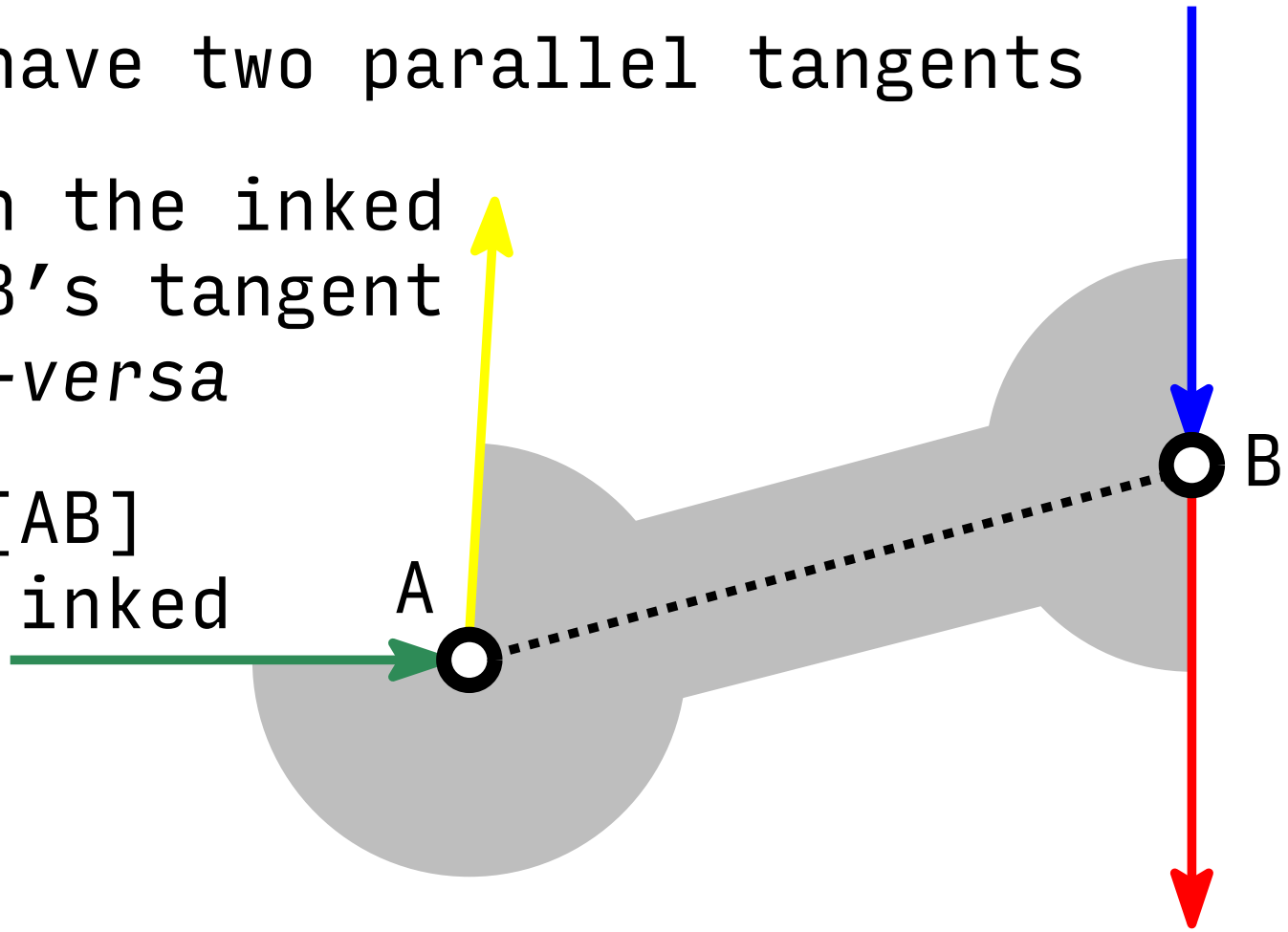
The two points A and B form a stem when:



Defining a ‘stem’

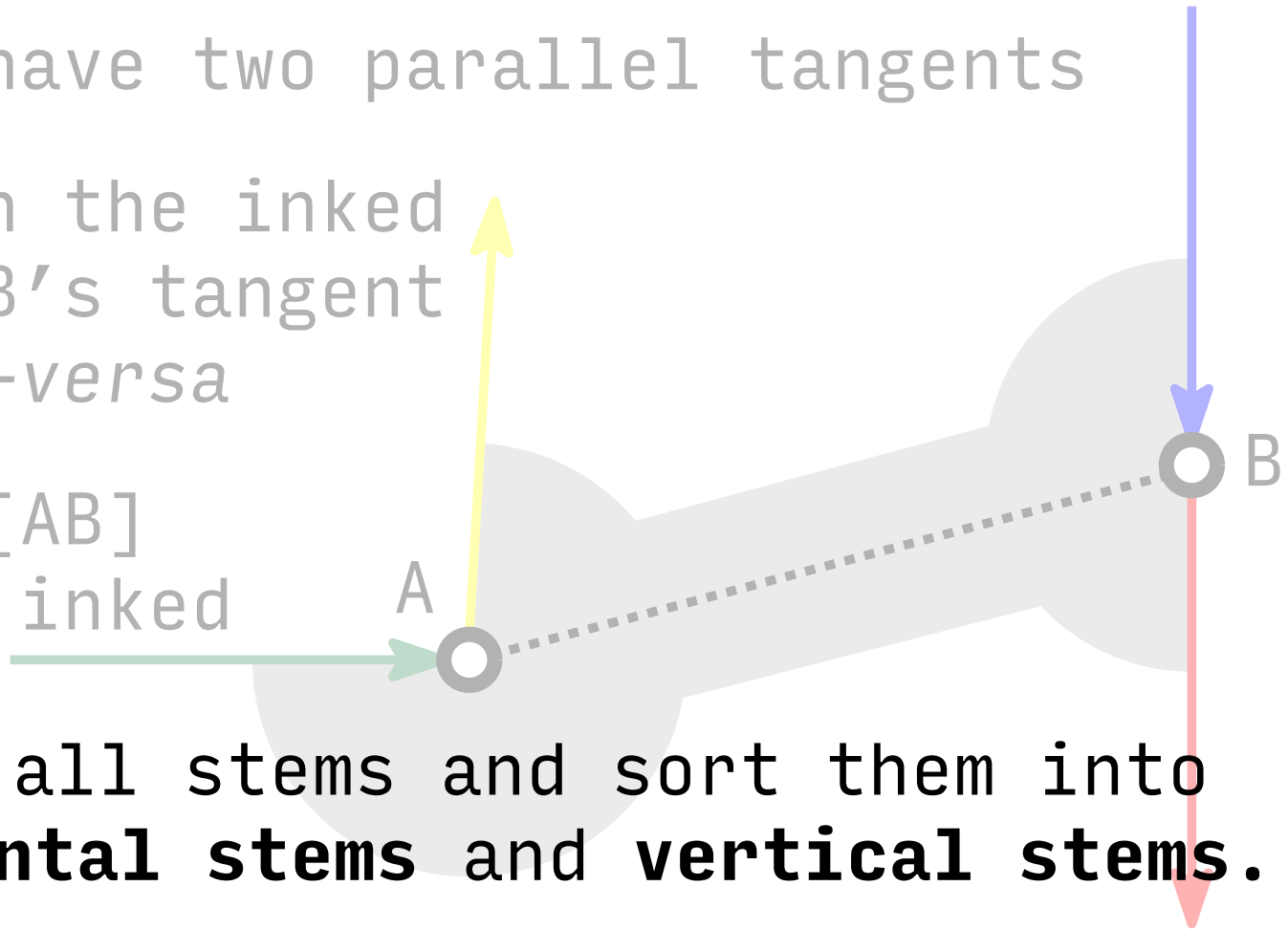
The two points A and B form a stem when:

- A and B have two parallel tangents
- A lies on the inked side of B’s tangent and *vice-versa*
- Segment [AB] is fully inked



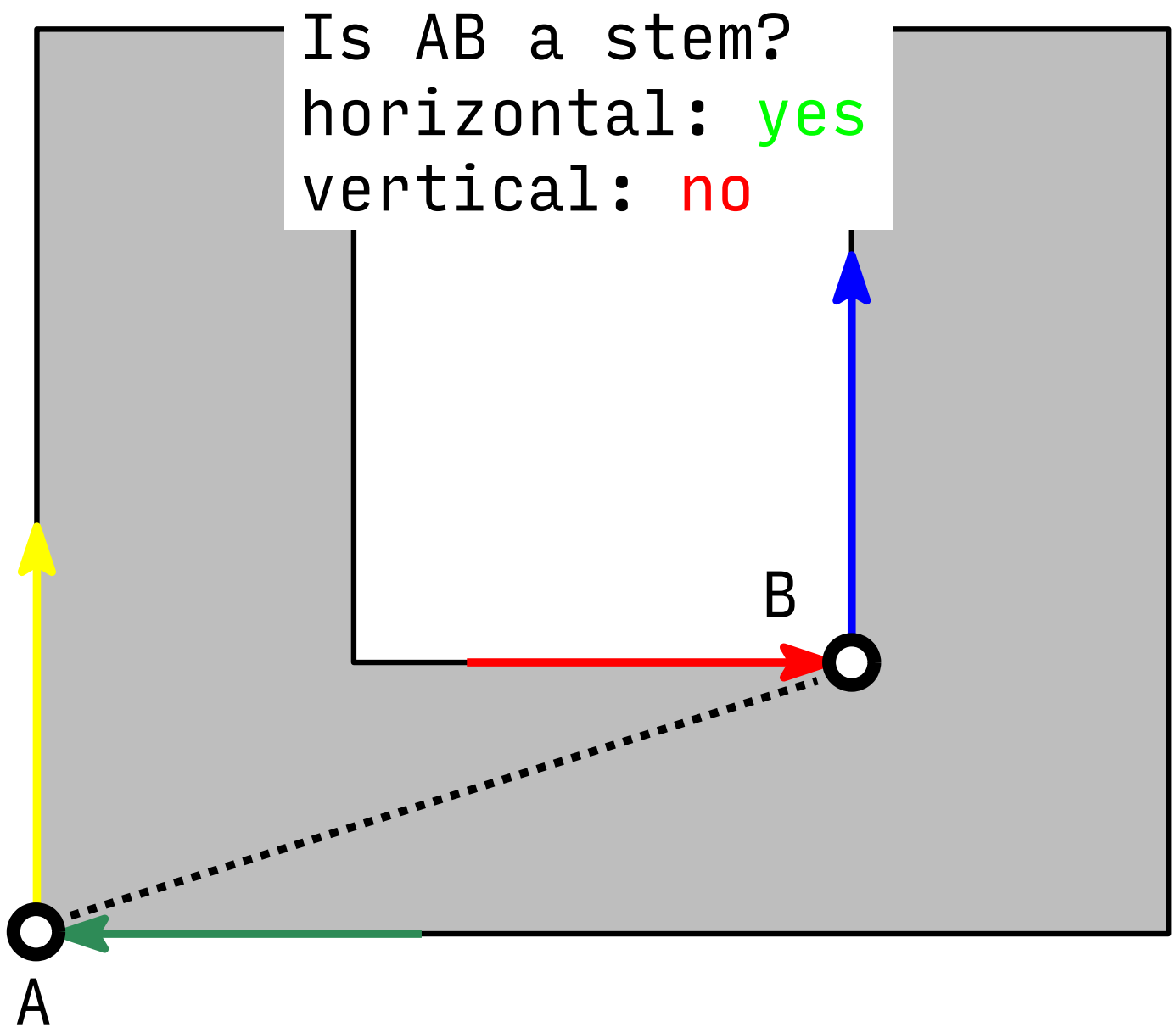
Finding ‘stems’

- A and B have two parallel tangents
- A lies on the inked side of B’s tangent and *vice-versa*
- Segment [AB] is fully inked



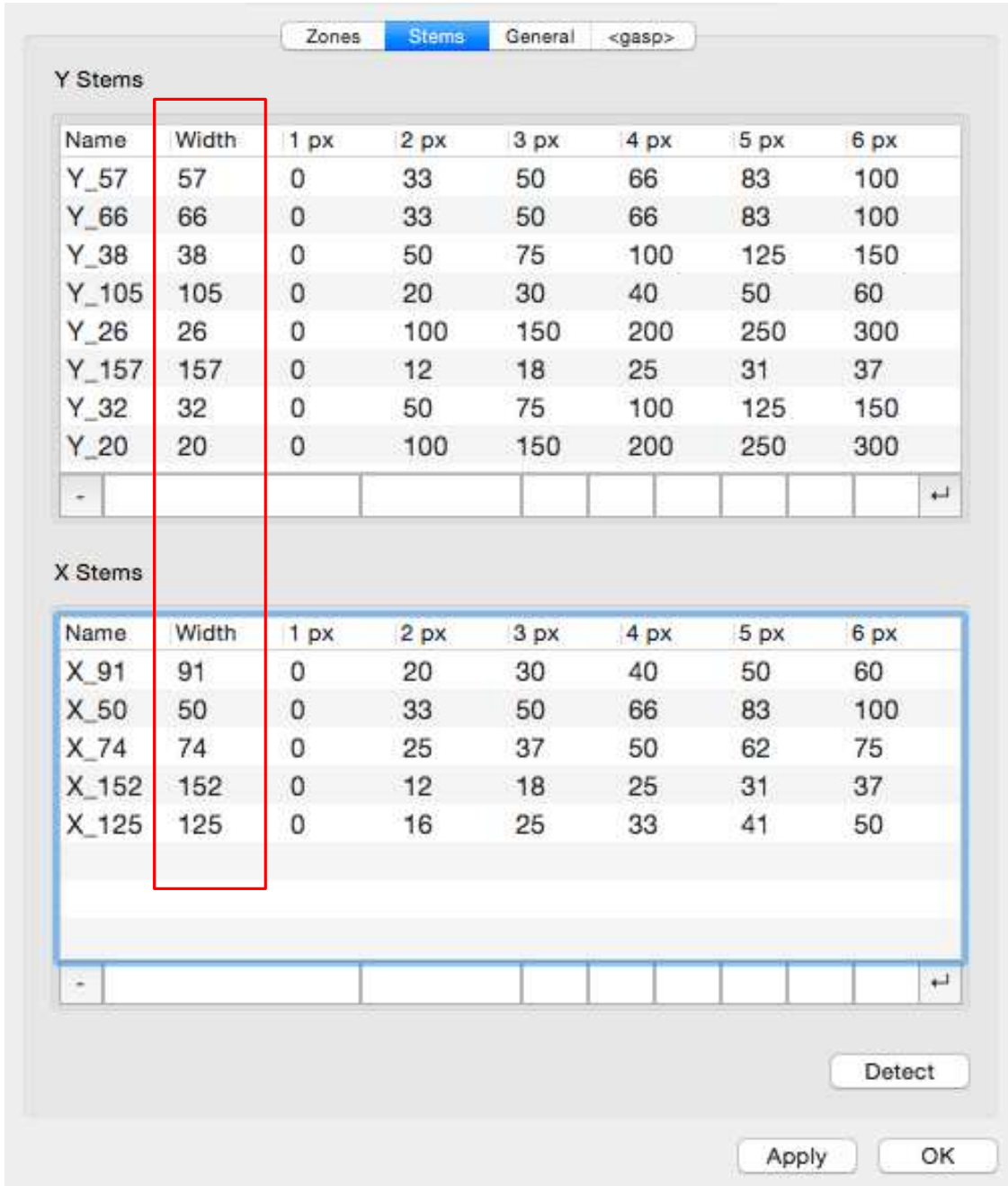
Find all stems and sort them into **horizontal stems** and **vertical stems.**

Finding ‘stems’



