L2/21-112

Issues in Devanagari cluster validation

Norbert Lindenberg
August 25, 2020

The Unicode Standard and the documentation of the OpenType Devanagari and Universal shaping engines don't agree on the definition of a valid Devanagari cluster, and Devanagari cluster validation in OpenType shaping engines and in fonts produces inconsistent results.

Contents

What's cluster validation?

In complex writing systems, it's not always obvious in which order characters that are pronounced together or that form a cluster in rendering should be stored in a Unicode character sequence. Glyphs are often rendered in a different sequence than the corresponding sounds in spoken language are pronounced, and commonly some glyphs are shown above or below other

glyphs, where the ordering is not clear. However, a defined character sequence is often important for correct processing of the text – sorting strings, searching for specific words, finding line breaks, and rendering the text using fonts.

Both the Unicode Standard and the OpenType script development documents therefore commonly define the structure of a valid cluster in a script. OpenType shaping engines for complex scripts usually validate incoming text as the first step in transforming a character sequence into a two-dimensional arrangement of glyphs, and insert a dotted circle ◌ whenever they find a character where they don't expect it. Fonts implemented using the Graphite and Apple Advanced Typography shaping systems have to implement validation themselves.

Unfortunately the Unicode Standard and the OpenType script development documents don't always agree with each other on the structure of valid clusters, or even have internal inconsistencies, and shaping engine implementations don't always follow either of these documents.

What's a valid Devanagari cluster?

Defining a cluster model for Devanagari is complicated by the fact that Devanagari has been used over a very long time (close to 2000 years) and for over 200 languages (SIL). Documentation about the script tends to focus on contemporary use for a few popular languages such as Hindi, Marathi, and Nepali, and on historical use for Sanskrit. It's easy to derive from such documentation a set of character combinations that must be supported, but it's not sufficient to derive which combinations should be prohibited. This document therefore compares current documentation of cluster models in the Unicode standard and OpenType documentation as well as a range of implementations to find disagreements that need to be investigated and resolved.

Characters used in and with the Devanagari script

Based on the the core specification and the data for Unicode 13.0, this document uses the following Devanagari character set:

- The characters U+0900..U+0952 and U+0955..U+097F in the Unicode block "Devanagari". These characters have the Script property value "Devanagari". Two characters in this block, U+0953 DEVANAGARI GRAVE ACCENT and U+0954 DEVANAGARI ACUTE ACCENT, should no longer be used.
- The characters in the Unicode block "Devanagari Extended", U+A8E0 – U+A8FF. These characters have the Script property value "Devanagari".
- The characters U+1CD0 – U+1CF6 and U+1CF8 – U+1CF9 in the Unicode block "Vedic Extensions". These characters have the Script property value "Common" or "Inherited" and a Script_Extensions property value that includes the script code "Deva". Two characters in this block, U+1CF7 and U+1CFA, do not include "Deva" in their Script_Extensions property values and are therefore excluded.
- The characters in the Unicode block "Common Indic Number Forms", U+A830 – U+A839. These characters have the Script property value "Common" and a

Script_Extensions property value that includes the script code “Deva”.

- The character U+20F0 COMBINING ASTERISK ABOVE. This character has the Script property value “Inherited” and a Script_Extensions property value that includes the script code “Deva”.
- The characters U+002C COMMA, U+002E FULL STOP, U+02BC MODIFIER LETTER APOSTROPHE, U+200C ZERO WIDTH NON-JOINER, and U+200D ZERO WIDTH JOINER. These characters have the Script property value “Common” or “Inherited” and are mentioned in the Devanagari section of the Unicode Standard as being used with Devanagari. U+0300 COMBINING GRAVE ACCENT and U+0301 COMBINING ACUTE ACCENT are also mentioned in the chapter, but apparently only intended for use with Latin.
- The characters U+00A0 NO-BREAK SPACE and U+25CC DOTTED CIRCLE. These characters have the Script property value “Common” and are commonly used as bases to show stand-alone marks in any script.

Unicode character properties for Devanagari

The Unicode Standard provides several character properties that can help describe the structure of Devanagari clusters. The Universal Shaping Engine (USE), part of OpenType rendering systems, uses these properties to define character classes, which are used in its definition of (generic Brahmic) clusters. The following table shows the Devanagari characters identified above and their properties and classes.

Code points	Characters	General category	Canonical combining class	Indic syllabic category	Indic positional category	USE subclass
0971	◌	Lm	0	Other	NA	BASE_IND
1CF2..1CF3	◌	Lo	0	Consonant_Dead	NA	BASE_IND
0950, A8F4..A8F7, A8FB, A8FD	◌	Lo	0	Other	NA	BASE_IND
1CED	◌	Mn	220	Other	Bottom	BASE_IND
1CE2..1CE8	◌	Mn	1	Other	Overstruck	BASE_IND
002C, 002E, 0964..0965, 0970, 1CD3, A8F8..A8FA, A8FC	, . । ॥ ॰ ꣽ ꣾ ◌	Po	0	Other	NA	BASE_IND

02BC	’	Lm	0	Other	NA	OTHER
1CE9..1CEC, 1CEE..1CF1	ᳩ ᳪ ᳫ ᳬ ᳭ ᳮ ᳯ ᳰ ᳱ	Lo	0	Other	NA	OTHER
A830..A835	꠰ ꠱ ꠲ ꠳ ꠴ ꠵	No	0	Other	NA	OTHER
1CF5..1CF6	ᳵ ᳶ	Lo	0	Consonant_With_Stacker	NA	CONS_WITH_STACKER
093D	ऽ	Lo	0	Avagraha	NA	BASE
A8F2..A8F3	ꣲ ꣳ	Lo	0	Bindu	NA	BASE
0915..0939, 0958..095F, 0978..097F	क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ द ध न ऩ प फ ब भ म य र ऱ ल ळ ऴ व श ष स ह क़ ख़ ग़ ज़ ड़ ढ़ फ़ य़ ऱ ज़ ष ग़ ज़ ॻ ज़ ड़ ब़	Lo	0	Consonant	NA	BASE
0904..0914, 0960..0961, 0972..0977, A8FE	ॲ अ आ इ ई उ ऊ ऋ ऌ ऍ ऎ ए ऐ ऑ ऒ ओ औ ॠ ॡ ॵ अं अः औ ॳ ॴ ऍै	Lo	0	Vowel_Independent	NA	BASE
0966..096F	० १ २ ३ ४ ५ ६ ७ ८ ९	Nd	0	Number	NA	BASE
25CC	◌	So	0	Consonant_Placeholder	NA	BASE_OTHER
00A0		Zs	0	Consonant_Placeholder	NA	BASE_OTHER
093C	़	Mn	7	Nukta	Bottom	CONS_MOD_BELOW
094D	्	Mn	9	Virama	Bottom	HALANT
093F, 094E	ि ॎ	Mc	0	Vowel_Dependent	Left	VOWEL_PRE
093A, 0945..0948, 0955, A8FF	ं ॅ ॆ े ै ॕ ॏ	Mn	0	Vowel_Dependent	Top	VOWEL_ABOVE