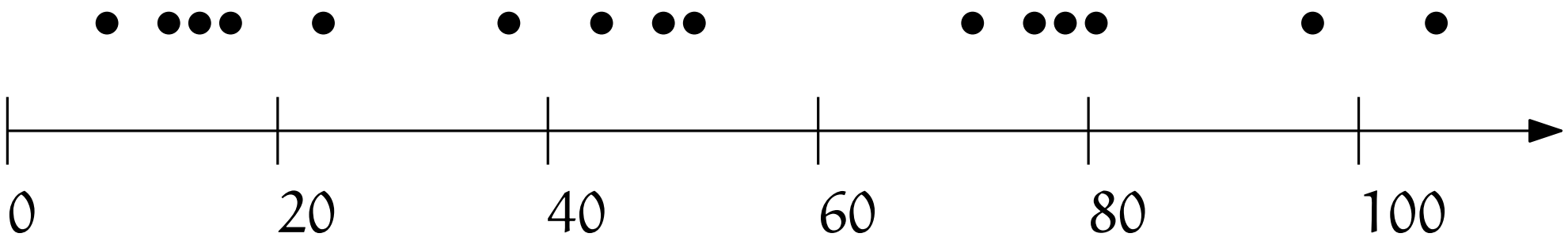
Auto-stem

[filling the CVT with relevant stem widths]



Auto-stem

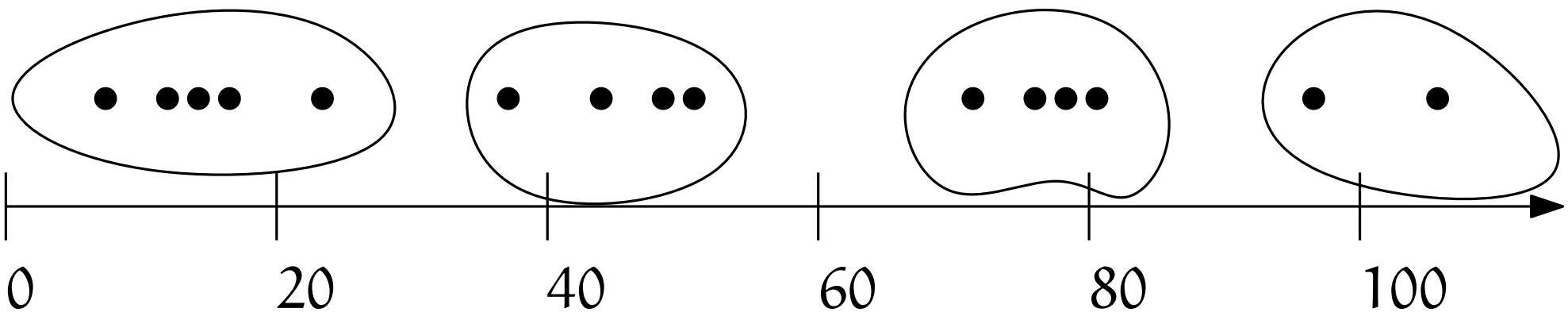
1 Gather stem widths into an array
  (of numbers)



Auto-stem

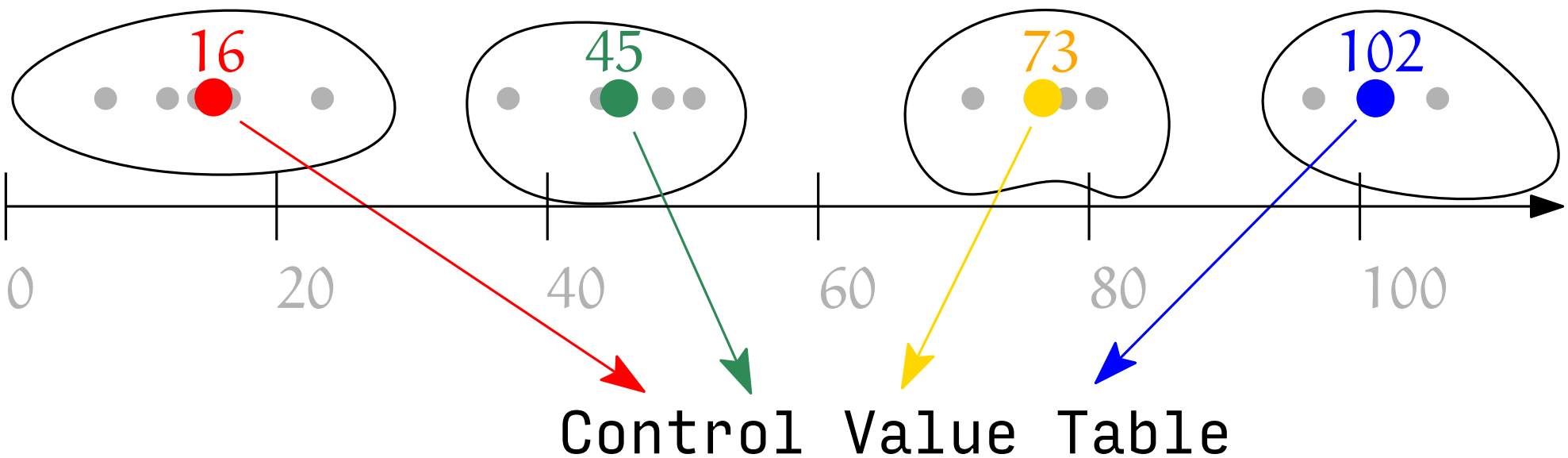
1 Gather stem widths into an array (of numbers)

2 Cluster them into groups (k-means)



Auto-stem

1. Gather stem widths into an array (of numbers)
2. Cluster them into groups (k-means)
3. Take the mean of each group as a relevant stem width

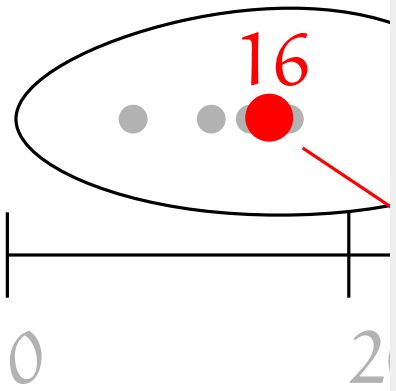


Auto-stem

1 Gather
(of nu

2 Cluste s)

3 Take t
releva



Zones | Stems | General | <gasp>

Y Stems

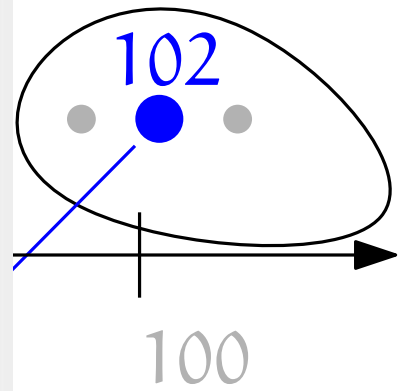
Name	Width	1 px	2 px	3 px	4 px	5 px	6 px
Y_57	57	0	33	50	66	83	100
Y_66	66	0	33	50	66	83	100
Y_38	38	0	50	75	100	125	150
Y_105	105	0	20	30	40	50	60
Y_26	26	0	100	150	200	250	300
Y_157	157	0	12	18	25	31	37
Y_32	32	0	50	75	100	125	150
Y_20	20	0	100	150	200	250	300

X Stems

Name	Width	1 px	2 px	3 px	4 px	5 px	6 px
X_91	91	0	20	30	40	50	60
X_50	50	0	33	50	66	83	100
X_74	74	0	25	37	50	62	75
X_152	152	0	12	18	25	31	37
X_125	125	0	16	25	33	41	50

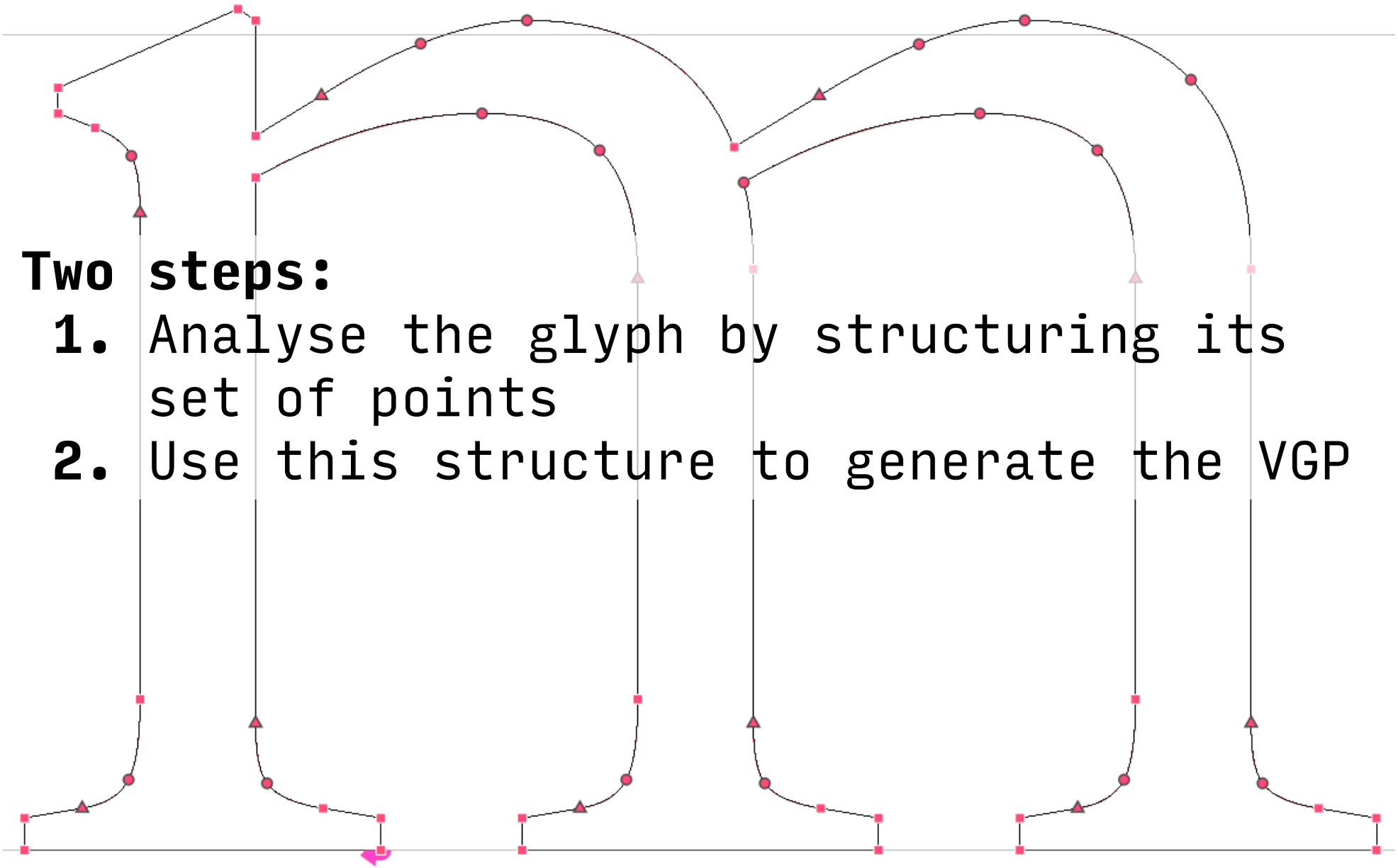
Detect

Apply OK



Auto-hinting

[Automatically generates a VGP]

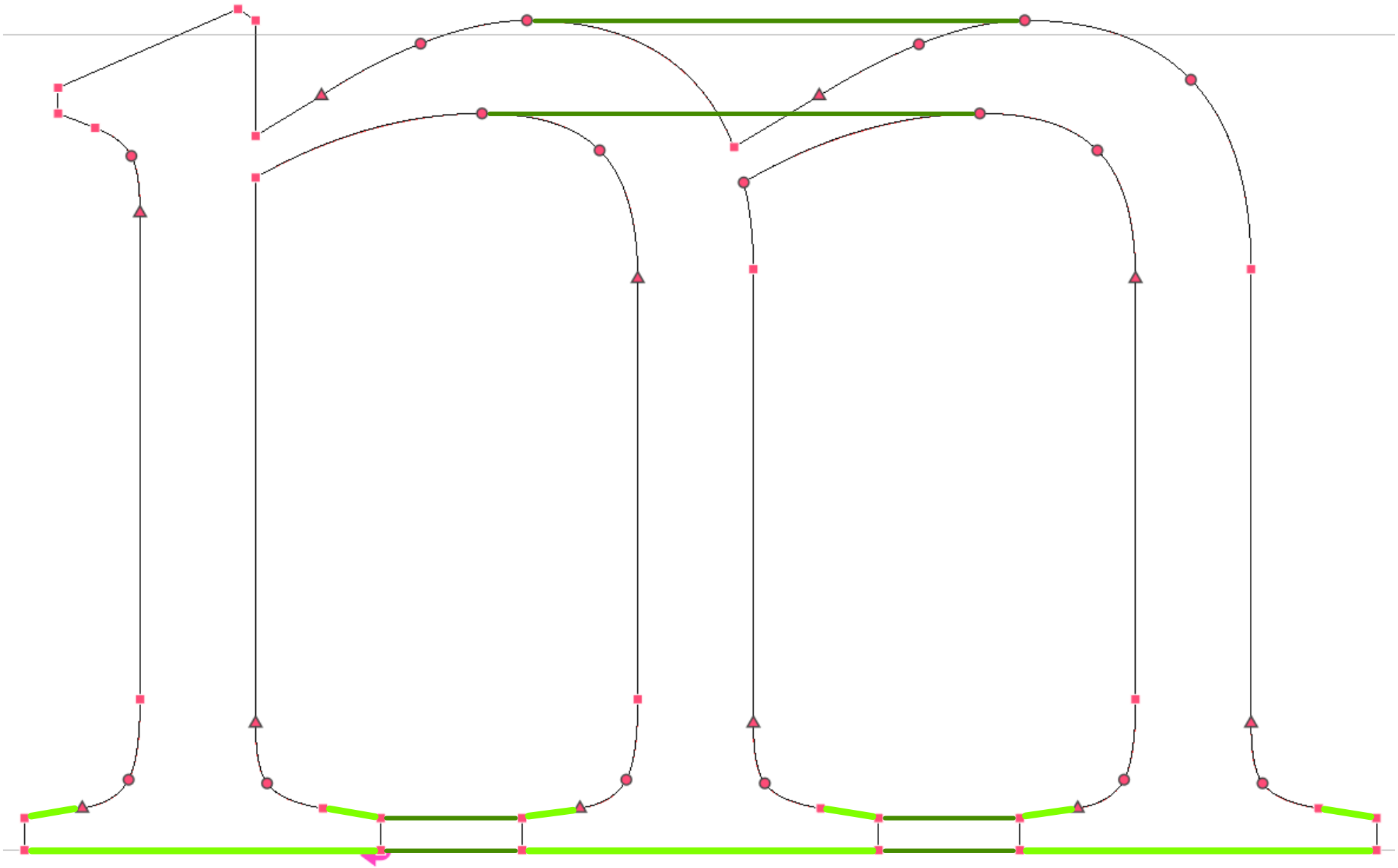


Two steps:

1. Analyse the glyph by structuring its set of points
2. Use this structure to generate the VGP

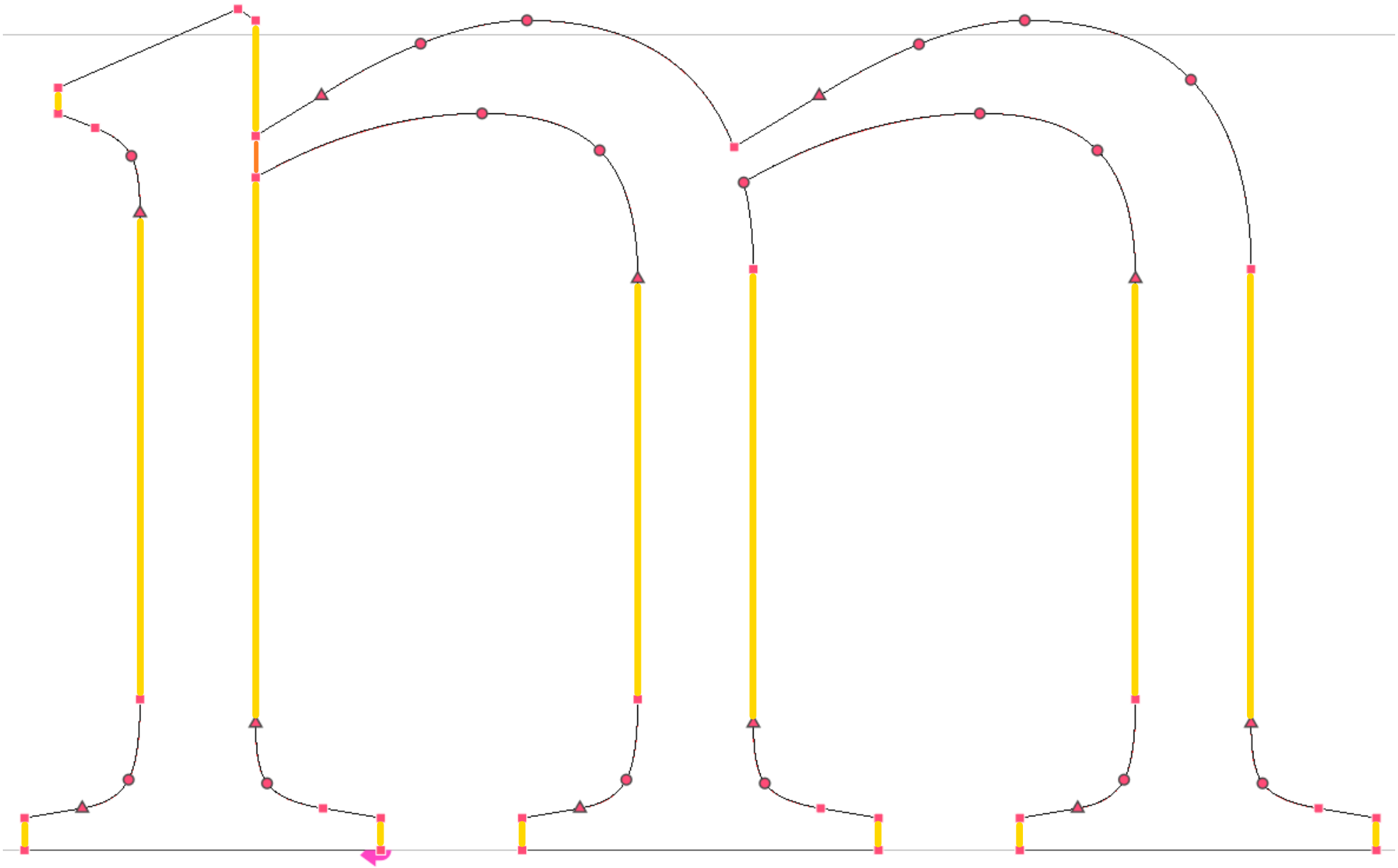
Alignments

horizontal alignments for vertical hinting

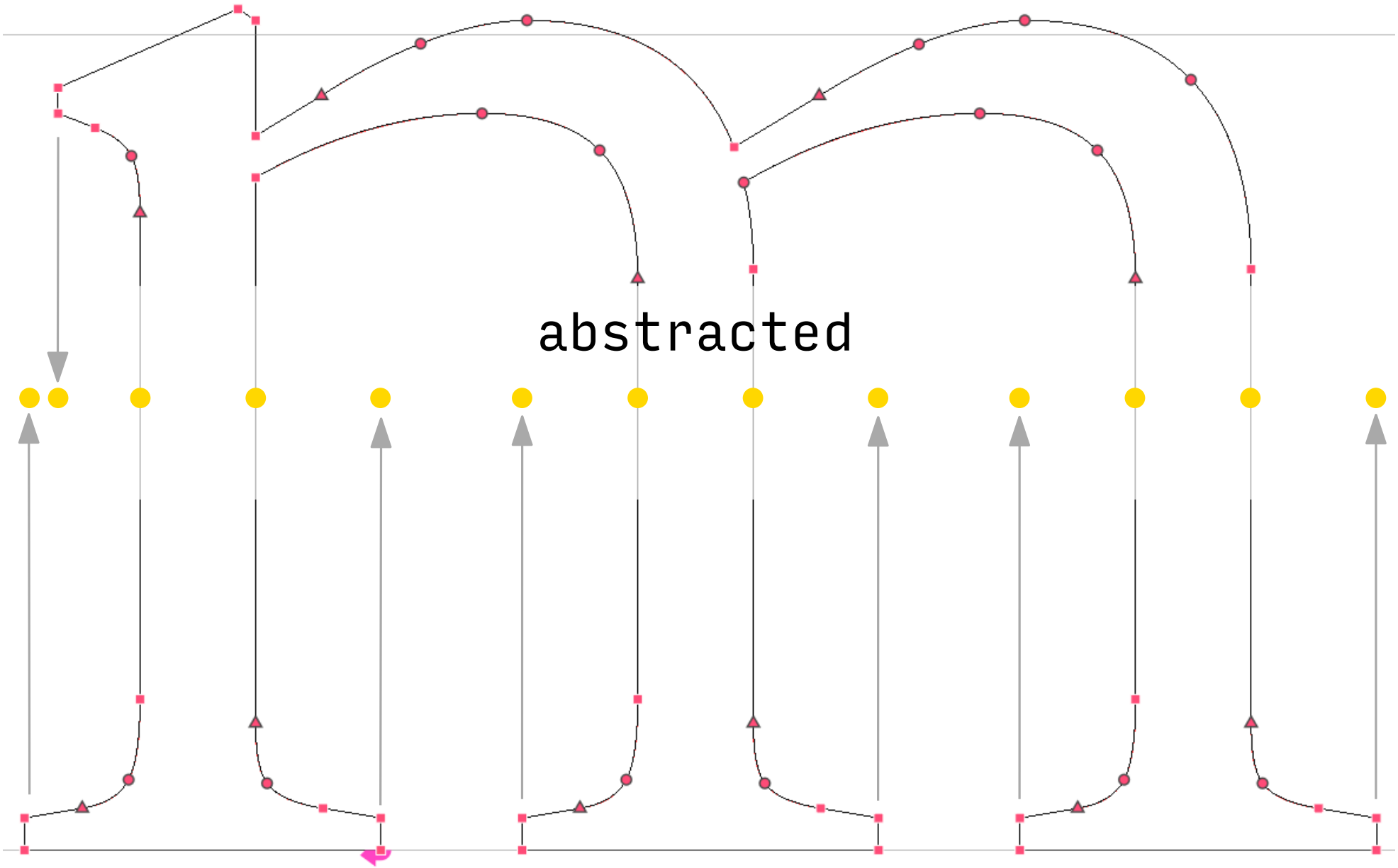


Alignments

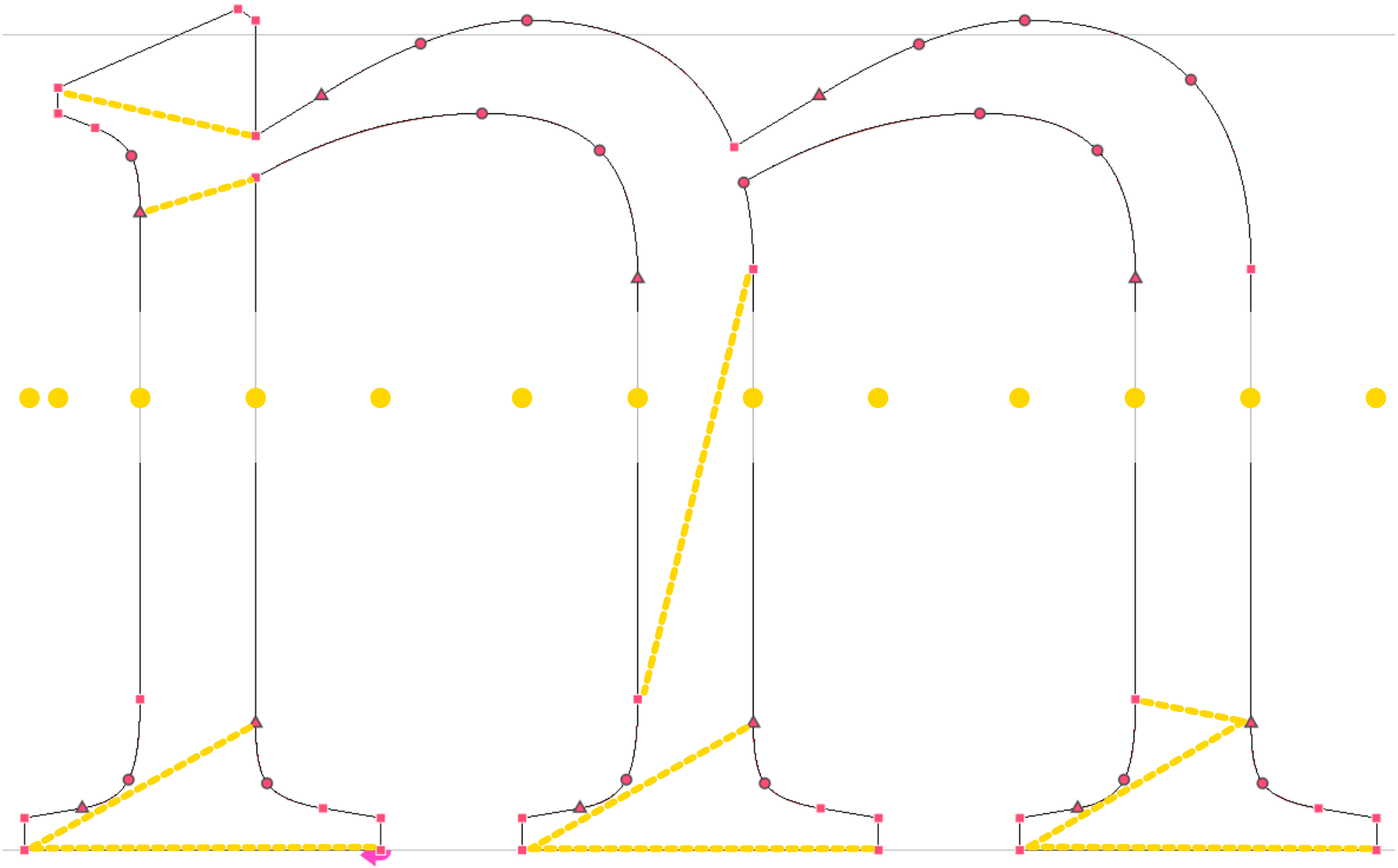
vertical alignments for horizontal hinting