0941..0944, 0956..0957, 0962..0963	◌ু ◌ূ ◌ৃ ◌ৄ ◌ৢ ◌ৣ	Mn	0	Vowel_Dependent	Bottom	VOWEL_BELOW
093B, 093E, 0940, 0949..094C, 094F	◌ा ◌ॉ ◌ी ◌ॊ ◌ो ◌ौ ◌ॏ	Mc	0	Vowel_Dependent	Right	VOWEL_POST
0900..0902	◌ँ ◌ं ◌ः	Mn	0	Bindu	Top	VOWEL_MOD_ABOVE
0951, 1CD0..1CD2, 1CDA..1CDB, 1CE0, 1CF4, 20F0, A8E0..A8F1	◌॑ ◌॒ ◌॓ ◌॔ ◌ॕ ◌ॖ ◌ॗ ◌᳐ ◌᳑ ◌᳒ ◌᳓ ◌᳔ ◌᳕ ◌᳖ ◌᳗ ◌᳘ ◌᳙ ◌᳚ ◌᳛ ◌᳜ ◌᳝ ◌᳞ ◌᳟ ◌᳠ ◌᳡ ◌᳢ ◌᳣ ◌᳤ ◌᳥ ◌᳦ ◌᳧ ◌᳨ ◌᳭ ◌ᳲ ◌ᳳ ◌ᳵ ◌ᳶ ◌ᷧ ◌ᷨ ◌ᷩ ◌ᷪ ◌ᷫ ◌ᷬ ◌ᷭ ◌ᷮ ◌ᷯ ◌ᷰ ◌ᷱ ◌ᷲ ◌ᷳ ◌ᷴ ◌꣠ ◌꣡ ◌꣢ ◌꣣ ◌꣤ ◌꣥ ◌꣦ ◌꣧ ◌꣨ ◌꣩ ◌꣪ ◌꣫ ◌꣬ ◌꣭ ◌꣮ ◌꣯ ◌꣰ ◌꣱ ◌ꣿ ◌꦳ ◌ꦴ ◌ꦵ ◌ꦶ ◌ꦷ ◌ꦸ ◌ꦹ ◌ꦺ ◌ꦻ ◌ꦼ ◌ꦽ ◌ꦾ ◌ꦿ ◌ꧥ ◌ꨩ ◌ꨪ ◌ꨫ ◌ꨬ ◌ꨭ ◌ꨮ ◌ꨯ ◌ꨰ ◌ꨱ ◌ꨲ ◌ꨳ ◌ꨴ ◌ꩃ ◌ꩌ ◌ꩍ ◌ꩻ ◌ꩼ ◌ꩽ ◌ꪰ ◌ꪲ ◌ꪳ ◌ꪴ ◌ꪵ ◌ꪶ ◌ꪷ ◌ꪸ ◌ꪹ ◌ꪺ ◌ꪻ ◌ꪼ ◌ꪽ ◌ꫝ ◌꫞ ◌꫟ ◌ꫫ ◌ꫬ ◌ꫭ ◌ꫮ ◌ꫯ ◌꫰ ◌꫱ ◌ꯣ ◌ꯤ ◌ꯥ ◌ꯦ ◌ꯧ ◌ꯨ ◌ꯩ ◌ﬞ ◌𐇽 ◌𐋠 ◌𐍶 ◌𐍷 ◌𐍸 ◌𐨝 ◌𐨸 ◌𐨹 ◌𐨺 ◌𐨿 ◌𐫥 ◌𐫦 ◌𐴤 ◌𐴥 ◌𐴦 ◌𐴧 ◌𐺫 ◌𑀀 ◌𑀁 ◌𑁆 ◌𑁿 ◌𑂀 ◌𑂁 ◌𑂂 ◌𑂹 ◌𑂻 ◌𑂼 ◌𑂾 ◌𑂿 ◌𑃀 ◌𑃁 ◌𑄀 ◌𑄁 ◌𑄂 ◌𑄧 ◌𑄬 ◌𑄭 ◌𑄮 ◌𑄯 ◌𑅅 ◌𑅆 ◌𑅳 ◌𑆀 ◌𑆁 ◌𑆂 ◌𑆳 ◌𑆴 ◌𑆵 ◌𑆶 ◌𑆷 ◌𑆸 ◌𑆹 ◌𑆺 ◌𑇀 ◌𑇁 ◌𑇊 ◌𑇋 ◌𑇌 ◌𑈬 ◌𑈭 ◌𑈮 ◌𑈯 ◌𑈰 ◌𑈱 ◌𑈲 ◌𑈳 ◌𑈴 ◌𑈵 ◌𑈶 ◌𑈷 ◌𑈸 ◌𑈹 ◌𑈺 ◌𑈻 ◌𑈼 ◌𑈽 ◌𑋟 ◌𑋠 ◌𑋡 ◌𑋢 ◌𑌀 ◌𑌁 ◌𑌻 ◌𑌼 ◌𑍢 ◌𑍣 ◌𑐵 ◌𑐶 ◌𑐷 ◌𑐸 ◌𑐹 ◌𑐺 ◌𑑂 ◌𑑃 ◌𑑄 ◌𑑅 ◌𑑆 ◌𑑞 ◌𑒰 ◌𑒱 ◌𑒲 ◌𑒳 ◌𑒴 ◌𑒵 ◌𑒶 ◌𑒷 ◌𑒸 ◌𑒹 ◌𑒺 ◌𑒻 ◌𑒼 ◌𑒽 ◌𑒾 ◌𑒿 ◌𑓀 ◌𑓁 ◌𑓂 ◌𑓃 ◌𑓄 ◌𑓅 ◌𑖯 ◌𑖰 ◌𑖱 ◌𑖲 ◌𑖳 ◌𑖴 ◌𑖵 ◌𑖶 ◌𑖷 ◌𑖸 ◌𑖹 ◌𑖺 ◌𑖻 ◌𑖼 ◌𑖽 ◌𑖾 ◌𑖿 ◌𑗀 ◌𑗁 ◌𑗂 ◌𑗃 ◌𑗄 ◌𑗅 ◌𑗆 ◌𑗇 ◌𑗈 ◌𑚸 ◌𑜝 ◌𑜞 ◌𑜟 ◌𑠬 ◌𑠭 ◌𑠮 ◌𑠯 ◌𑠰 ◌𑠱 ◌𑠲 ◌𑠳 ◌𑠴 ◌𑠵 ◌𑠶 ◌𑠷 ◌𑠸 ◌𑠹 ◌𑠺 ◌𑧀 ◌𑧁 ◌𑧂 ◌𑧃 ◌𑧄 ◌𑧅 ◌𑧆 ◌𑧇 ◌𑧈 ◌𑧉 ◌𑧊 ◌𑧋 ◌𑧌 ◌𑧍 ◌𑧎 ◌𑧏 ◌𑧑 ◌𑧒 ◌𑧓 ◌𑧔 ◌𑧕 ◌𑧖 ◌𑧗 ◌𑧘 ◌𑧙 ◌𑧚 ◌𑧛 ◌𑧜 ◌𑧝 ◌𑧞 ◌𑧟 ◌𑧠 ◌𑨁 ◌𑨂 ◌𑨃 ◌𑨄 ◌𑨅 ◌𑨆 ◌𑨸 ◌𑨹 ◌𑨺 ◌𑩇 ◌𑩑 ◌𑩒 ◌𑩓 ◌𑩔 ◌𑩕 ◌𑩖 ◌𑩗 ◌𑩘 ◌𑪙 ◌𑪚 ◌𑪛 ◌𑪜 ◌𑪰 ◌𑪱 ◌𑪲 ◌𑪳 ◌𑪴 ◌𑪵 ◌𑪶 ◌𑫄 ◌𑫅 ◌𑫦 ◌𑫧 ◌𑫨 ◌𑫩 ◌𑫪 ◌𑫫 ◌𑫬 ◌𑫭 ◌𑫮 ◌𑫯 ◌𑫰 ◌𑫱 ◌𑫲 ◌𑫳 ◌𑫴 ◌𑫵 ◌𑫶 ◌𑫷 ◌𑫸 ◌𑰯 ◌𑰰 ◌𑰱 ◌𑰲 ◌𑰳 ◌𑰴 ◌𑰵 ◌𑰶 ◌𑰷 ◌𑲩 ◌𑲪 ◌𑲫 ◌𑲬 ◌𑲭 ◌𑲮 ◌𑲯 ◌𑲰 ◌𑲱 ◌𑲲 ◌𑲳 ◌𑲴 ◌𑲵 ◌𑲶 ◌𑲷 ◌𑲸 ◌𑲹 ◌𑲺 ◌𑲻 ◌𑲼 ◌𑲽 ◌𑲾 ◌𑲿 ◌𑴱 ◌𑴲 ◌𑴳 ◌𑴴 ◌𑴵 ◌𑴶 ◌𑴷 ◌𑶊 ◌𑶋 ◌𑶌 ◌𑶍 ◌𑶎 ◌𑶏 ◌𑶐 ◌𑻳 ◌𑻴 ◌𑼃 ◌𑼄 ◌𑼅 ◌𑼆 ◌𑼇 ◌𑼈 ◌𑼉 ◌𑼊 ◌𑼋 ◌𑼌 ◌𑼍 ◌𑼎 ◌𑼏 ◌𑼐 ◌𑼑 ◌𑼒 ◌𑼓 ◌𑼔 ◌𑼕 ◌𑼖 ◌𑼗 ◌𑼘 ◌𑼙 ◌𑼚 ◌𑼛 ◌𑼜 ◌𑼝 ◌𑼞 ◌𑼟 ◌𑼠 ◌𑼡 ◌𑼢 ◌𑼣 ◌𑼤 ◌𑼥 ◌𑼦 ◌𑼧 ◌𑼨 ◌𑼩 ◌𑼪 ◌𑼫 ◌𑼬 ◌𑼭 ◌𑼮 ◌𑽁 ◌𑽂 ◌𑽃 ◌𑽄 ◌𑽅 ◌𑽆 ◌𑾸 ◌𑾹 ◌𑾺 ◌𑾻 ◌𑾼 ◌𑾽 ◌𑾾 ◌𑿀 ◌𑿁 ◌𑿂 ◌𑿃 ◌𑿄 ◌𑿅 ◌𖄰 ◌𖄱 ◌𖄲 ◌𖄳 ◌𖄴 ◌𖄵 ◌𖄶 ◌𖄷 ◌𖄸 ◌𖄹 ◌𖫰 ◌𖫱 ◌𖫲 ◌𖫳 ◌𖫴 ◌𖬰 ◌𖬱 ◌𖬲 ◌𖬳 ◌𖬴 ◌𖬵 ◌𖬶 ◌𖿰 ◌𖿱 ◌𛲝 ◌𛲞 ◌𝅥 ◌𝅦 ◌𝅧 ◌𝅨 ◌𝅩 ◌𝆪 ◌𝆫 ◌𝆬 ◌𝆭 ◌𝉂 ◌𝉃 ◌𞀀 ◌𞀁 ◌𞀂 ◌𞀃 ◌𞀄 ◌𞀅 ◌𞀆 ◌𞀈 ◌𞀉 ◌𞀊 ◌𞀋 ◌𞀌 ◌𞀍 ◌𞀎 ◌𞀏 ◌𞀐 ◌𞀑 ◌𞀒 ◌𞀓 ◌𞀔 ◌𞀕 ◌𞀖 ◌𞀗 ◌𞀘 ◌𞀛 ◌𞀜 ◌𞀝 ◌𞀞 ◌𞀟 ◌𞀠 ◌𞀡 ◌𞀢 ◌𞀣 ◌𞀤 ◌𞄰 ◌𞄱 ◌𞄲 ◌𞄳 ◌𞄴 ◌𞄵 ◌𞄶 ◌𞋬 ◌𞋭 ◌𞋮 ◌𞋯 ◌𞓫 ◌𞥄 ◌𞥅 ◌𞥆 ◌𞥇 ◌𞥈 ◌𞥉 ◌𞥊	Mn	230	Cantillation_Mark	Top	VOWEL_MOD_ABOVE