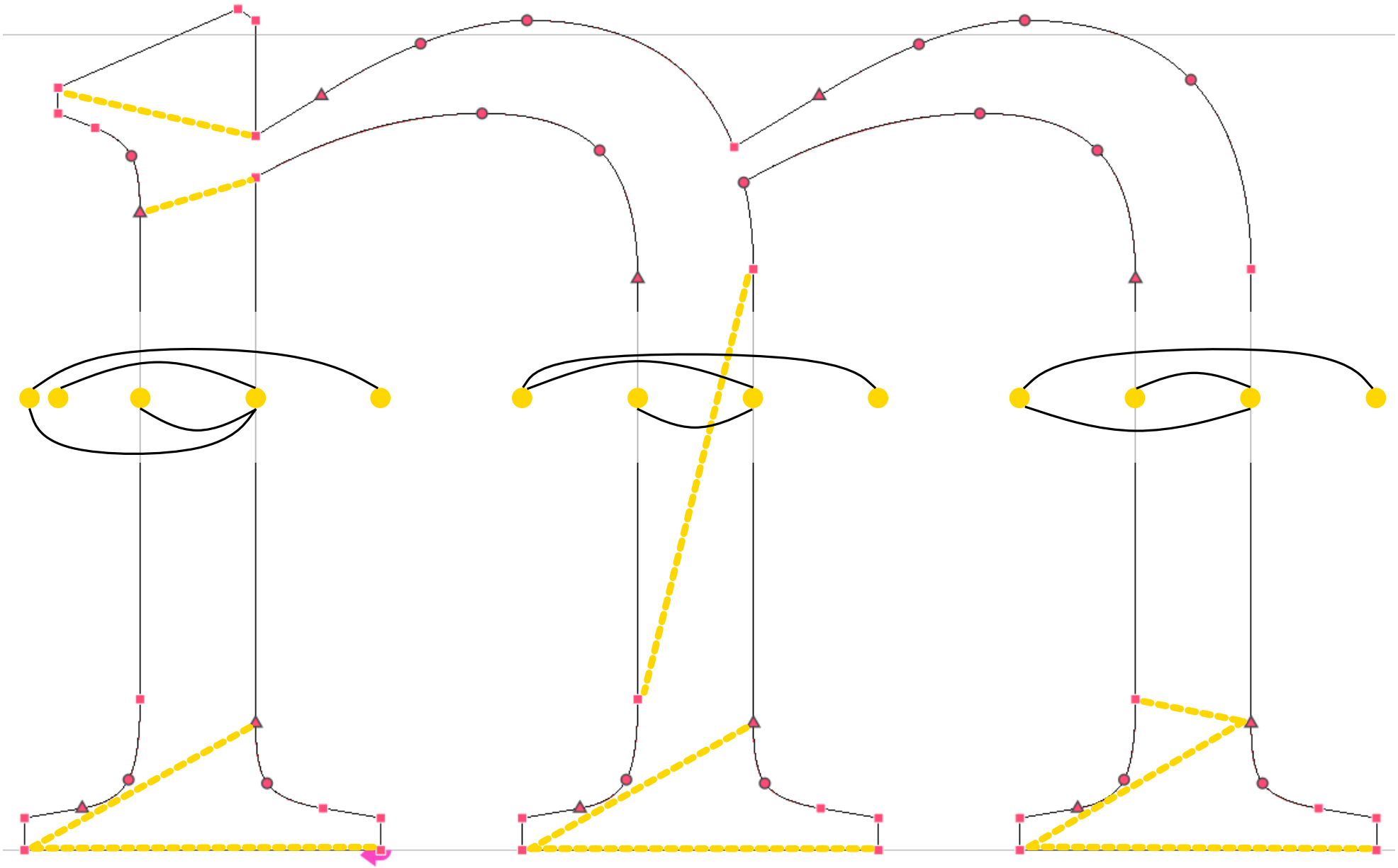
Alignments



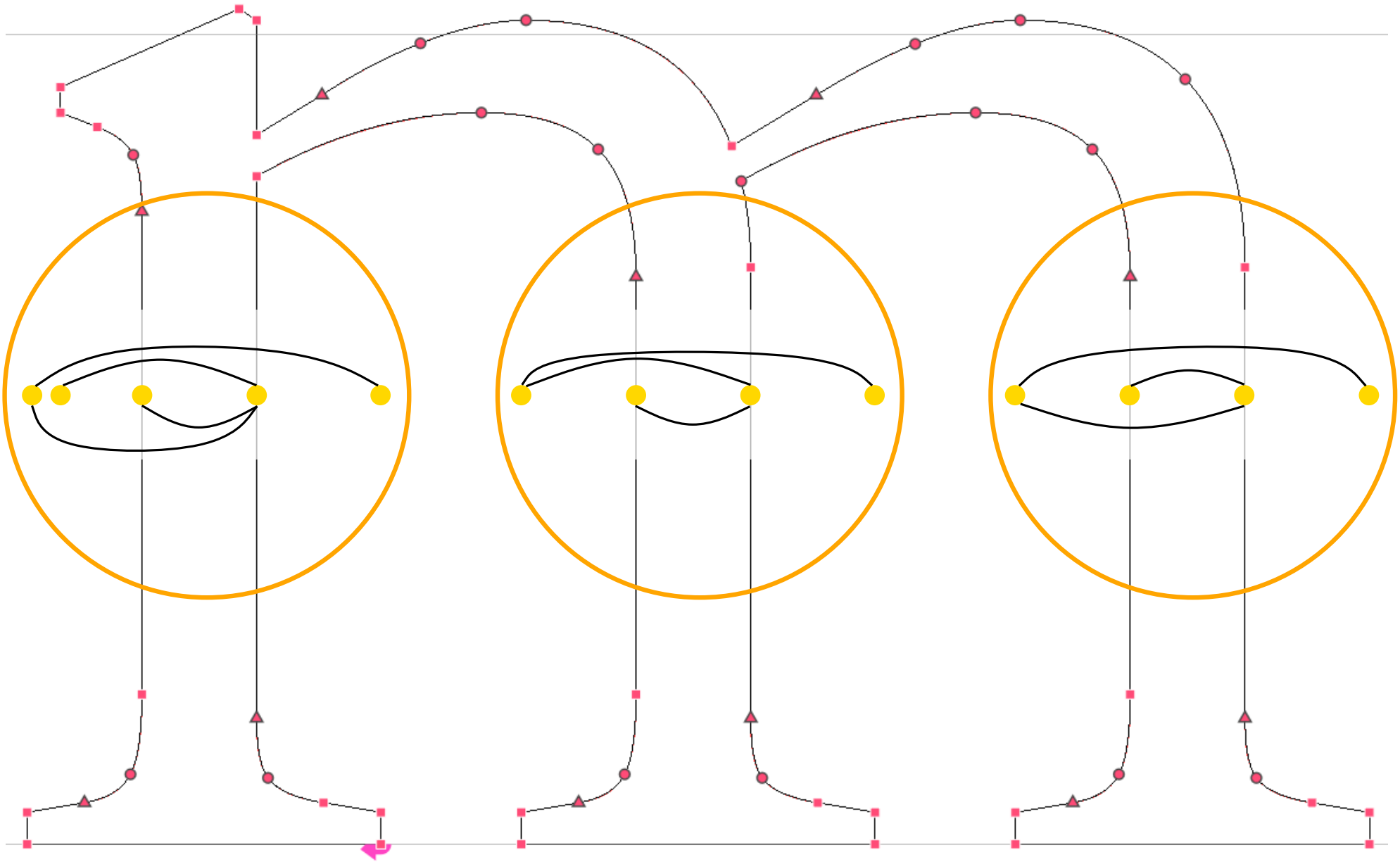
Groups



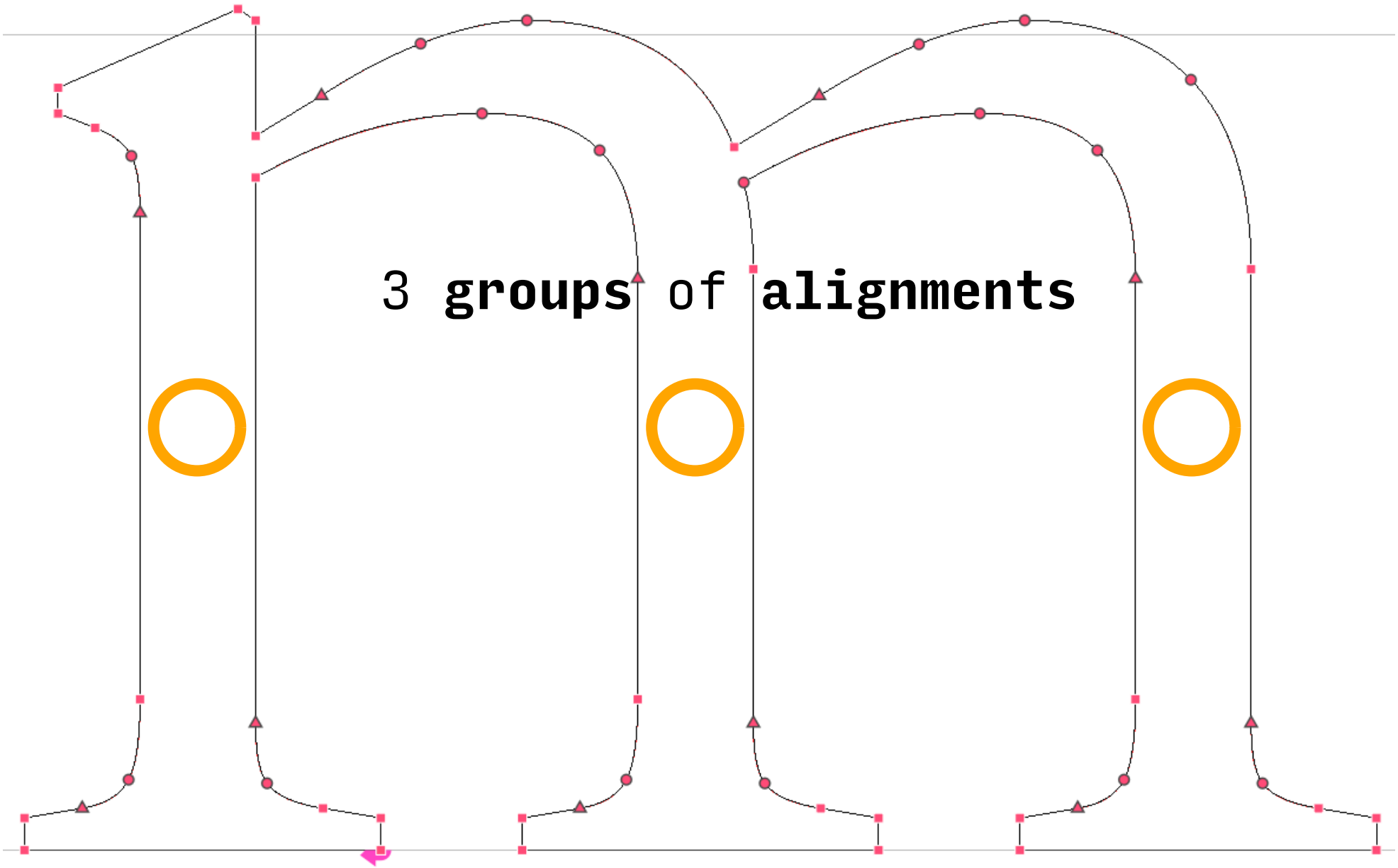
Groups



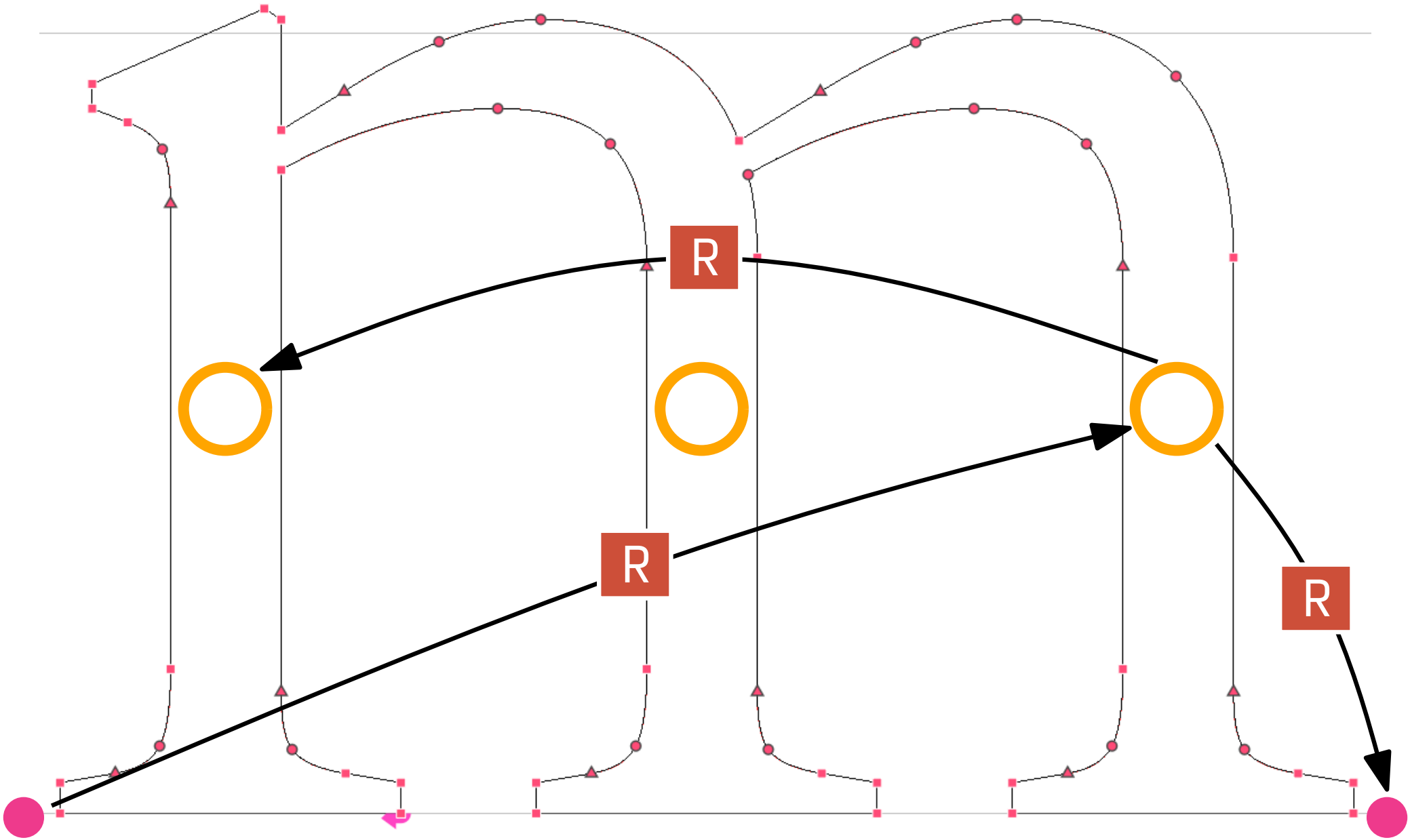
Groups



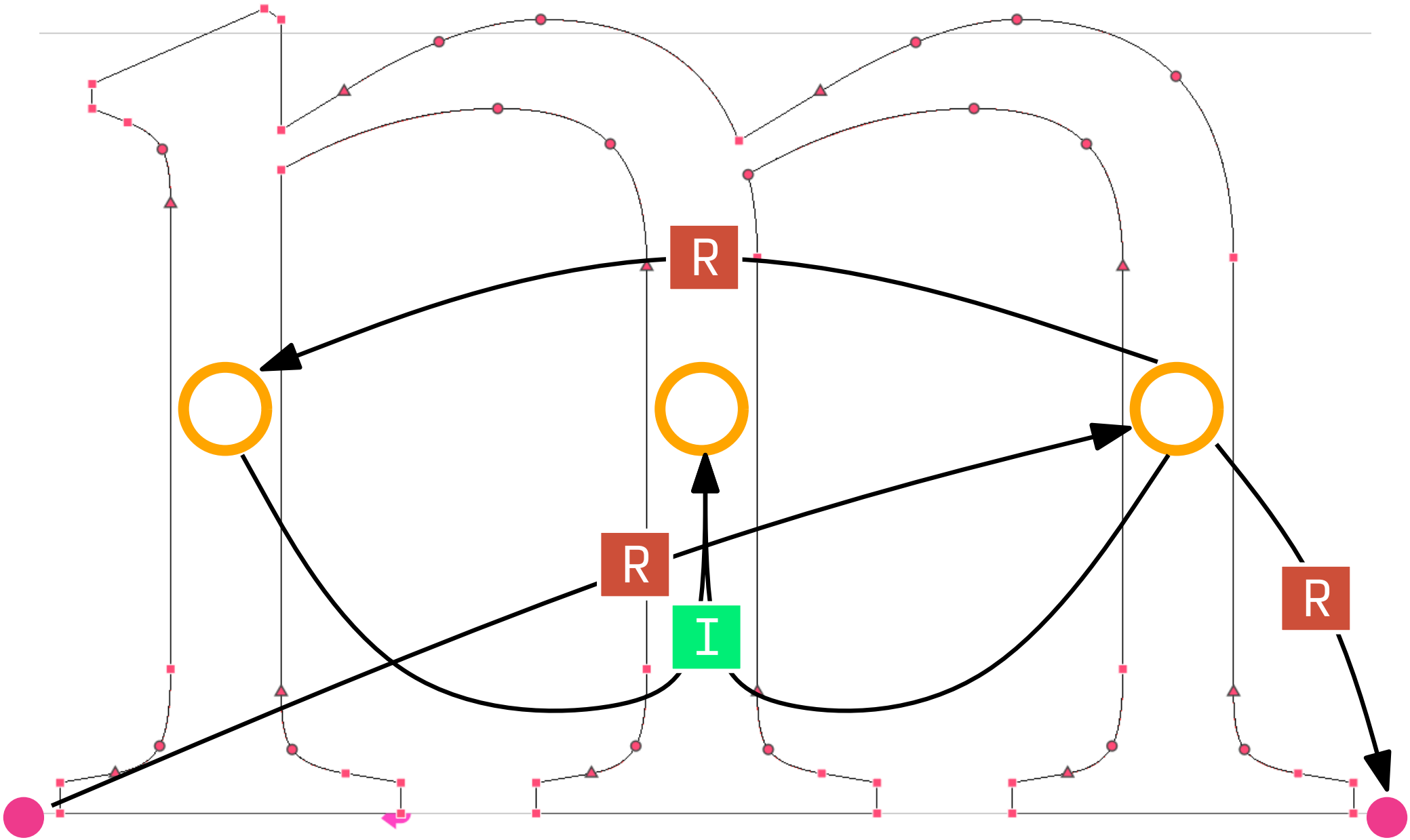
Groups



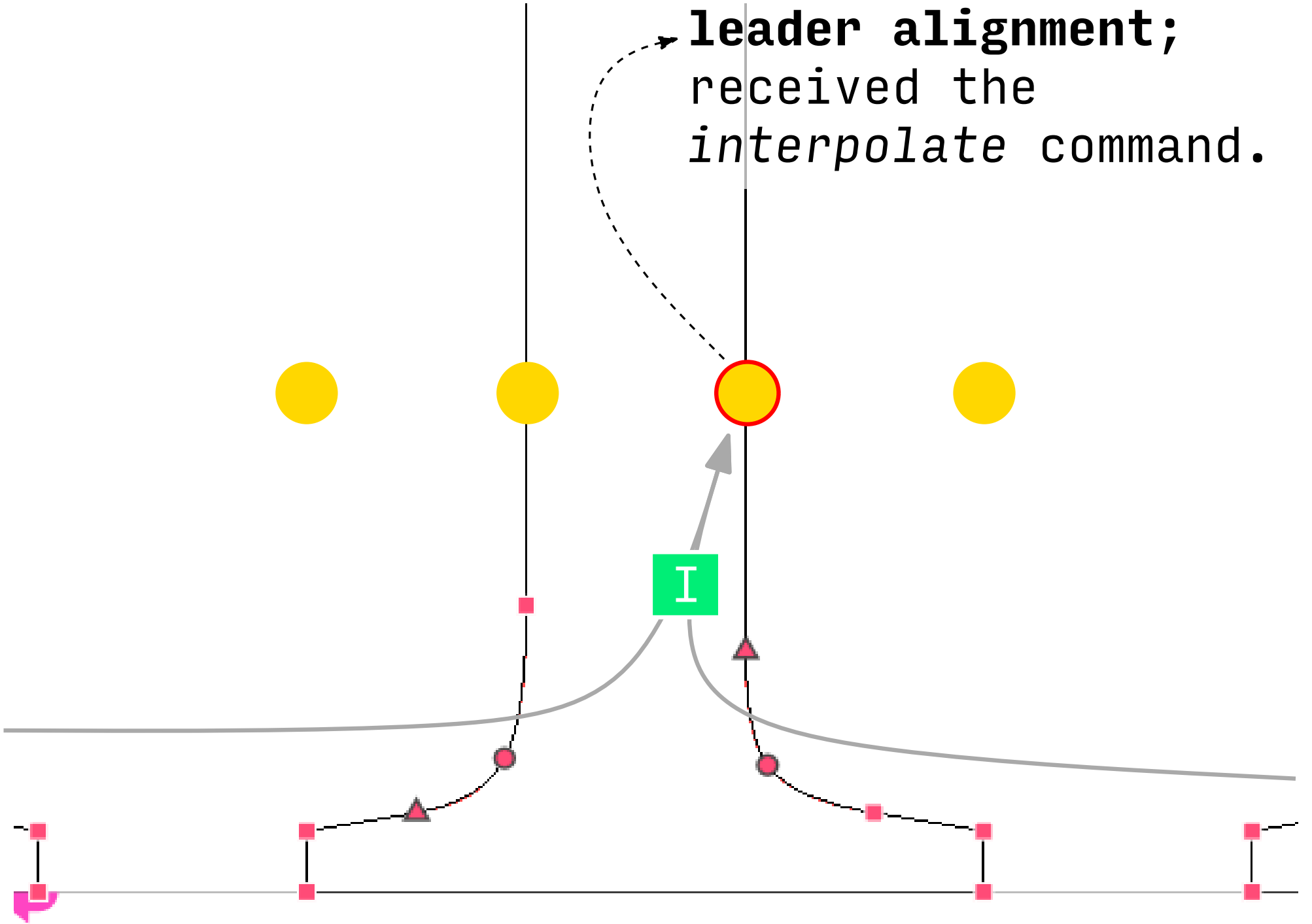
Hinting the groups (in X)