0952, 1CD5..1CD9, 1CDC..1CDF	◌॒ ◌॓ ◌॔ ◌ॖ ◌ॗ ◌᳞ ◌᳟ ◌᳠ ◌᳡ ◌᳢ ◌᳣ ◌᳤ ◌᳥ ◌᳦ ◌᳧ ◌᳨ ◌᳭ ◌ᳲ ◌ᳳ ◌ᳵ ◌ᳶ ◌ᷧ ◌ᷨ ◌ᷩ ◌ᷪ ◌ᷫ ◌ᷬ ◌ᷭ ◌ᷮ ◌ᷯ ◌ᷰ ◌ᷱ ◌ᷲ ◌ᷳ ◌ᷴ ◌꣠ ◌꣡ ◌꣢ ◌꣣ ◌꣤ ◌꣥ ◌꣦ ◌꣧ ◌꣨ ◌꣩ ◌꣪ ◌꣫ ◌꣬ ◌꣭ ◌꣮ ◌꣯ ◌꣰ ◌꣱ ◌ꣿ ◌꦳ ◌ꦴ ◌ꦵ ◌ꦶ ◌ꦷ ◌ꦸ ◌ꦹ ◌ꦺ ◌ꦻ ◌ꦼ ◌ꦽ ◌ꦾ ◌ꦿ ◌ꧥ ◌ꨩ ◌ꨪ ◌ꨫ ◌ꨬ ◌ꨭ ◌ꨮ ◌ꨯ ◌ꨰ ◌ꨱ ◌ꨲ ◌ꨳ ◌ꨴ ◌ꩃ ◌ꩌ ◌ꩍ ◌ꩻ ◌ꩼ ◌ꩽ ◌ꪰ ◌ꪲ ◌ꪳ ◌ꪴ ◌ꪵ ◌ꪶ ◌ꪷ ◌ꪸ ◌ꪹ ◌ꪺ ◌ꪻ ◌ꪼ ◌ꪽ ◌ꫝ ◌꫞ ◌꫟ ◌ꫫ ◌ꫬ ◌ꫭ ◌ꫮ ◌ꫯ ◌꫰ ◌꫱ ◌ꯣ ◌ꯤ ◌ꯥ ◌ꯦ ◌ꯧ ◌ꯨ ◌ꯩ ◌ﬞ ◌𐇽 ◌𐋠 ◌𐍶 ◌𐍷 ◌𐍸 ◌𐨝 ◌𐨸 ◌𐨹 ◌𐨺 ◌𐨿 ◌𐫥 ◌𐫦 ◌𐴤 ◌𐴥 ◌𐴦 ◌𐴧 ◌𐺫 ◌𑀀 ◌𑀁 ◌𑁆 ◌𑁿 ◌𑂀 ◌𑂁 ◌𑂂 ◌𑂹 ◌𑂻 ◌𑂼 ◌𑂾 ◌𑂿 ◌𑃀 ◌𑃁 ◌𑄀 ◌𑄁 ◌𑄂 ◌𑄧 ◌𑄬 ◌𑄭 ◌𑄮 ◌𑄯 ◌𑅅 ◌𑅆 ◌𑅳 ◌𑆀 ◌𑆁 ◌𑆂 ◌𑆳 ◌𑆴 ◌𑆵 ◌𑆶 ◌𑆷 ◌𑆸 ◌𑆹 ◌𑆺 ◌𑇀 ◌𑇁 ◌𑇊 ◌𑇋 ◌𑇌 ◌𑈬 ◌𑈭 ◌𑈮 ◌𑈯 ◌𑈰 ◌𑈱 ◌𑈲 ◌𑈳 ◌𑈴 ◌𑈵 ◌𑈶 ◌𑈷 ◌𑈸 ◌𑈹 ◌𑈺 ◌𑈻 ◌𑈼 ◌𑈽 ◌𑋟 ◌𑋠 ◌𑋡 ◌𑋢 ◌𑌀 ◌𑌁 ◌𑌻 ◌𑌼 ◌𑍢 ◌𑍣 ◌𑐵 ◌𑐶 ◌𑐷 ◌𑐸 ◌𑐹 ◌𑐺 ◌𑑂 ◌𑑃 ◌𑑄 ◌𑑅 ◌𑑆 ◌𑑞 ◌𑒰 ◌𑒱 ◌𑒲 ◌𑒳 ◌𑒴 ◌𑒵 ◌𑒶 ◌𑒷 ◌𑒸 ◌𑒹 ◌𑒺 ◌𑒻 ◌𑒼 ◌𑒽 ◌𑒾 ◌𑒿 ◌𑓀 ◌𑓁 ◌𑓂 ◌𑓃 ◌𑓄 ◌𑓅 ◌𑖯 ◌𑖰 ◌𑖱 ◌𑖲 ◌𑖳 ◌𑖴 ◌𑖵 ◌𑖶 ◌𑖷 ◌𑖸 ◌𑖹 ◌𑖺 ◌𑖻 ◌𑖼 ◌𑖽 ◌𑖾 ◌𑖿 ◌𑗀 ◌𑗁 ◌𑗂 ◌𑗃 ◌𑗄 ◌𑗅 ◌𑗆 ◌𑗇 ◌𑗈 ◌𑚸 ◌𑜝 ◌𑜞 ◌𑜟 ◌𑠬 ◌𑠭 ◌𑠮 ◌𑠯 ◌𑠰 ◌𑠱 ◌𑠲 ◌𑠳 ◌𑠴 ◌𑠵 ◌𑠶 ◌𑠷 ◌𑠸 ◌𑠹 ◌𑠺 ◌𑧀 ◌𑧁 ◌𑧂 ◌𑧃 ◌𑧄 ◌𑧅 ◌𑧆 ◌𑧇 ◌𑧈 ◌𑧉 ◌𑧊 ◌𑧋 ◌𑧌 ◌𑧍 ◌𑧎 ◌𑧏 ◌𑧑 ◌𑧒 ◌𑧓 ◌𑧔 ◌𑧕 ◌𑧖 ◌𑧗 ◌𑧘 ◌𑧙 ◌𑧚 ◌𑧛 ◌𑧜 ◌𑧝 ◌𑧞 ◌𑧟 ◌𑧠 ◌𑨁 ◌𑨂 ◌𑨃 ◌𑨄 ◌𑨅 ◌𑨆 ◌𑨸 ◌𑨹 ◌𑨺 ◌𑩇 ◌𑩑 ◌𑩒 ◌𑩓 ◌𑩔 ◌𑩕 ◌𑩖 ◌𑩗 ◌𑩘 ◌𑪙 ◌𑪚 ◌𑪛 ◌𑪜 ◌𑪰 ◌𑪱 ◌𑪲 ◌𑪳 ◌𑪴 ◌𑪵 ◌𑪶 ◌𑫄 ◌𑫅 ◌𑫦 ◌𑫧 ◌𑫨 ◌𑫩 ◌𑫪 ◌𑫫 ◌𑫬 ◌𑫭 ◌𑫮 ◌𑫯 ◌𑫰 ◌𑫱 ◌𑫲 ◌𑫳 ◌𑫴 ◌𑫵 ◌𑫶 ◌𑫷 ◌𑫸 ◌𑰯 ◌𑰰 ◌𑰱 ◌𑰲 ◌𑰳 ◌𑰴 ◌𑰵 ◌𑰶 ◌𑰷 ◌𑲩 ◌𑲪 ◌𑲫 ◌𑲬 ◌𑲭 ◌𑲮 ◌𑲯 ◌𑲰 ◌𑲱 ◌𑲲 ◌𑲳 ◌𑲴 ◌𑲵 ◌𑲶 ◌𑲷 ◌𑲸 ◌𑲹 ◌𑲺 ◌𑲻 ◌𑲼 ◌𑲽 ◌𑲾 ◌𑲿 ◌𑴱 ◌𑴲 ◌𑴳 ◌𑴴 ◌𑴵 ◌𑴶 ◌𑴷 ◌𑶊 ◌𑶋 ◌𑶌 ◌𑶍 ◌𑶎 ◌𑶏 ◌𑶐 ◌𑻳 ◌𑻴 ◌𑼃 ◌𑼄 ◌𑼅 ◌𑼆 ◌𑼇 ◌𑼈 ◌𑼉 ◌𑼊 ◌𑼋 ◌𑼌 ◌𑼍 ◌𑼎 ◌𑼏 ◌𑼐 ◌𑼑 ◌𑼒 ◌𑼓 ◌𑼔 ◌𑼕 ◌𑼖 ◌𑼗 ◌𑼘 ◌𑼙 ◌𑼚 ◌𑼛 ◌𑼜 ◌𑼝 ◌𑼞 ◌𑼟 ◌𑼠 ◌𑼡 ◌𑼢 ◌𑼣 ◌𑼤 ◌𑼥 ◌𑼦 ◌𑼧 ◌𑼨 ◌𑼩 ◌𑼪 ◌𑼫 ◌𑼬 ◌𑼭 ◌𑼮 ◌𑽁 ◌𑽂 ◌𑽃 ◌𑽄 ◌𑽅 ◌𑽆 ◌𑾸 ◌𑾹 ◌𑾺 ◌𑾻 ◌𑾼 ◌𑾽 ◌𑾾 ◌𑿀 ◌𑿁 ◌𑿂 ◌𑿃 ◌𑿄 ◌𑿅 ◌𖄰 ◌𖄱 ◌𖄲 ◌𖄳 ◌𖄴 ◌𖄵 ◌𖄶 ◌𖄷 ◌𖄸 ◌𖄹 ◌𖫰 ◌𖫱 ◌𖫲 ◌𖫳 ◌𖫴 ◌𖬰 ◌𖬱 ◌𖬲 ◌𖬳 ◌𖬴 ◌𖬵 ◌𖬶 ◌𖿰 ◌𖿱 ◌𛲝 ◌𛲞 ◌𝅥 ◌𝅦 ◌𝅧 ◌𝅨 ◌𝅩 ◌𝆪 ◌𝆫 ◌𝆬 ◌𝆭 ◌𝉂 ◌𝉃 ◌𞀀 ◌𞀁 ◌𞀂 ◌𞀃 ◌𞀄 ◌𞀅 ◌𞀆 ◌𞀈 ◌𞀉 ◌𞀊 ◌𞀋 ◌𞀌 ◌𞀍 ◌𞀎 ◌𞀏 ◌𞀐 ◌𞀑 ◌𞀒 ◌𞀓 ◌𞀔 ◌𞀕 ◌𞀖 ◌𞀗 ◌𞀘 ◌𞀛 ◌𞀜 ◌𞀝 ◌𞀞 ◌𞀟 ◌𞀠 ◌𞀡 ◌𞀢 ◌𞀣 ◌𞀤 ◌𞄰 ◌𞄱 ◌𞄲 ◌𞄳 ◌𞄴 ◌𞄵 ◌𞄶 ◌𞋬 ◌𞋭 ◌𞋮 ◌𞋯 ◌𞓫 ◌𞥄 ◌𞥅 ◌𞥆 ◌𞥇 ◌𞥈 ◌𞥉 ◌𞥊	Mn	220	Cantillation_Mark	Bottom	VOWEL_MOD_BELOW
1CD4	◌᳔	Mn	1	Cantillation_Mark	Overstruck	VOWEL_MOD_BELOW
1CE1	◌᳡	Mc	0	Cantillation_Mark	Right	VOWEL_MOD_POST
0903	◌ः	Mc	0	Visarga	Right	VOWEL_MOD_POST
A838	꠸	Sc	0	Other	NA	SYM *or* BASE_IND
A836..A837, A839	꠶ ꠷ ꠹	So	0	Other	NA	SYM *or* BASE_IND
200D		Cf	0	Joiner	NA	ZERO_WIDTH_JOINER
200C		Cf	0	Non_Joiner	NA	ZERO_WIDTH_NON_JOINER
1CF8..1CF9	◌᳸ ◌᳹	Mn	230	Cantillation_Mark	NA	*undefined*

There are some problems in this table:

- The classification of code points 1CED and 1CE2..1CE8 is clearly wrong – combining marks aren't meant to be used by themselves, as BASE_IND would indicate.
- Code points A838 and A836..A837, A839 could by the current classification be either BASE_IND or SYM.
- For code points 1CF8..1CF9 the Unicode data isn't sufficient to fully determine the USE character classes, although it appears that their positional category should be Top and their USE subclass therefore VOWEL_MOD_ABOVE.

We'll investigate below what this means for validation specifications and implementations.

Devanagari clusters in the Unicode Standard

The description of Devanagari in the Unicode Standard doesn't provide a concise definition of an orthographic Devanagari cluster. Instead, there are several separate pieces of information scattered over the chapter:

- A *dead consonant* is a consonant followed by a virama; it can't be followed by a vowel, whether independent or dependent (page 447).
- A *live consonant* is a consonant not followed by a virama (it doesn't necessarily mean that the inherent vowel in a live consonant is actually pronounced).
- Consonant clusters consist of an (often empty) sequence of dead clusters, followed by one live consonant (page 451).
- A live consonant can be followed by a dependent vowel (page 451).
- A dead consonant shouldn't be followed by an independent or dependent vowel (page 451).
- The tables 12.1, 12.2, and 12.3 show glyphs for vowel letters, atomic consonants, and consonant conjuncts respectively that have a simple encoding as well as longer character sequences that could result in the same rendering. The longer character sequences should not be used (pages 450–452 and 454).
- A dead consonant can be followed by U+200C ZERO WIDTH NON-JOINER or U+200D ZERO WIDTH JOINER to control conjunct formation (pages 454–455).
- Independent vowels stand on their own (page 449).
- Dependent vowels do not stand on their own; they're used in combination with a base letterform (page 449).
- The nukta sign can immediately follow a consonant, and precede a virama (page 460).
- Other modifying marks should follow all other characters in the cluster. Bindus follow vowel signs; svaras come last (page 460). Unfortunately, the standard doesn't define what it means by "bindus" and "svaras" in this context. "Bindus" are later (page 466) described as "represented by U+0901 DEVANAGARI SIGN CANDRABINDU and U+0902 DEVANAGARI SIGN ANUSVARA", but may also correspond to the Indic shaping category "Bindu", which has three additional Devanagari characters. "Svara" commonly means "vowel", but the text makes clear that it here refers to some "other type of combining mark" besides vowels.
- Storage of plain text generally follows phonetic order; that is, a CV syllable with a dependent vowel is always encoded as a consonant letter C followed by a vowel sign V (page 461).
- A cluster-initial RA can have U+200D ZERO WIDTH JOINER before or after the virama to control alternate forms (page 462).

From that we can derive a regular expression for a Devanagari cluster, assuming terms relating to Indic syllabic categories mean those categories and that the undefined "svara" means all remaining categories that contain marks, and using the syntax characters of Perl/Python/Java/JavaScript regular expression patterns:

```
Vowel_Independent | (0930 Nukta? Joiner? Virama)? (Consonant Nukta? Virama
```

```
(Joiner | Non_Joiner)?)* Consonant Nukta? (Virama | Vowel_Dependent? [0900-
0902]* (Cantillation_Mark | Visarga | 1CED | [1CE2-1CE8])*)
```

The canonical combining classes don't always align with this, however. In particular, the cantillation marks U+1CD4 and U+1CE2..U+1CE8 have combining class 1, so normalization would reorder them to before a nukta, which has combining class 7.

The regular expression does not capture that an independent vowel shouldn't follow a virama. The Unicode Standard also includes three tables showing glyphs for vowel letters, atomic consonants, and consonant conjuncts that have a simple encoding as well as longer character sequences that could result in the same rendering. The longer character sequences should not be used. Validation has to check for any such character sequences and reject them.

Devanagari clusters in the Devanagari shaping engine

In OpenType rendering systems, Devanagari is usually handled by the Devanagari shaping engine. The documentation for this engine provides three patterns for Devanagari syllables:

```
{C+[N]+<H+[<ZWNJ|ZWJ>]|<ZWNJ|ZWJ>+H>} + C+[N]+[A] + [< H+
[<ZWNJ|ZWJ>] | {M}+[N]+[H]>]+[SM]+[(VD)]

[Ra+H]+V+[N]+[<[<ZWJ|ZWNJ>]+H+C|ZWJ+C>]+[{M}+[N]+[H]]+
[SM]+[(VD)]

#[Ra+H]+NBSP+[N]+[<[<ZWJ|ZWNJ>]+H+C>]+[{M}+[N]+[H]]+[SM]+
[(VD)]
```

The syntax of these patterns is documented, except for the "#" character, but of the nonterminals only some clearly map to Unicode character categories, and especially SM, syllable modifier signs, and VD, vedic, seem to be buckets with a large variety of characters. My best guess for the meaning of the symbols that don't directly map to Indic syllabic categories:

- M, matra: Vowel_Dependent
- SM, syllable modifier signs: [0900-0902] | 0951 | 20F0 | [A8E0-A8F1] | Visarga
- VD, vedic: [1CD0-1CD2] | [1CDA-1CDB] | 1CE0 | 1CF4 | [1CD5-1CD9] | [1CDC-1CDF] | 1CD4 | 1CE1 | [1CF8-1CF9] | 1CED | [1CE2-1CE8], or in shorter form [1CD0-1CD2] | [1CD4-1CE8] | 1CED | 1CF4 | [1CF8-1CF9]

With that, and omitting the unexplained "#", the patterns become:

```
(Consonant Nukta? (Virama (Non_Joiner | Joiner)? | (Non_Joiner | Joiner)
Virama))* Consonant Nukta? 0952? (Virama (Non_Joiner | Joiner)? |
Vowel_Dependent* Nukta? Virama?)? SM? VD{0,2}

(0930 Virama)? Vowel_Independent Nukta? ((Joiner | Non_Joiner)? Virama
Consonant | Joiner Consonant)? (Vowel_Dependent* Nukta? Virama?)? SM?
VD{0,2}

(0930 Virama)? 00A0 Nukta? ((Joiner | Non_Joiner)? Virama Consonant)?
```

```
(Vowel_Dependent* Nukta? Virama?)? SM? VD{0,2}
```

Devanagari clusters in the Universal shaping engine

The Universal shaping engine (USE) in OpenType is a generic shaping engine designed primarily for Brahmic scripts. It is not the default engine for Devanagari, but the OpenType implementations in HarfBuzz and CoreText let a font opt into shaping by the USE by using the "dev3" script tag.