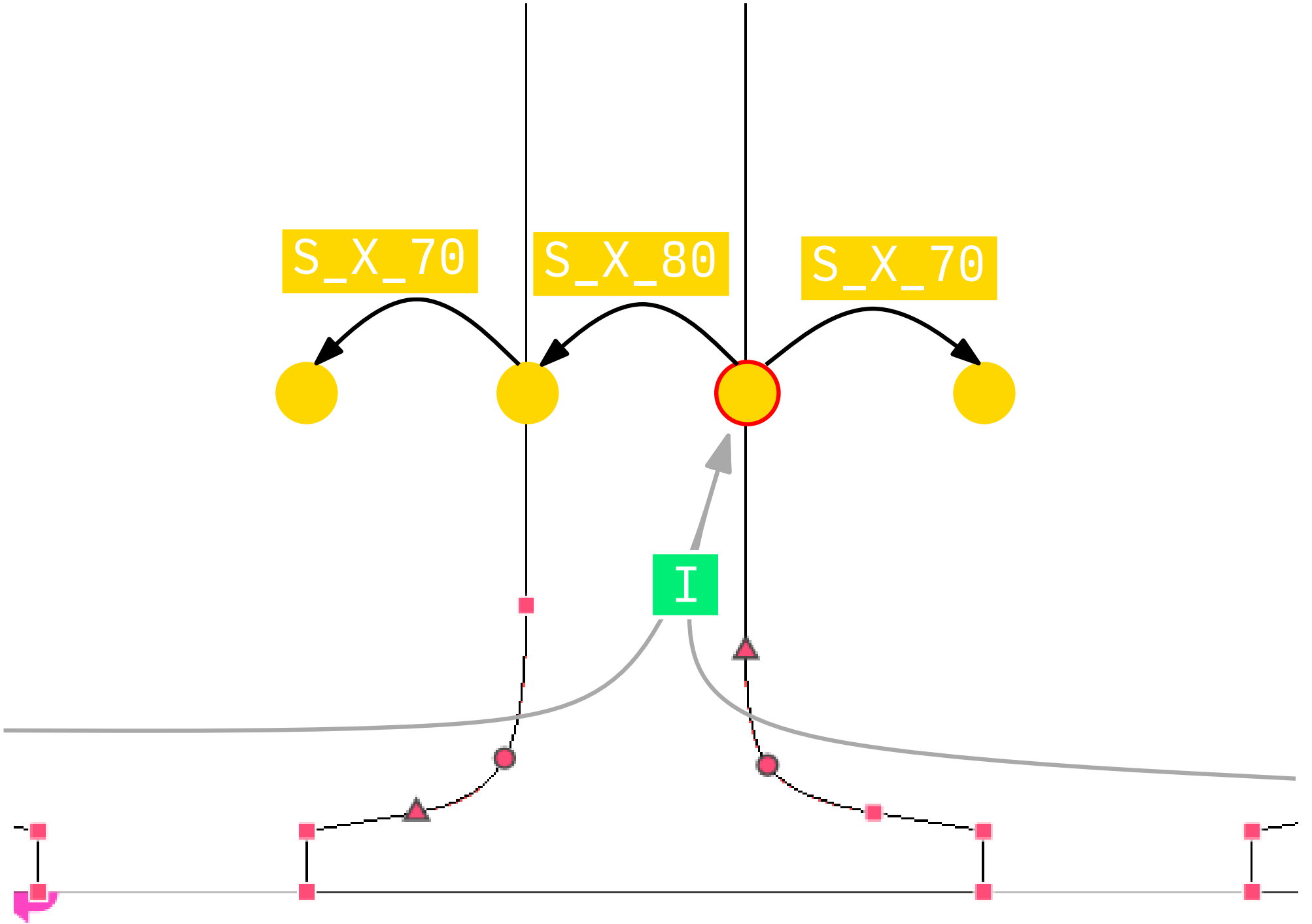
Hinting the groups (in X)



Hinting alignments in a group (in X)

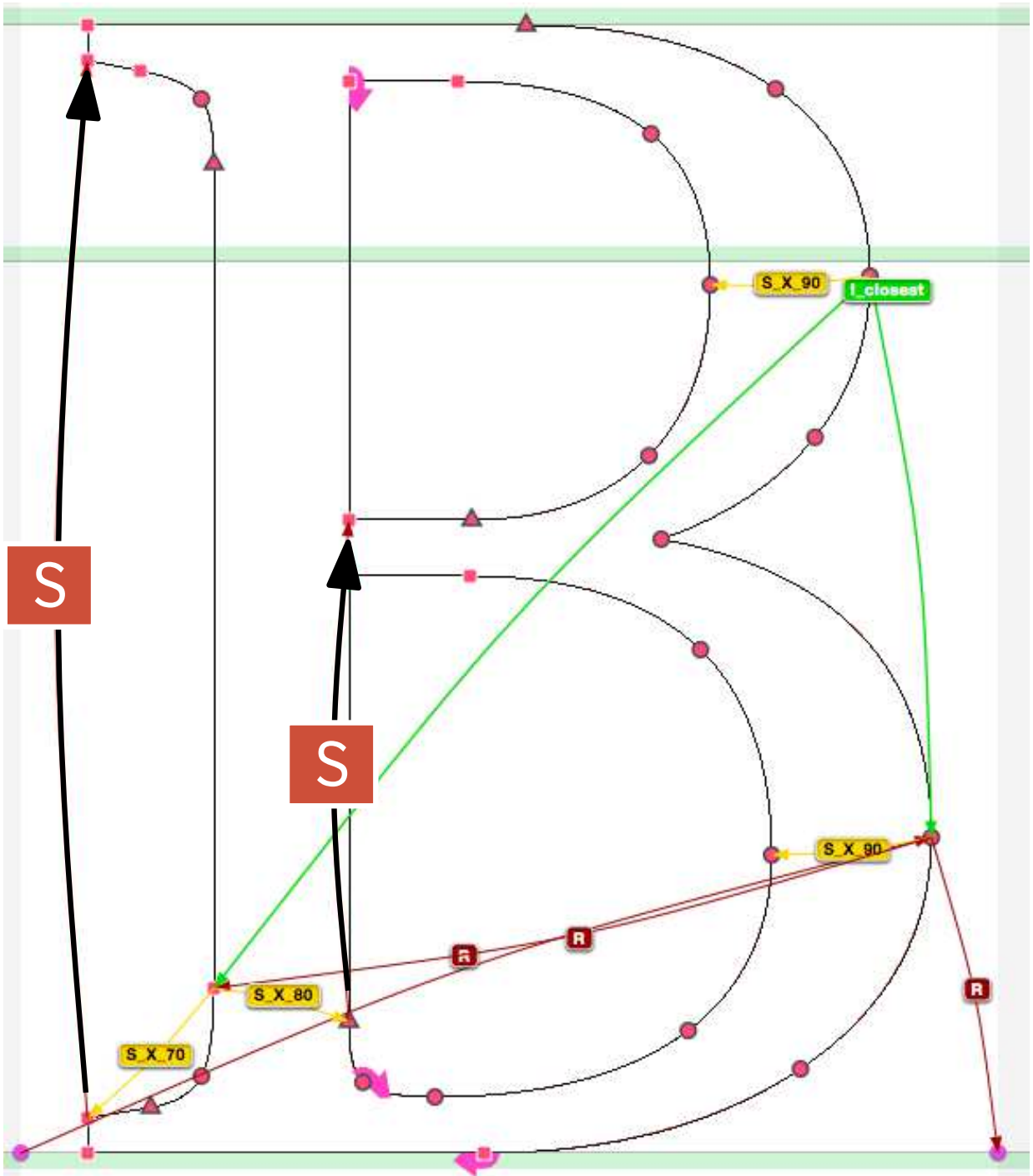


Hinting alignments in a group (in X)

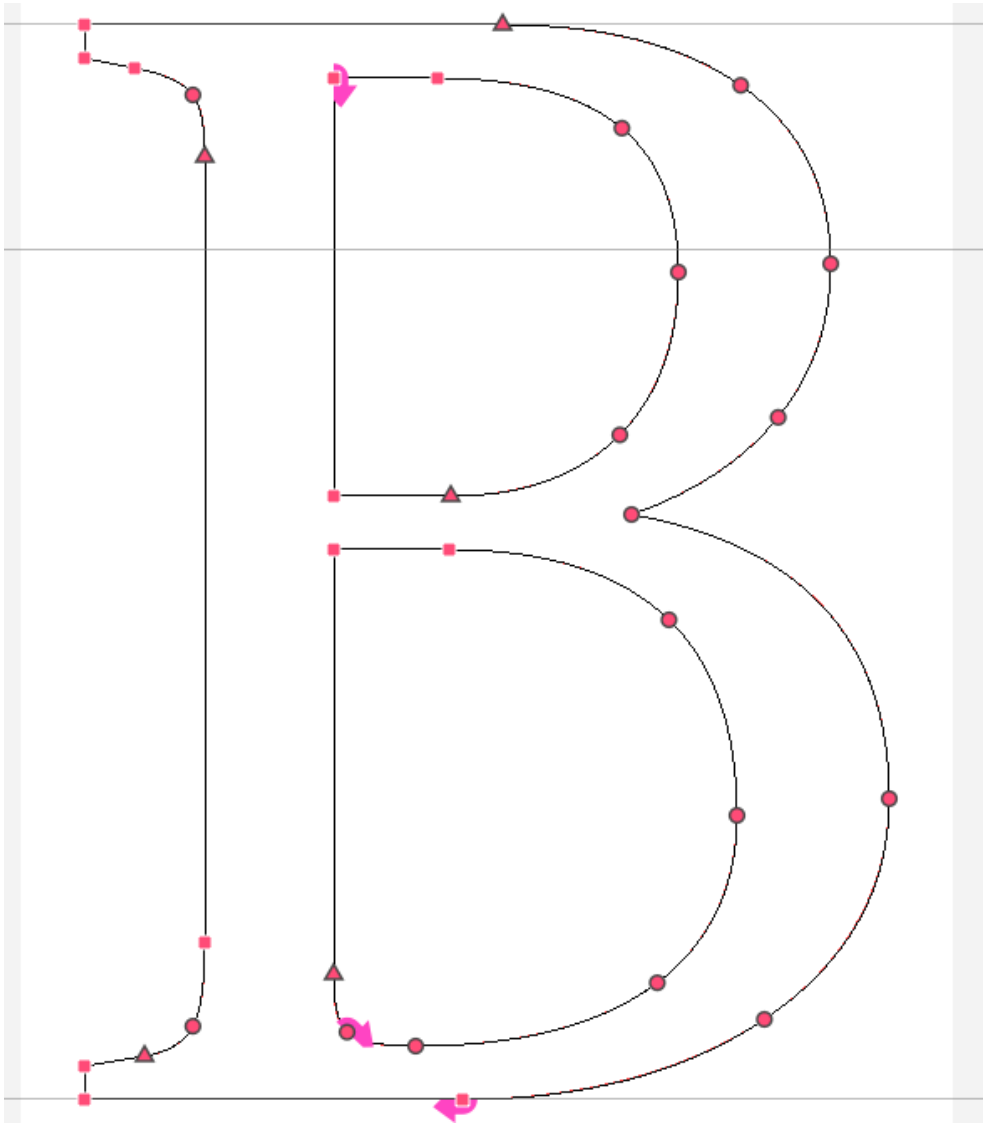


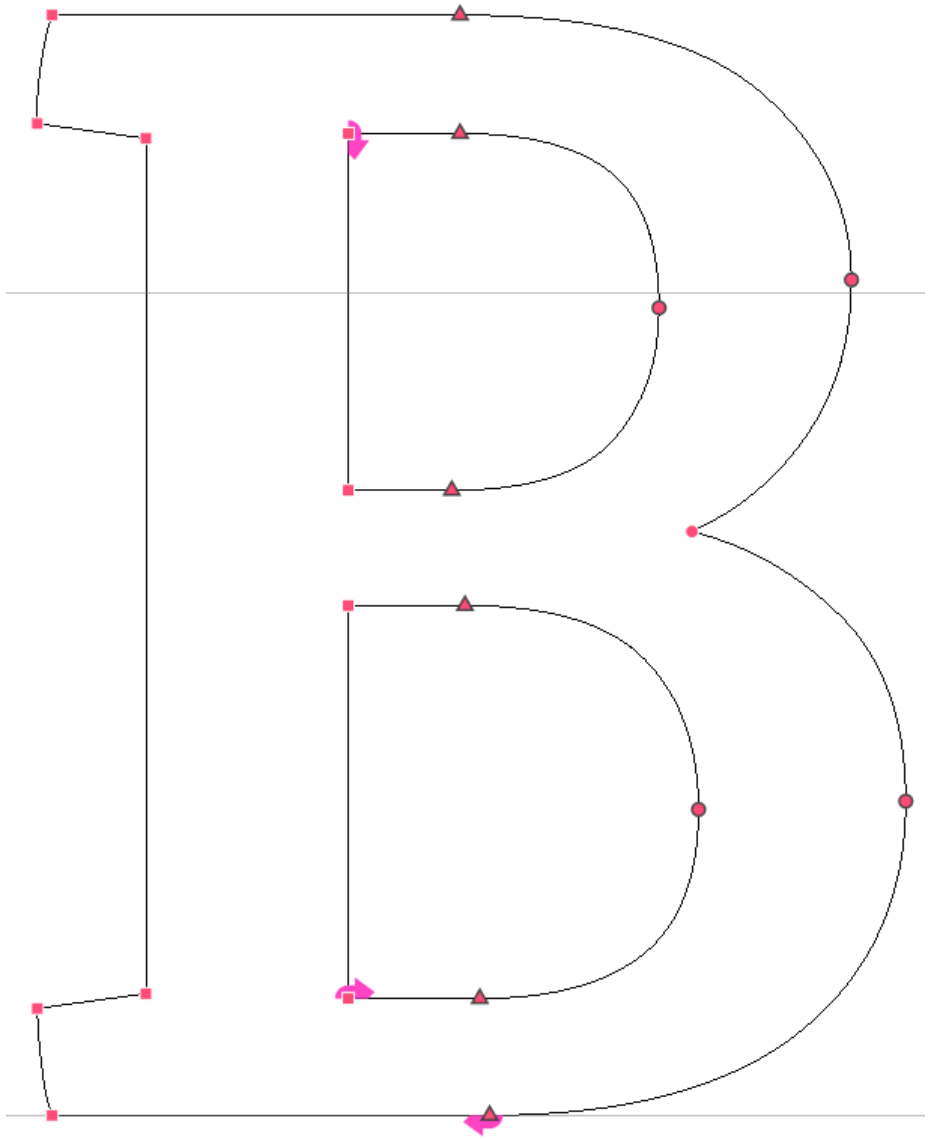
Hinting alignments in a group (in X)

Finally,
the points in an alignment are *single-link*'d.