The USE's cluster validation is quite clearly defined, as long as all characters involved have their USE classes fully defined – which, as we've seen above, unfortunately isn't the case for Devanagari. My best guesses for the classes of characters where the USE class is currently undefined, wrong, or ambiguous:

- 1CF8..1CF9: VOWEL_MOD_ABOVE
- 1CED: VOWEL_MOD_BELOW
- 1CE2..1CE8: VOWEL_MOD_BELOW (CONS_FINAL_MOD might be possible as well)
- A838, A836..A837, A839: SYM

Omitting classes and cluster patterns that aren't relevant to Devanagari, the patterns defined in the USE documentation become:

```
BASE_IND | OTHER

CONS_WITH_STACKER? (BASE | BASE_OTHER) CONS_MOD_BELOW*
(HALANT BASE CONS_MOD_BELOW*)* VOWEL_PRE* VOWEL_ABOVE*
VOWEL_BELOW* VOWEL_POST* VOWEL_MOD_ABOVE*
VOWEL_MOD_BELOW* VOWEL_MOD_POST*

CONS_WITH_STACKER? (BASE | BASE_OTHER) CONS_MOD_BELOW*
(HALANT BASE CONS_MOD_BELOW*)* HALANT

SYM
```

Test environments

The following sections look at the differences between these definitions and test how different implementations handle them. The comparison tables have the following columns, some of them only in Firefox or in Safari:

- *Text* or *Noto*: Shows a Devanagari character sequence that may or may not be a valid cluster, rendered using an OpenType Devanagari shaping engine and the Noto Sans Devanagari font.
- *Dev3* (only in Firefox and Safari): Shows the same Devanagari character sequence, rendered using an OpenType Universal shaping engine and a modified version of an older version of the Noto Sans Devanagari font (the last one under the Apache license), in which "dev2" script tags have been replaced with "dev3" script tags. In environments

supporting “dev3” script tags, this causes the font to be handled by the USE. Note that the USE differs from the Devanagari shaping engine not only in validation (which we want to test here), but also in subsequent reordering and feature processing steps. The font is not designed for compatibility with these differences, so some character sequences may be rendered in ways not intended by its designers.

- *Annapurna* (only in Firefox): Shows the same Devanagari character sequence, rendered using the Graphite shaping system and the Annapurna font. As a Graphite font, this font has to implement cluster validation itself.
- *Sangam* and *MT* (only in Safari): Shows the same Devanagari character sequence, rendered using the AAT shaping system and the Devanagari Sangam MN and Devanagari MT fonts. As AAT fonts, these fonts should implement cluster validation themselves. As it turned out, Devanagari MT doesn’t do any validation and doesn’t even have a dotted circle glyph.
- *Code*: Shows the Unicode code points of the Devanagari character sequence, as the tests often involve reordering the sequence of non-spacing marks or invisible characters.
- *Unicode*, *OpenType dev2*, and *OpenType dev3*: Whether the cluster definitions discussed above would consider the character sequence a valid cluster (✓) or invalid (◌).
- *Edge*, *Firefox dev2*, *Safari dev2*, *Firefox dev3*, and *Safari dev3*: Whether these browsers and the underlying OpenType shaping systems (DirectWrite for Edge, HarfBuzz for Firefox, and CoreText for Safari) consider the character sequence a valid cluster or not, as evidenced by the rendering using Noto Sans Devanagari (dev2) or the modified Dev3 font (dev3) in these browsers. For Edge, the “legacy” version was tested, as the new Chromium-based version uses HarfBuzz rather than DirectWrite for shaping.
- *Annapurna*, *Sangam*, and *MT*: Whether the Annapurna, Devanagari Sangam MN, and Devanagari MT fonts consider the character sequence a valid cluster or not, as evidenced by the rendering in Firefox or Safari.

When a font doesn’t support all the characters in the tested character sequence, the corresponding cells are shown in gray.

Cluster bases

The Devanagari section of the Unicode Standard describes only consonants as cluster bases; the Devanagari shaping engine adds independent vowels and NO-BREAK SPACE; the Universal Shaping Engine also allows digits, avagraha characters, and DOTTED CIRCLE.

This table tests with the consonant क, the independent vowel अ, the digit ०, the DEVANAGARI SIGN AVAGRAHA ऽ, the VEDIC SIGN ARDHAVISARGA ऻ, and the dotted circle by attaching the nukta ◌़, the pre-base vowel ◌ि, the anusvara ◌ं, and the visarga ◌ः.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
क़	क़	क़	क़	0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

				093C											
कि	कि	कि	कि	0915 093F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
कं	कं	कं	कं	0915 0902	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
कः	कः	कः	कः	0915 0903	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़	क़	क़	क़	0905 093C	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
कि	कि	कि	कि	0905 093F	○	✓	✓	✓	✓	✓	✓	✓	(✓)	○	✓
अं	अं	अं	अं	0905 0902	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
अः	अः	अः	अः	0905 0903	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
०़	०़	०़	०़	0966 093C	○	○	✓	○	✓	○	✓	✓	✓	○	(✓)
०ि	०ि	०ि	०ि	0966 093F	○	○	✓	○	✓	○	✓	✓	(✓)	○	(✓)
०ं	०ं	०ं	०ं	0966 0902	○	○	✓	○	✓	○	✓	✓	✓	○	(✓)
०ः	०ः	०ः	०ः	0966 0903	○	○	✓	○	✓	○	✓	✓	✓	○	✓
ऽ़	ऽ़	ऽ़	ऽ़	093D 093C	○	○	✓	○	✓	○	✓	✓	✓	○	(✓)
ऽि	ऽि	ऽि	ऽि	093D 093F	○	○	✓	○	○	○	✓	✓	(✓)	○	(✓)
ऽं	ऽं	ऽं	ऽं	093D 0902	○	○	✓	○	✓	○	✓	✓	✓	○	(✓)
ऽः	ऽः	ऽः	ऽः	093D 0903	○	○	✓	○	✓	○	✓	✓	✓	○	✓
ᳲ़	ᳲ़			1CF2 093C	○	○	○	○	✓	○	○	○			
ᳲि	ᳲि			1CF2 093F	○	○	○	○	✓	○	○	○			
ᳲं	ᳲं			1CF2 0902	○	○	○	○	✓	○	○	○			
ᳲः	ᳲः			1CF2 0903	○	○	○	○	✓	○	○	○			

◌़	◌़	◌◌़		25CC 093C	○	○	✓	✓	✓	✓	✓	✓	✓	○	
◌ि	◌ि	◌◌ि		25CC 093F	○	○	✓	✓	✓	✓	✓	✓	(✓)	○	
◌॑	◌॑	◌◌॑		25CC 0902	○	○	✓	✓	✓	✓	✓	✓	✓	○	
◌ः	◌ः	◌◌ः		25CC 0903	○	○	✓	✓	✓	✓	✓	✓	✓	○	
◌़	◌़			00A0 093C	○	✓	✓	✓	✓	✓	✓	✓	✓		
◌ि	◌			00A0 093F	○	✓	✓	✓	✓	○	✓	✓	✓		
◌॑	◌॑			00A0 0902	○	✓	✓	✓	✓	✓	✓	✓	✓		
◌ः	◌ः			00A0 0903	○	✓	✓	✓	✓	✓	✓	✓	✓		

Observations: The set of supported cluster bases varies significantly. Shaping engines support more than the Unicode Standard and the Devanagari shaping engine documentation would suggest, but support for digits and avagraha is limited. It's not clear whether HarfBuzz's support for ardhavisarga as a base is based on evidence that marks attach to it.

Recommendations: Cluster models for Devanagari should include consonants, independent vowels, digits, avagraha, dotted circle, and no-break space as base characters. Which Vedic signs to include as bases needs to be decided based on evidence that marks attach to them – see the section Vedic character combinations for some examples.

Repha

Repha is an above-base mark representing an initial dead consonant RA in a cluster. According to the Unicode Standard, it's encoded as RA plus VIRAMA when followed by a consonant, but as just RA when followed by a vowel (primarily vocalic r and l), which should be encoded in its dependent form. The regular expressions for the Devanagari shaping engine, on the other hand, expect repha-vowel combinations to be encoded as RA plus VIRAMA followed by the independent vowel, and add the use of repha with NO-BREAK SPACE. In this case, we're not only checking whether the character sequence is accepted as valid, but also whether the sequence of RA and VIRAMA is recognized as part of a larger cluster and the *repha* glyph is used. A checkmark in parentheses indicates that this does not happen.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT

र्क	र्क	र्क	र्क	0930 094D 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
र्अ	र्अ	र्अ	र्अ	0930 094D 0905	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऋ	र्ऋ	र्ऋ	र्ऋ	0930 094D 090B	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॠ	र्ॠ	र्ॠ	र्ॠ	0930 094D 0960	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऌ	र्ऌ	र्ऌ	र्ऌ	0930 094D 090C	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॡ	र्ॡ	र्ॡ	र्ॡ	0930 094D 0961	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॲ	र्ॲ	र्ॲ	र्ॲ	0930 094D 0972	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऍ	र्ऍ	र्ऍ	र्ऍ	0930 094D 0945	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऎ	र्ऎ	र्ऎ	र्ऎ	0930 094D 0946	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऑ	र्ऑ	र्ऑ	र्ऑ	0930 094D 0949	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऒ	र्ऒ	र्ऒ	र्ऒ	0930 094D 094A	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ओ	र्ओ	र्ओ	र्ओ	0930 094D 094B	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्औ	र्औ	र्औ	र्औ	0930 094D 094C	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऋ	र्ृ	र्ृ	र्ृ	0930 094D	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॠ	र्ॄ	र्ॄ	र्ॄ	0930 094D 0944	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऌ	र्ॢ	र्ॢ	र्ॢ	0930 094D 0962	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॡ	र्ॣ	र्ॣ	र्ॣ	0930 094D 0963	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ॲ	र्ॅ	र्ॅ	र्ॅ	0930 094D 0945	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऍ	र्ॅ	र्ॅ	र्ॅ	0930 094D 0945	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऎ	र्े	र्े	र्े	0930 094D 0947	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऑ	र्ॉ	र्ॉ	र्ॉ	0930 094D 0949	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऒ	र्ॊ	र्ॊ	र्ॊ	0930 094D 094A	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ओ	र्ो	र्ो	र्ो	0930 094D 094B	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्औ	र्ौ	र्ौ	र्ौ	0930 094D 094C	◌	✓	✓	✓	✓	(✓)	✓	✓	(✓)	(✓)	(✓)
र्ऋ	र्ृ	र्ृ	र्ृ	0930 094D 0943	✓	✓	✓	✓	✓	✓	(✓)	(✓)	(✓)	(✓)	✓
र्	र्			0930 094D 00A0	◌	✓	✓	(✓)	(✓)	(✓)	✓	✓	(✓)		
◌ं	◌ं	र्◌		0930 094D 25CC	◌	◌	✓	✓	✓	✓	✓	✓	(✓)	(✓)	

Observations: Every implementation allows every representation of repha, including the one that the Unicode Standard prohibits. However, it seems independent vowels, no-break space, and dotted circle are sometimes treated as new clusters rather than continuations of the ra-virama cluster, as the repha glyph is not used.

Recommendations: The combination of RA, VIRAMA with an independent vowel should be treated as two separate clusters, so that the *repha* glyph is not used. No-break space and dotted circle should be treated as continuations of the cluster that the ra-virama sequence started, so that the *repha* glyph is used.

Nukta

The Unicode Standard and the USE treat nukta signs as consonant modifiers and require them to immediately follow consonants. The Devanagari shaping engine also allows them after dependent vowels, and both OpenType shaping engines allow them after independent vowels.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
क़ि	क़ि	क़ि	क़ि	0915 093C 093F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
़कि	कि़	़कि	क़	0915	◌	✓	◌	✓	✓	✓	◌	◌	✓	✓	(✓)

				093F 093C											
क़ा	क़ा	क़ा	क़ा	0915 093C 093E	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
का़	का़	का़	का़	0915 093E 093C	○	✓	○	✓	✓	✓	○	○	✓	✓	✓
अ़	अ़	अ़	अ़	0905 093C	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations: Support for nukta after dependent vowels is spotty. Using nukta in this position could be for two reasons: either users want to actually indicate that the dependent vowel has a different pronunciation, or they intend it to apply to the consonant and just don't know that they should input it before the vowel. Only the first reason would justify support.

Recommendation: Investigate the reasons for using nukta after dependent vowels.

Virama

The Unicode Standard and the USE require a virama to follow a consonant, possibly after an intervening nukta, joiner, or non-joiner. The Devanagari shaping engine also allows it after a dependent vowel, and both OpenType shaping engines allow it after an independent vowel.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
क्	क	क्	क्	0915 094D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़्	क़	क़्	क़्	0915 093C 094D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क्क	क्क	क्क	क्क	0915 094D 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क्‌क	क्‌क	क्‌क	क्‌क	0915 200C 094D 0915	○	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
क्‍क	क्‍क	क्‍क	क्‍क	0915 200D	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

				094D 0915											
क्क	क्क	क्क	क्क	0915 094D 200C 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क्क	क्क	क्क	क्क	0915 094D 200D 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़्क	क़्क	क़्क	क़्क	0915 093C 200C 094D 0915	○	✓	✓	✓	✓	✓	○	✓	✓	✓	✓
क़्क	क़्क	क़्क	क़्क	0915 093C 200D 094D 0915	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़्क	क़्क	क़्क	क़्क	0915 093C 094D 200C 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़्क	क़्क	क़्क	क़्क	0915 093C 094D 200D 0915	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
कि्	कि्	कि्	कि्	0915 093F 094D	○	✓	○	✓	✓	✓	✓	✓	✓	✓	✓
का्	का्	का्	का्	0915 093E 094D	○	✓	○	✓	✓	✓	✓	✓	✓	✓	✓
अ्	अ्	अ◌्	अ्	0905 094D	○	✓	✓	✓	✓	✓	✓	✓	✓	○	✓

Observations: Almost all implementations allow virama after both dependent and independent vowels.

Recommendations: Investigate why that’s allowed.

Unusual mark combinations

The mark combinations tested in this section are not common, but they do occur, and a consistent interpretation across standards, shaping engines, and fonts would be helpful. In clusters with multiple vowels, we also check in which order the vowels are expected.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
क्ं	क्ं	क्ं	क्ं	0915 094D 0902	○	✓	○	✓	✓	✓	○	✓	✓	○	✓
क्:	क्:	क्:	क्:	0915 094D 0903	○	✓	○	✓	✓	✓	○	✓	✓	○	✓
क्ं	क्ं	क्ं	क्ं	0915 094D 0951	○	✓	○	✓	✓	✓	○	✓	✓	○	✓
कं:	कं:	कं:	कं:	0915 0902 0903	✓	○	✓	✓	✓	✓	✓	✓	✓	○	✓
किं्	किं्	किं्	किं्	0915 093F 0902 094D	○	○	○	○	○	○	✓	○	✓	○	✓
किा	किा	किा	किा	0915 093F 093E	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
काि	काि	काि	काि	0915 093E 093F	○	✓	○	○	✓	✓	○	○	(✓)	(✓)	(✓)
कि	कि	कि	कि	0915 093F 0941	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	(✓)
कुि	कुि	कुि	कुि	0915 0941 093F	○	✓	○	○	✓	✓	○	○	(✓)	(✓)	(✓)
कुे	कुे	कुे	कुे	0915	○	✓	○	○	✓	✓	○	○	✓	✓	(✓)

				0941 0947										
कु	कु	कु	कु	0915 0947 0941	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	(✓)
कुॅ	कुॅ	कॅु	कॅु	0915 0945 0941	○	✓	✓	✓	✓	✓	✓	✓	✓	○	(✓)
कुॅ	कुॅ	कुॅ	कुॅ	0915 0941 0945	○	✓	○	○	✓	✓	○	○	✓	✓	(✓)

Observations: Multiple vowels are allowed if you get the order right – and that order may differ between Sangam and the shaping engines. For other unusual combinations, it's hard to predict whether they will be accepted or not. The USE implementation in CoreText seems to align with implementations of the Devanagari shaping engine rather than with the documentation of the USE when it comes to virama combinations.

Recommendations: Investigate which of these combinations make sense in real life, support those well in one specific order, and discontinue support for others.

Udatta and Anudatta

The OpenType Devanagari shaping engine places U+0952 DEVANAGARI STRESS SIGN ANUDATTA before dependent vowels, while U+0951 DEVANAGARI STRESS SIGN UDATTA, which might be classified as either a syllable modifier sign or a vedic sign, should follow the vowel and possibly other syllable modifiers or vedic signs. The USE treats both as vowel modifiers, and places them after vowels.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
कं॑ि	कं॑ि	कं॑ि	कं॑ि	0915 0951 093F	○	○	○	○	○	○	○	○	(✓)	○	(✓)
किं	किं	किं	किं	0915 093F 0951	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क॒ि	क॒ि	क॒ि	क॒ि	0915 0952 093F	○	✓	○	○	○	○	○	○	(✓)	○	(✓)
कि॒	कि॒	कि॒	कि॒	0915	✓	○	✓	✓	✓	✓	✓	✓	✓	✓	✓

				093F 0952											
क॑ं	क॑	क॑ं	क॑	0915 0951 0902	○	○	✓	○	○	○	✓	✓	✓	○	✓
कँ॑	कँ॑	कँ॑ं	कँ॑	0915 0902 0951	✓	○	✓	✓	✓	✓	✓	✓	✓	○	✓
क़॑ं	क़॑ं	क़॑ं	क़॑	0915 0952 0902	○	✓	○	○	○	○	○	○	✓	○	✓
क़ँ॑	क़ँ॑	क़ँ॑ं	क़ँ॑	0915 0902 0952	✓	○	✓	✓	✓	✓	✓	✓	✓	○	✓
क॑ः	क॑:	क॑ः	क॑:	0915 0951 0903	✓	○	✓	○	○	○	✓	✓	✓	○	✓
क॑:	क॑:ं	क॑:ं	क॑:	0915 0903 0951	✓	○	○	✓	✓	✓	○	○	✓	○	✓
क़॑ः	क़॑:	क़॑ः	क़॑:	0915 0952 0903	✓	✓	✓	○	○	○	✓	✓	✓	○	✓
क़॑:	क़॑:ु	क़॑:ु	क़॑:	0915 0903 0952	✓	○	○	✓	✓	✓	○	○	✓	○	✓
क॑ि॑ं	क॑ि॑ं	क॑ि॑ं	क॑ि॑	0915 0951 093F 0902	○	○	○	○	○	○○	○	○○	(✓)	○○	(✓)
कि॑ं	कि॑	कि॑ं	कि॑	0915 093F 0951 0902	○	○	✓	○	○	○	✓	✓	✓	○	✓
कि॑	कि॑	कि॑ं	कि॑	0915 093F 0902 0951	✓	○	✓	✓	✓	(✓)	✓	✓	✓	○	✓
क़ि॑ं	क़ि॑ं	क़ि॑ं	क़ि॑	0915 0952 093F	○	✓	○	○	○	○○	○	○○	(✓)	○○	(✓)