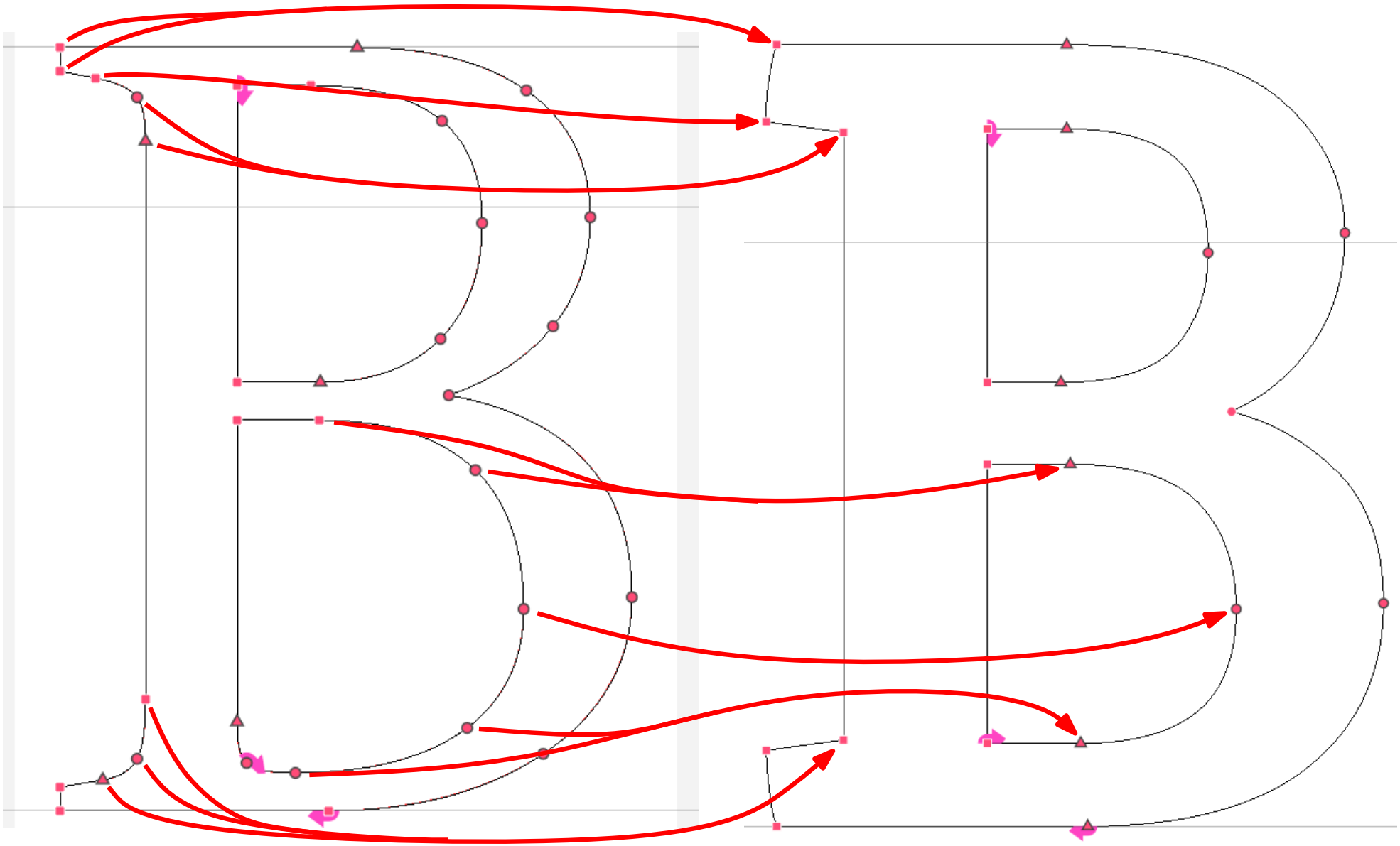


VGP transfer via contour matching

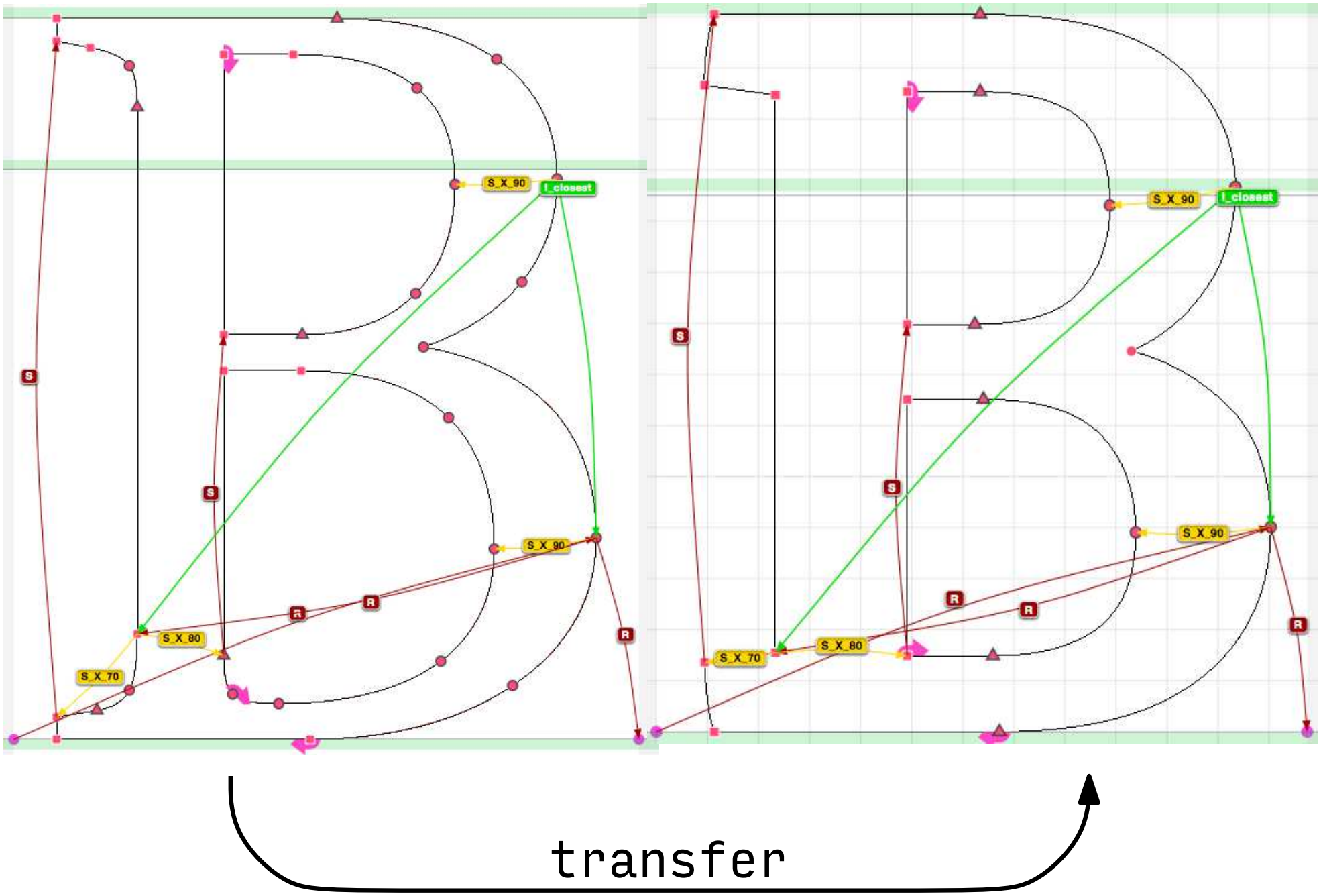




VGP transfer via contour matching



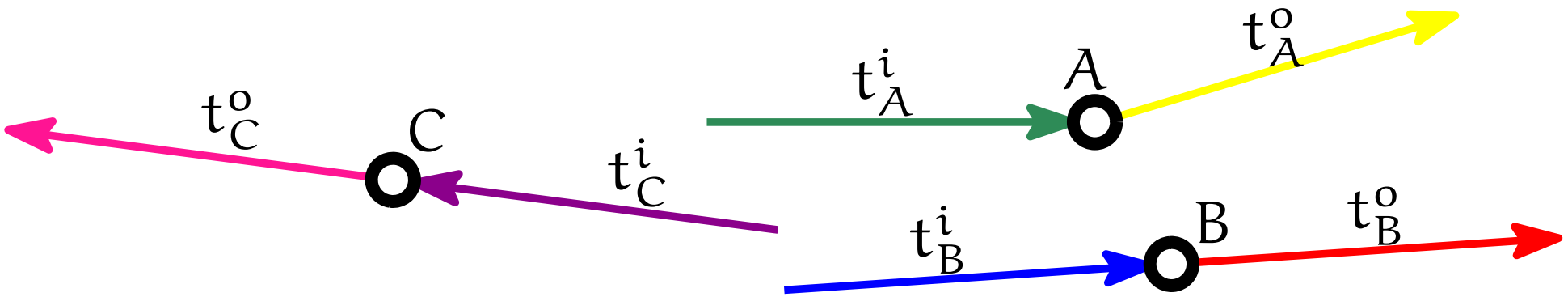
VGP transfer via contour matching



VGP transfer via contour matching

1. Measuring dissimilarity of 2 points
2. Matching 2 *pointed* contours
   (with a startpoint)
3. Matching 2 contours
4. Matching 2 glyphs

2 points dissimilarity



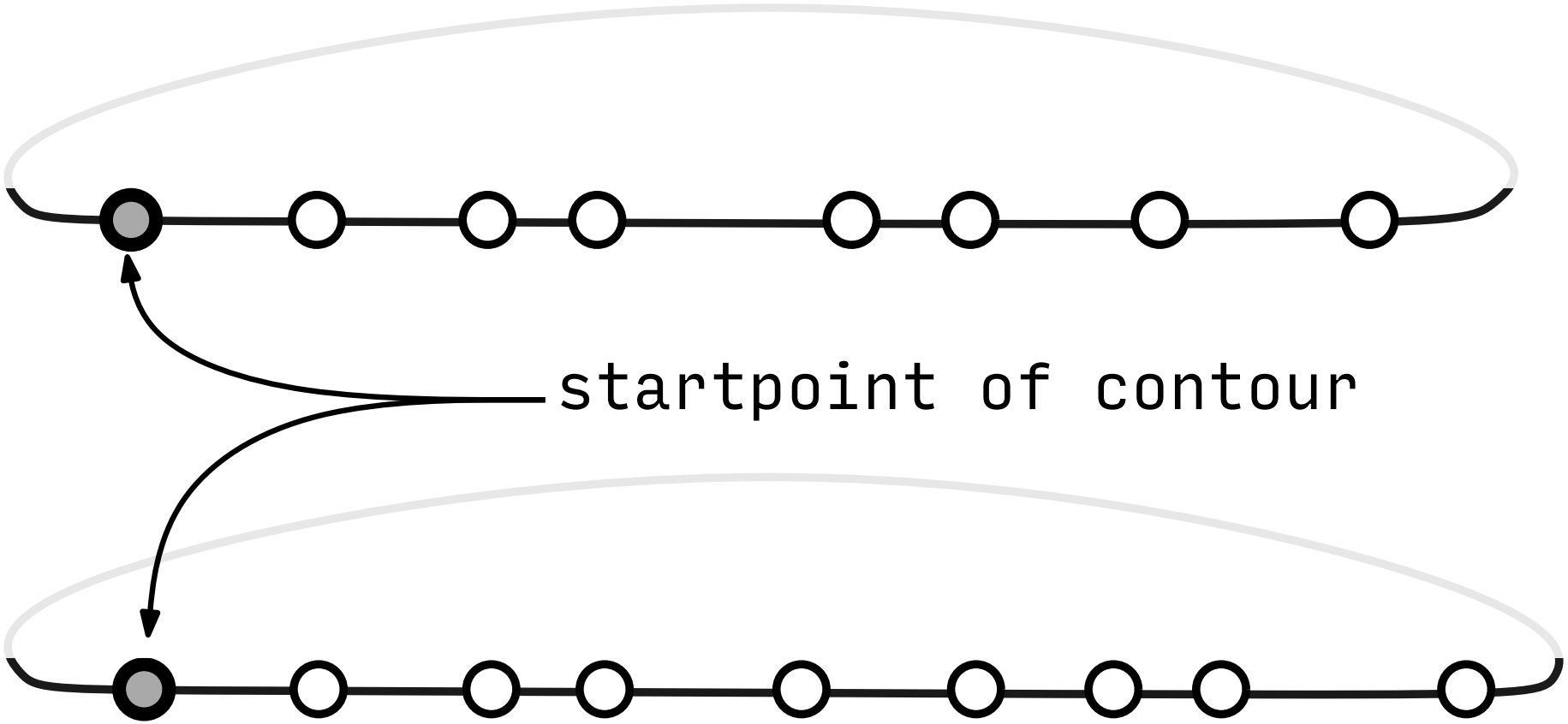
We choose $\mathcal{D}(A, B)$ so that it is small when A and B are close and have similar tangents.

$$\mathcal{D}(A, B) = \left(f(t_A^i, t_B^i) + f(t_A^o, t_B^o)\right) \times \|A - B\|^2$$

$$\text{where } f(t_A, t_B) = \frac{2}{1 + t_A \cdot t_B}$$

For example $\mathcal{D}(A, B) < \mathcal{D}(A, C)$

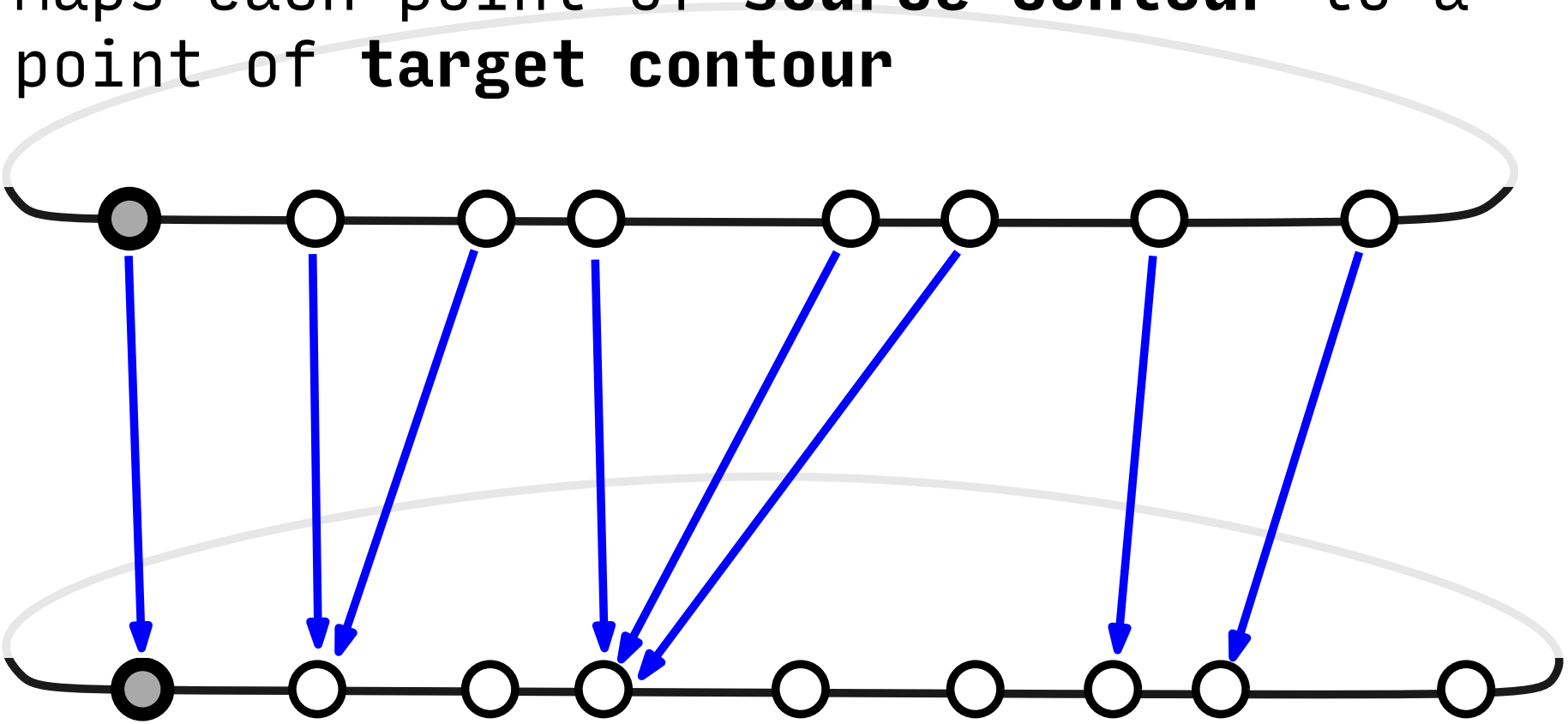
Matching two pointed contours



What is a matching?