				0902											
क़ि॑	क़ि॑	क़ि॑	क़ि॑	0915 093F 0952 0902	○	○	○	○	○	○	○	○	✓	○	✓
क़ि॒	क़ि॒	क़ि॒	क़ि॒	0915 093F 0902 0952	✓	○	✓	✓	✓	(✓)	✓	✓	✓	○	✓
कऺि॑ः	कऺि॑ः	कऺि॑ः	कऺि॑ः	0915 0951 093F 0903	○	○	○	○	○	○ ○	○	○ ○	(✓)	○ ○	(✓)
कि॑ः	कि॑ः	कि॑ः	कि॑ः	0915 093F 0951 0903	✓	○	✓	○	○	○	✓	✓	✓	○	✓
कि॑ः	कि॑ः॑	कि॑ः॑	कि॑ः	0915 093F 0903 0951	✓	○	○	✓	✓	(✓)	○	○	✓	○	✓
क़ि॑ः	क़ि॑ः	क़ि॑ः	क़ि॑ः	0915 0952 093F 0903	○	✓	○	○	○	○ ○	○	○ ○	(✓)	○ ○	(✓)
क़ि॑ः	क़ि॑ः	क़ि॑ः	क़ि॑ः	0915 093F 0952 0903	✓	○	✓	○	○	○	✓	✓	✓	○	✓
क़ि॑ः	क़ि॑ः॒	क़ि॑ः॒	क़ि॑ः	0915 093F 0903 0952	✓	○	○	✓	✓	(✓)	○	○	✓	○	✓
क़॑	क़॑	क़॑	क़॑	0915 0951 0952	✓	○	✓	✓	✓	✓	○	✓	✓	○	✓
क़॑	क़॑	क़॑	क़॑	0915 0952 0951	✓	✓	○	✓	✓	✓	○	○	✓	○	✓
अ॑	अ॑	अ॑	अ॑	0905 0951	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

अ॒	अ॒	अ॒	अ॒	0905 0952	○	○	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations: Nobody follows the Devanagari shaping engine document's suggestion that anudatta should be allowed before vowels. There's not enough consensus where these marks should go otherwise – in particular, implementations of the Devanagari shaping engine require them to come after visarga, while implementations of the USE require them to precede that character. Sangam doesn't seem to like them in any more complex cluster.

Recommendations: Treat both udatta and anudatta the same way. Keep them in the position where the implementations of the Devanagari shaping engine expect them, that is after the visarga and similar marks. This requires a modification to the USE documentation.

Vedic character combinations

This section investigates character sequences that include Vedic characters from the Devanagari and Vedic extension blocks. The samples are taken from documents in the Unicode document registry that provided attestations for the characters: Everson et al. 2007; South Asia subcommittee 2008; Sharma 2011a; Sharma 2009; Srinidhi and Sridatta 2017; Sharma 2011b.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
ये॑	ये॑			092F 0947 1CE0	✓	✓	✓	✓	✓	✓	✓	✓			
ऊ᳑	ऊ᳑			090A A8E1 A8E1	○	○	✓	○	✓	✓	✓	✓	(✓)		
रा᳒	रा᳒			0930 093E A8E2 A8EF	✓	○	✓	✓	✓	✓	✓	✓	(✓)		
यो᳓	यो᳓			092F 094B A8E9	✓	✓	✓	✓	✓	✓	✓	✓	✓		
३ २उ चना	३ २उ चना			091A A8E3 0928 093E A8E2 A8EB	✓	○	✓	✓	✓	✓	✓	✓	(✓)		

२ प	२ प			0968 A8EE	○	○	✓	○	✓	○	✓	✓	✓		
रा वि	रा वि			0930 093E A8F0	✓	✓	✓	✓	✓	✓	✓	✓	✓		
नृभाऽ२ ३ ‒‒‒	नृभाऽ२ ३ ‒‒‒			0928 0943 A8E2 092D 093E 1CD2 093D 1CD2 0968 1CD2	○	○	✓	○ ○	✓	○ ○	✓	✓			
यु	यु			092F 0943 1CD8	✓	✓	✓	✓	✓	✓	✓	✓			
न्वो∫	न्वो∫			0928 094D 0935 094B 1CE1	✓	✓	✓	✓	✓	✓	✓	✓			
यं◌ः◌	यं◌ः◌			092F 0951 0903 1CE2	✓	○	○	○ ○	○	○ ○	○	○			
यं◌ः	यं◌ः			092F 0951 1CE2 0903	✓	○	✓	○	○	○	○	✓			
जँ	जँ	जँ◌ं		091C 0900 0951	✓	○	✓	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	○	
९◌ं	९ं			A8F4 0951	○	○	○	○	✓	○	○	✓	✓		
७◌ं◌ु	७ुं			1CE9 0902 1CD8	○	○	○	○ ○	✓	○ ○	○	○			
७◌ँ	छँ			1CEA 0901	○	○	○	○	✓	○	○	○			

◌ाे	◌ाे	का		0915 094E 094B	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	
ए॑	ए◌॑			090F 1CD8 A8E3	○	○	○	✓	✓	✓	○	○			
ए॑	ए॑			090F A8E3 1CD8	○	✓	✓	✓	✓	✓	○	✓			
तु	तु			0924 0941 1CD8	✓	✓	✓	✓	✓	✓	✓	✓			
तॉ	तॉ			0924 094B 1CF4	✓	✓	✓	✓	✓	✓	✓	✓			
×क	×क			1CF5 0915	✓	○	✓	(✓)	(✓)	(✓)	(✓)	(✓)			
आँ				0906 1CF8	○	✓	✓	✓	✓	✓					

Observations: No shaping engine treated all samples as valid; the HarfBuzz Devanagari shaping engine came closest. In the second-to-last line, the glyph for क should be stacked on top of the one for U+1CF5 VEDIC SIGN JIHVAMULIYA ×; it's not clear whether the rendering failures are due to the shaping engines, or the font, or both.

Recommendations: As all samples above are attested, they should all be supported. Vedic signs that serve as bases for other signs, such as U+A8FA DEVANAGARI SIGN DOUBLE CANDRABINDU VIRAMA ◌ꣳ or U+1CF5 VEDIC SIGN JIHVAMULIYA ×, need to be classified to allow for such use. For marks, the correct order needs to be determined, in a way that cooperates with normalization (see the next section).

Canonical-equivalent mark sequences

The Unicode Standard specifies that certain character sequences are canonical-equivalent to each other, and should therefore display identically. Equivalence is determined through three operations: decomposition, composition, and reordering of marks based on their canonical combining class. Within the Devanagari script, there's no decomposition or composition, but some marks have canonical combining class values that enable reordering: Nukta, virama, cantillation marks, and the unclassified marks U+1CED and U+1CE2 through U+1CE8. In the following table, rows that use the same code points (in different order) are canonical-equivalent and should be displayed identically. In each case, the first row contains the sequence in normalization form C.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
क़	क़			0915 093C 1CD4	✓	✓	✓	✓	○	✓	○	✓			
क़्	क़्			0915 1CD4 094D	○	○	○	○	○	○	✓	○			
क्‌	क्‌			0915 094D 1CD4	○	✓	○	✓	○	✓	✓	✓			
क॑	क॑			0915 1CD4 0951	✓	○	○	✓	✓	✓	○	○			
क॑	क॑			0915 0951 1CD4	✓	✓	✓	✓	✓	✓	○	✓			
क़्	क़	क्	क़्	0915 093C 094D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क़्	क़्	क़्	क़्	0915 094D 093C	○	○	○	✓	✓	✓	✓	○	✓	○	✓
क़॑	क़॑	क़॑	क़॑	0915 093C 0951	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
क्॑	क्॑	क्॑	क्॑	0915 0951 093C	○	○	○	○	✓	○	✓	○	✓	○	✓
क्॑	क॑	क्॑	क्॑	0915 094D 0951	○	✓	○	✓	✓	✓	○	✓	✓	○	✓
क्॑	क्॑	क्॑	क्॑	0915 0951 094D	○	○	○	○	✓	○	○	○	✓	○	✓
क़॑	क़॑	क़॑	क़॑	0915 0952	✓	✓	○	✓	✓	✓	○	○	✓	○	✓

				0951											
क़ु	क़ु	क़ु	क़ु	0915 0951 0952	✓	○	✓	✓	✓	✓	○	✓	✓	○	✓

Observations: The shaping system used in Firefox, HarfBuzz, normalizes any input string, so there's no visible difference between the different rows with the same code points. In a number of cases, however, normalization is incompatible with the expectations of the shaping engines. The combining character classes used in normalization express primarily the position of a mark relative to the base, while shaping engines expect character sequences to reflect a linguistic model. The workaround Unicode sometimes recommends for such cases, adding the character U+034F COMBINING GRAPHEME JOINER to block mark reordering, is not mentioned in the Devanagari section of the Unicode Standard.

Recommendations: The canonical combining class of a character can't be changed, no matter what problems it causes, because of the Unicode stability policy regarding normalization. The USE needs to be updated to be compatible with normalization. Cantillation marks with position "Overstruck" have to be a separate subclass, and must be allowed before nukta, but also in later places such as after visarga.

Confusable mark sequences

This section is similar to the one above in that it tests character sequences where the visual placement doesn't fully determine the encoding order. However, while sequences in the previous section were canonical-equivalent, sequences here are not equivalent and therefore should render differently. The Universal Shaping Engine documents the order in which marks should occur (generally left-top-bottom-right within each USE character class); the Unicode Standard and the Devanagari shaping engine don't.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
कं़	कं़	कं़	कं़	0915 0902 093C	○	○	○	○	○	○	○	○	✓	○	✓
क़ं	क़ं	क़ं	क़ं	0915 093C 0902	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
कुे	कुे	कुे	कुे	0915 0941 0947	○	✓	○	○	✓	✓	○	○	✓	✓	✓
कुे	कुे	कुे	कुे	0915	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

				0947 0941											
कि	कि	कि	कि	0915 093F 093E	○	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
काि	काि	काि	काि	0915 093E 093F	○	✓	○	○	✓	✓	○	○	(✓)	(✓)	(✓)
कं	कं			0915 0902 1CD4	✓	✓	✓	✓	✓	✓	✓	✓			
कः	कः			0915 1CD4 0902	○	○	○	○	○	○	○	○			
कु॑	कु॑	कु॑	कु॑	0915 0902 0952	✓	○	✓	✓	✓	✓	✓	✓	✓	○	✓
कु॑ः	कु॑ः	कु॑ः	कु॑ः	0915 0952 0902	○	✓	○	○	○	○	○	○	✓	○	✓

Observations: The good news is that for each tested mark pair there clearly is a preferred order. The not-so-good news is that the not-preferred order for vowels is still allowed by the Devanagari shaping engines in HarfBuzz and CoreText. Sangam doesn't seem to like any combination of anusvara and anudatta.