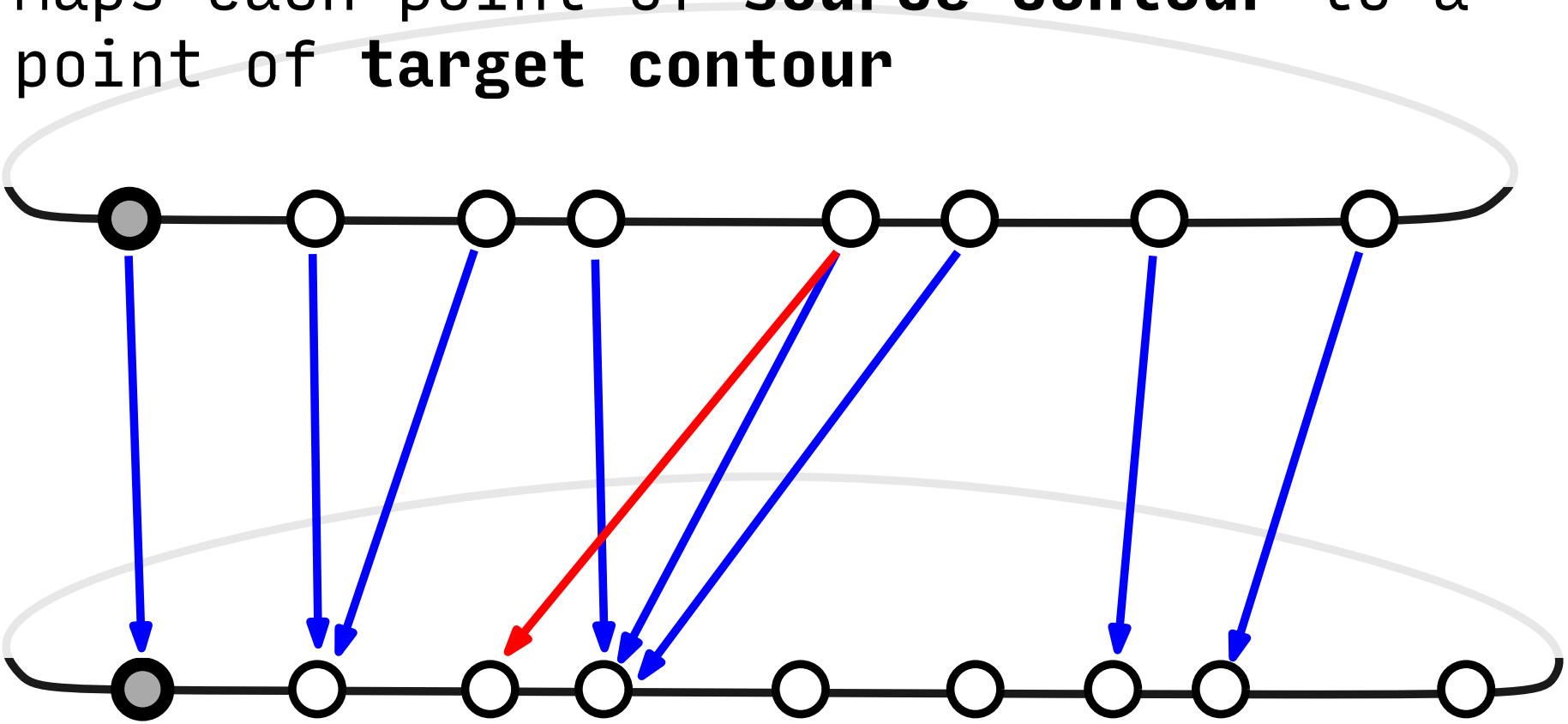
Matching two pointed contours

- Maps each point of **source contour** to a point of **target contour**



Matching two pointed contours

- Maps each point of **source contour** to a point of **target contour**

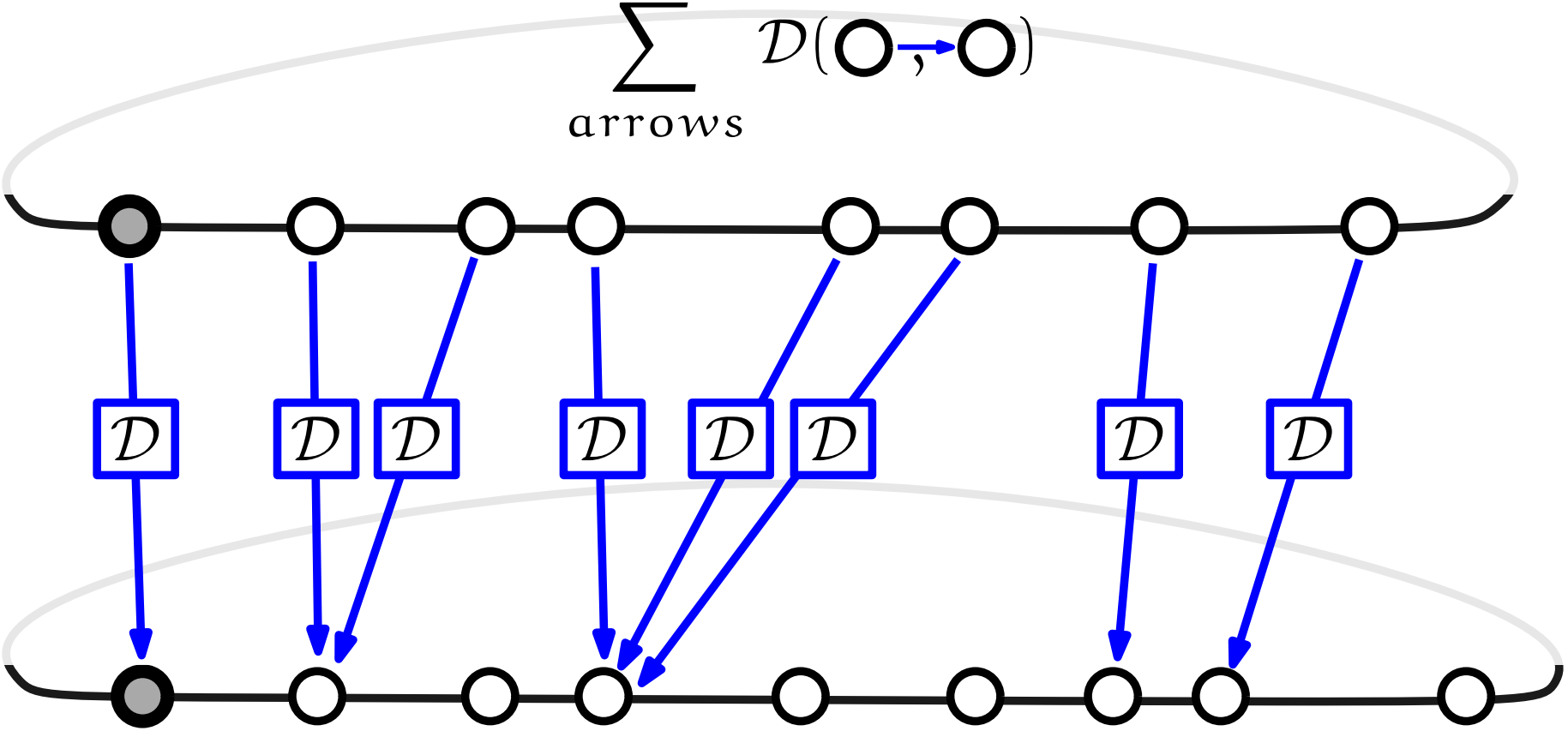


- Maps source startpoint to target startpoint
- Crossing arrows are forbidden

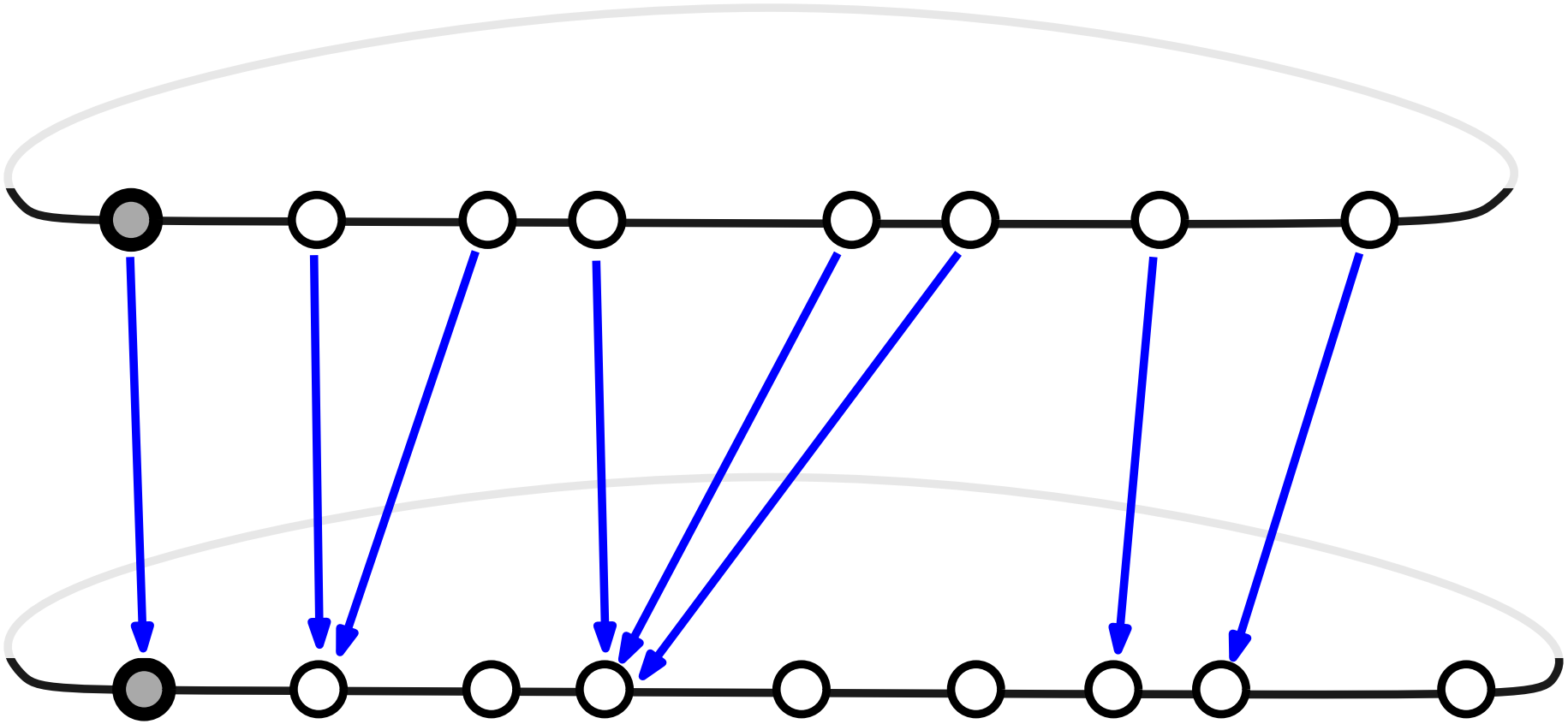
Matching two pointed contours

Score of this pointed matching is

$$\sum_{\text{arrows}} \mathcal{D}(\bullet \to \bullet)$$

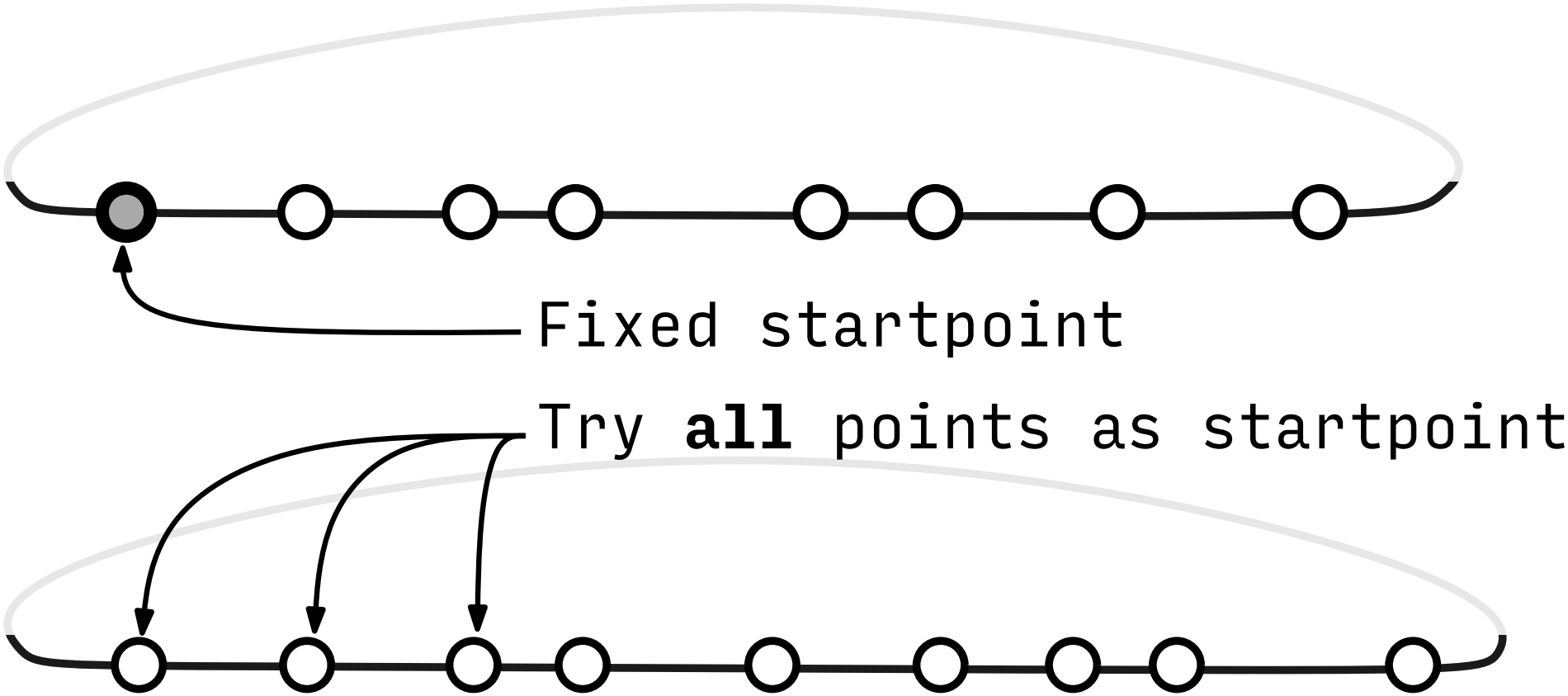


Matching two pointed contours

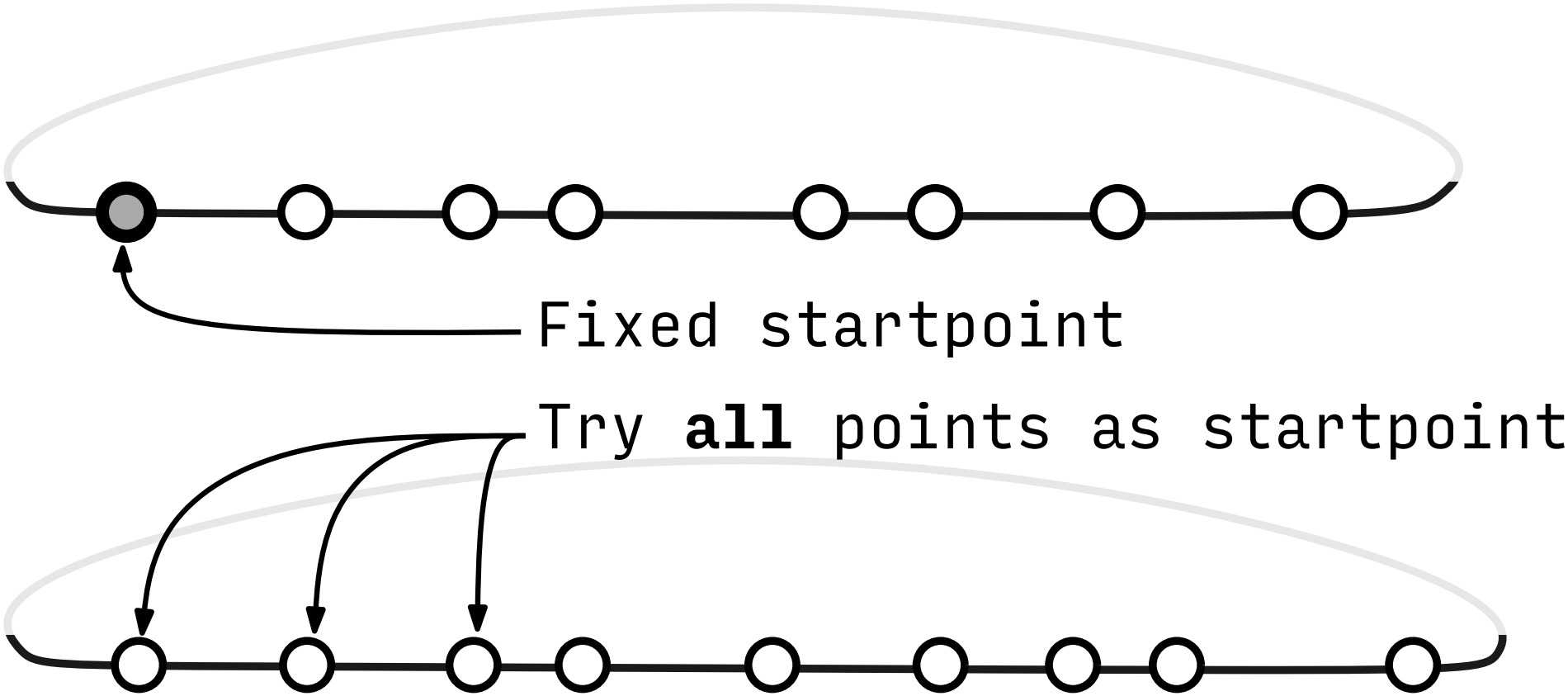


Find the **best pointed matching** (lowest score) with dynamic programming

Matching two contours

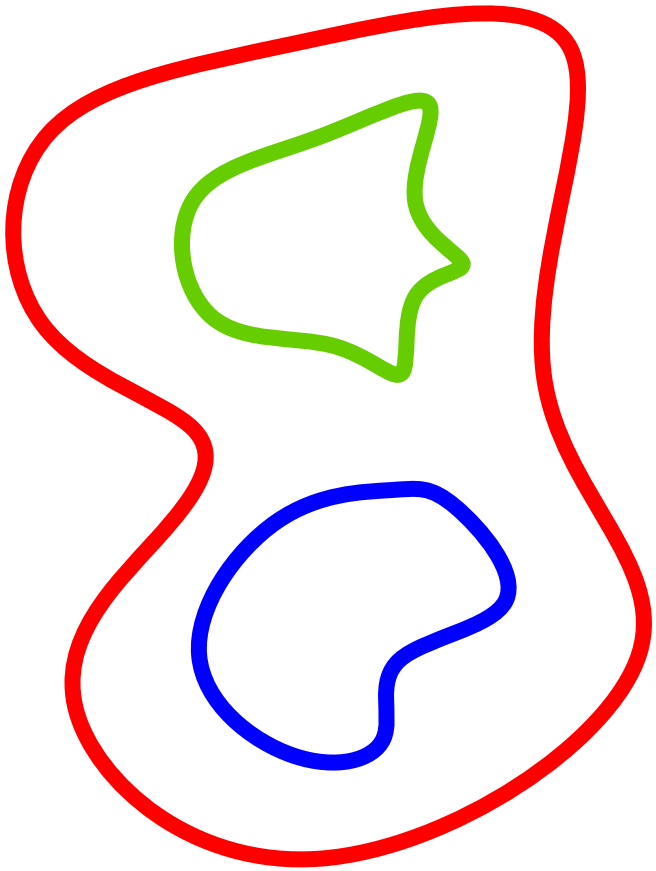


Matching two contours

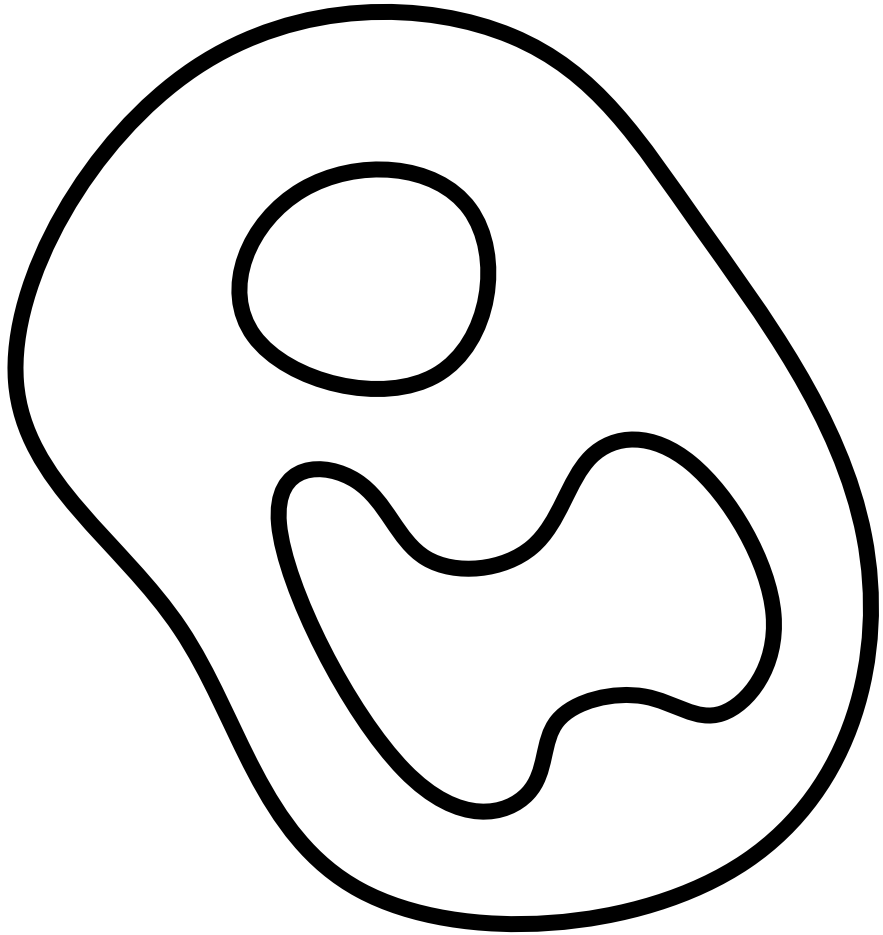


Find **best matching** by trying all pointed matchings and picking the best one

Matching two glyphs

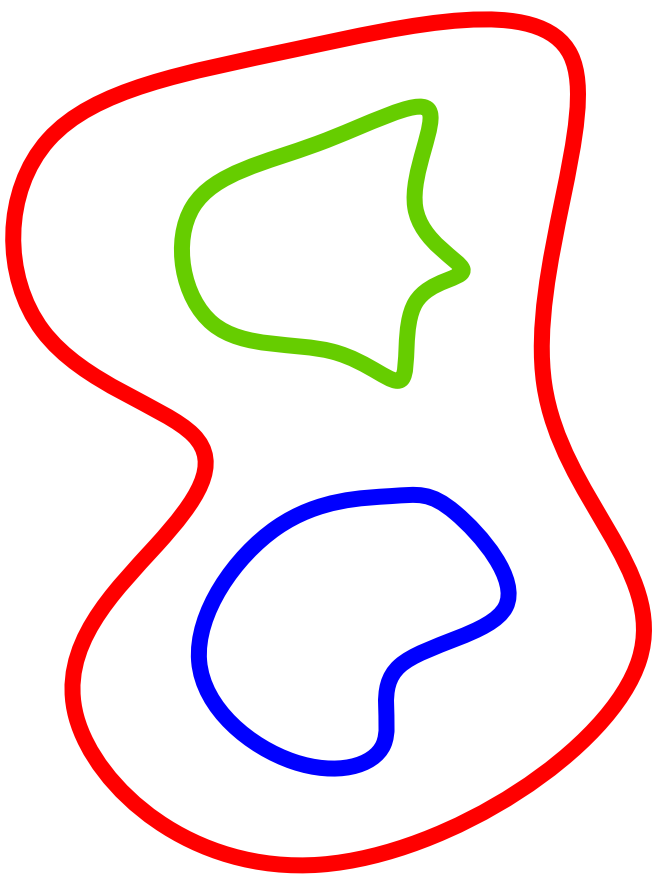


source glyph

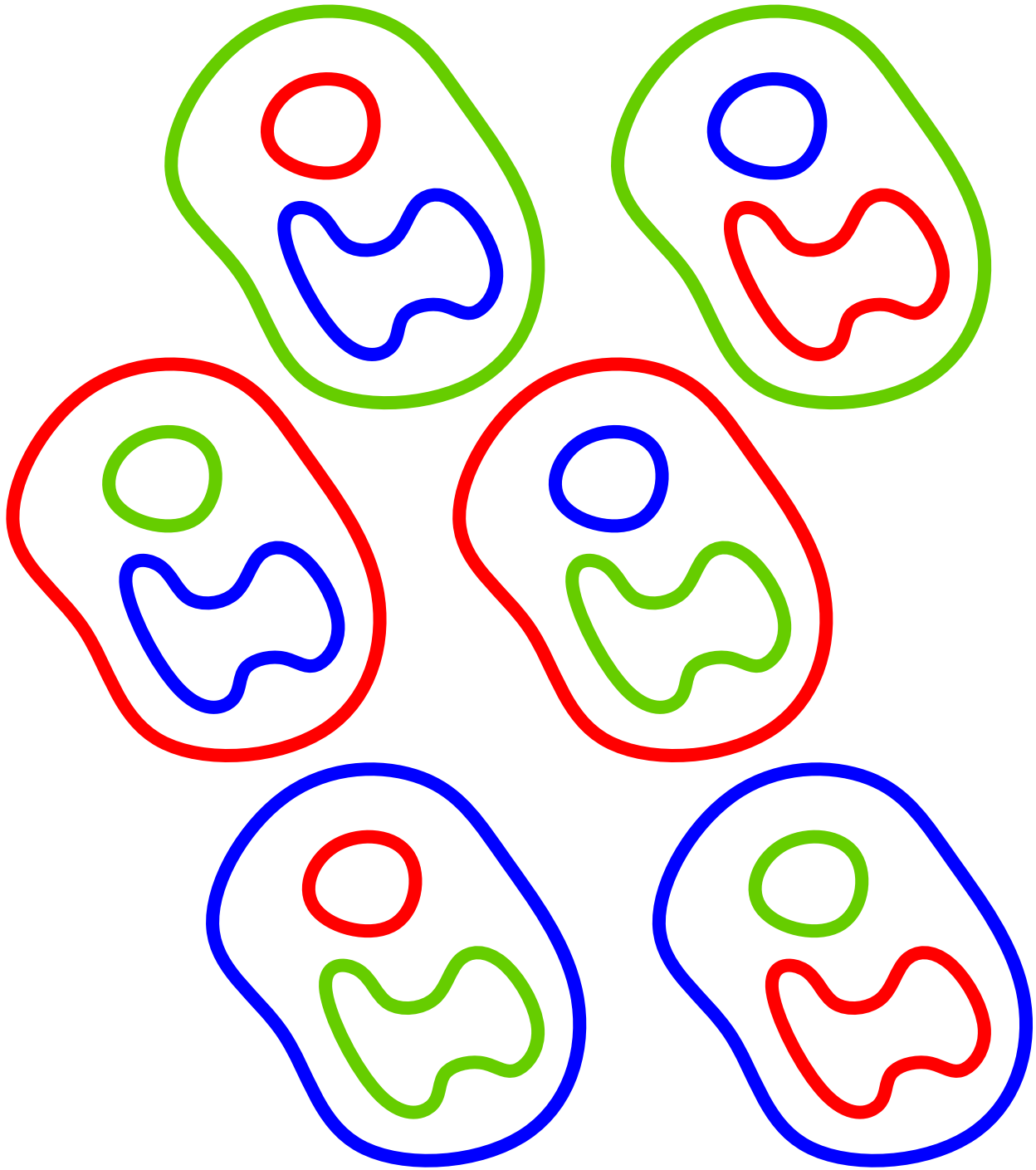


target glyph

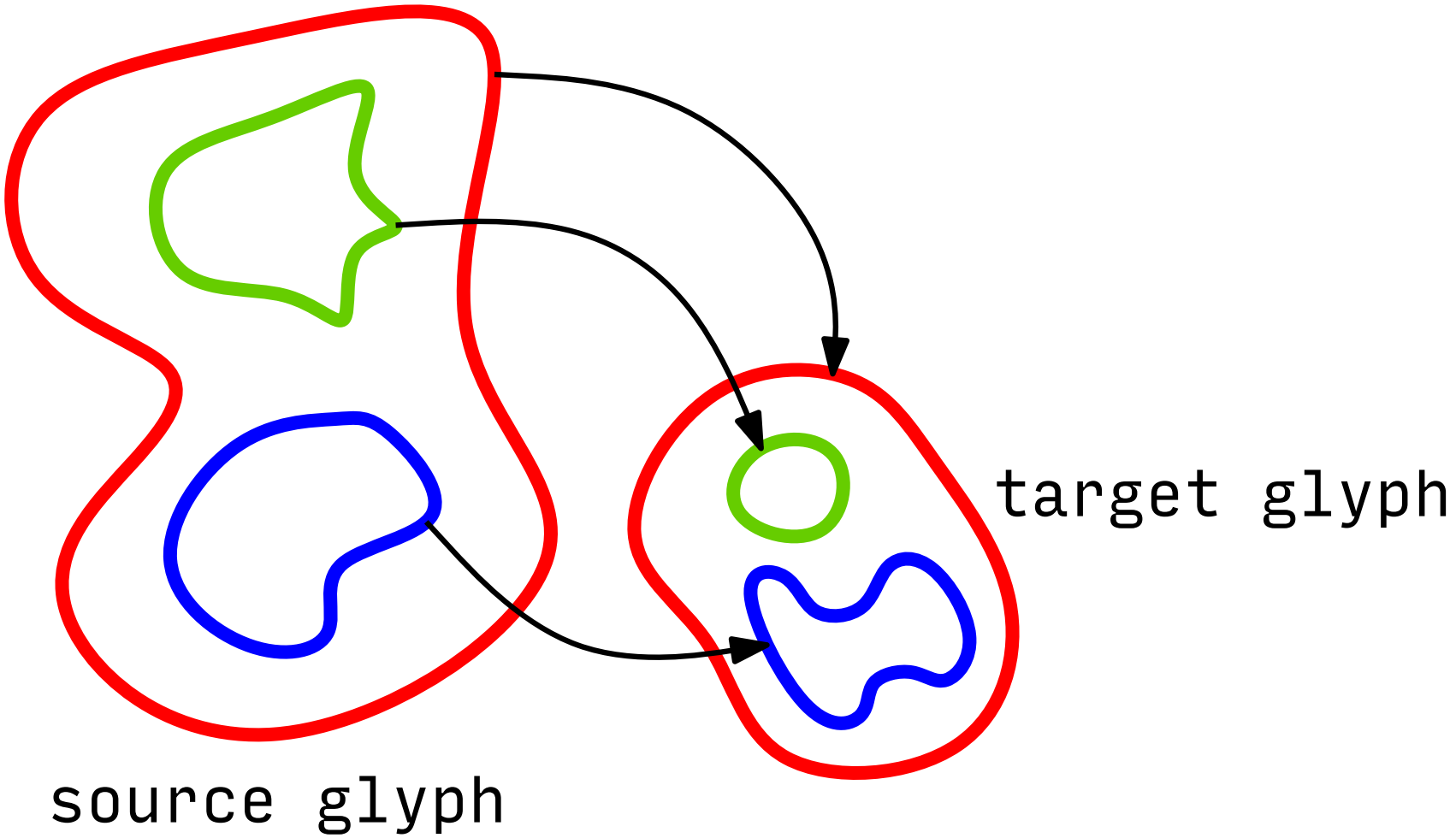
Matching two glyphs



source glyph



Matching two glyphs



VGP Transfer