Recommendations: Cluster patterns should allow only one preferred order of non-spacing marks to avoid confusable sequences.

Repeated marks

In a few combinations of Vedic signs, repeated marks in Devanagari are meaningful. Most of the time, they're spelling mistakes. In either case, they should be visible.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
कु॒ु	कु	कु	कु॒	0915 0941 0941	○	✓	✓	○	✓	○	✓	✓	○	✓	✓
कु॒ु॒	कु	कु॒	कु॒	0915	○	✓	✓	○ ○	✓	○ ○	✓	✓	○	○	✓

				0941 0941 0941											
कं॑	कं	कं॑	कं॑	0915 0902 0902	✓	○	✓	○	✓	○	✓	✓	○	○	✓
कं॑॑	कं	कं॑॑	कं॑॑	0915 0902 0902 0902	✓	○	✓	○ ○	○	○ ○	✓	✓	○	○ ○	✓
कः:	कः:	कः॑ः	कः:	0915 0903 0903	✓	○	✓	✓	✓	✓	✓	✓	✓	○	✓
कः::	कः::	कः॑ः॑ः	कः::	0915 0903 0903 0903	✓	○	✓	✓	○	✓	✓	✓	✓	○ ○	✓
क़॒	क़	क़॒	क़॒	0915 093C 093C	○	✓	✓	✓	✓	○	✓	✓	✓	○	✓
क़॒॒	क़	क़॒॒	क़॒॒	0915 093C 093C 093C	○	○	✓	○	○	○ ○	✓	✓	○	○ ○	✓
क़॒ि	क़ि	क़॒ि	क़ि॒	0915 093C 093C 093F	○	○	✓	✓	✓	○ ○	✓	✓	(✓)	○ ○	✓
क्॒	क्	क्॒	क्॒	0915 094D 094D	○	○	○	○	○	○	✓	○	○	○	✓
क्॒॒	क्॒	क्॒॒	क्॒॒	0915 094D 094D 094D	○	○	○	○ ○	○ ○	○ ○	✓	○ ○	○	○ ○	✓
क॒	क॒	क॒॒	क॒॒	0915 0952 0952	✓	○	✓	○	✓	✓	✓	✓	○	○	✓
क॒॒	क॒	क॒॒॒	क॒॒	0915 0952 0952 0952	✓	○	✓	○ ○	✓	○	✓	✓	○	○ ○	✓

क꣡	क꣡			0915 A8E1 A8E1	✓	○	✓	○	✓	✓	✓	✓	✓		
क꣡ं	क꣡			0915 A8E1 A8E1 A8E1	✓	○	✓	○ ○	✓	○	✓	✓	✓		
कु	कु			0915 1CD8 1CD8	✓	✓	✓	○	✓	✓	✓	✓			
कु◌	कु			0915 1CD8 1CD8 1CD8	✓	○	✓	○ ○	✓	○	✓	✓			

Observations: A mark repeated three times will never get you fewer dotted circles than the same mark repeated twice. Beyond that, there doesn't seem to be any recognizable logic behind these results. Note that repeated marks are the only situation where Annapurna inserts dotted circles.

Recommendations: Cluster patterns should allow repetition of marks if it's meaningful. Fonts should be designed to make repeated marks visible, normally by stacking them, but for certain Vedic signs by side-by-side positioning.

"Discouraged" characters

The Devanagari block of the Unicode Standard includes two characters that were intended for use with Latin, not with Devanagari, and whose use is now discouraged entirely: U+0953 DEVANAGARI GRAVE ACCENT and U+0954 DEVANAGARI ACUTE ACCENT. Note that U+0953 DEVANAGARI GRAVE ACCENT is easily confused with U+0947 DEVANAGARI VOWEL SIGN E.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
के	के	के	के	0915 0953	○	○	○	✓	✓	✓	✓	✓	✓	✓	✓
क॑	क॑	क॑	क॑	0915 0954	○	○	○	✓	✓	✓	✓	✓	✓	✓	✓
अे	अे	अे	अे	0905 0953	○	○	○	✓	✓	✓	✓	✓	✓	✓	✓
अ॑	अ॑	अ॑	अ॑	0905	○	○	○	✓	✓	✓	✓	✓	✓	✓	✓

				0954											
०ॅ	०ॆ	०े	०ॅ	0966 0953	○	○	○	○	✓	○	✓	✓	✓	○	✓
०ॉ	०ॊ	०ो	०ॉ	0966 0954	○	○	○	○	✓	○	✓	✓	✓	○	✓
ऽॅ	ऽॆ	ऽे	ऽॅ	093D 0953	○	○	○	○	✓	○	✓	✓	✓	○	✓
ऽॉ	ऽॊ	ऽो	ऽॉ	093D 0954	○	○	○	○	✓	○	✓	✓	✓	○	✓
ᳲॅ	ᳲॆ			1CF2 0953	○	○	○	✓	✓	✓	✓	✓			
ᳲॉ	ᳲॊ			1CF2 0954	○	○	○	✓	✓	✓	✓	✓			

Observations: Apparently no implementer got the message that these characters should not be used with Devanagari – where dotted circles are inserted, they seem to reflect unsupported base characters, not the “discouraged” marks.

Recommendations: The use of these characters in Devanagari clusters should be disallowed; a dotted circle should be inserted before them.

“Do not use” character sequences

The Devanagari section of the Unicode Standard includes three tables showing glyphs for vowel letters, atomic consonants, and consonant conjuncts that have a simple encoding as well as longer character sequences that could result in the same rendering. The longer character sequences should not be used, and validation should therefore reject them. They are tested here.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT
अॅ	अॅ	अॅ	ॲ	0905 0946	○	✓	○	○	○	○	○	○	✓	○	✓
अा	अा	अा	आ	0905 093E	○	✓	○	○	○	○	○	○	✓	○	✓
ऱ्इ	ऱ्इ	ऱ्इ	ऱ्इ	0930 094D 0907	○	✓	○	✓	○	✓	○	✓	✓	✓	✓
उु	उु	उु	उु	0909 0941	○	✓	○	✓	○	○	○	○	✓	○	✓

एॅ	एॅ	ऍ	ऍ	090F 0945	○	✓	○	○	○	○	○	○	✓	✓	✓
एे	एे	एे	ऐ	090F 0946	○	✓	○	○	○	○	○	○	✓	○	✓
एे	एे	एे	ऐ	090F 0947	○	✓	○	○	○	○	○	○	✓	○	✓
अॉ	अॉ	अॉ	ऑ	0905 0949	○	✓	○	○	○	○	○	○	✓	○	✓
आॅ	आॅ	ऑ	आॅ	0906 0945	○	✓	○	○	○	○	○	○	✓	✓	✓
अो	अो	अो	ओ	0905 094A	○	✓	○	○	○	○	○	○	✓	○	✓
आे	आे	आे	ओ	0906 0946	○	✓	○	○	○	○	○	○	✓	○	✓
अो	अो	अो	ओ	0905 094B	○	✓	○	○	○	○	○	○	✓	○	✓
आे	आे	आे	ओ	0906 0947	○	✓	○	○	○	○	○	○	✓	○	✓
अौ	अौ	अौ	औ	0905 094C	○	✓	○	○	○	○	○	○	✓	○	✓
आै	आै	आै	औ	0906 0948	○	✓	○	○	○	○	○	○	✓	○	✓
अॅ	अॅ	ॲ	ॲ	0905 0945	○	✓	○	✓	○	○	○	○	✓	✓	✓
अं	अं	अं		0905 093A	○	✓	○	✓	○	○	○	○	✓	○	
अा	अा	अा		0905 093B	○	✓	○	✓	○	○	○	○	✓	○	
आं	आं	आं		0906 093A	○	✓	○	✓	○	○	○	○	✓	○	
अौ	अौ	अौ		0905 094F	○	✓	○	✓	○	○	○	○	✓	○	
अु	अु	अु		0905 0956	○	✓	○	✓	○	○	○	○	✓	○	
अू	अू	अू		0905 0957	○	✓	○	✓	○	○	○	○	✓	○	
ख्‍ा	ख	ख्‍ा	ख्‍ा	0916 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓

ख॑ा	ख॑ा	ख॑ा	ख॑	0916 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ग़॑ा	ग॑	ग़॑ा	ग़॑	0917 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ग॑ा	ग॑ा	ग॑ा	ग॑	0917 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
घ़॑ा	घ॑	घ़॑ा	घ़॑	0918 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
घ॑ा	घ॑ा	घ॑ा	घ॑	0918 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
च़॑ा	च॑	च़॑ा	च़॑	091A 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
च॑ा	च॑ा	च॑ा	च॑	091A 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ज़॑ा	ज॑	ज़॑ा	ज़॑	091C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ज॑ा	ज॑ा	ज॑ा	ज॑	091C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
झ़॑ा	झ॑	झ़॑ा	झ़॑	091D 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
झ॑ा	झ॑ा	झ॑ा	झ॑	091D 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ञ़॑ा	ञ॑	ञ़॑ा	ञ़॑	091E 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓

ञ्‌ा	ञ	ञ्‌ा	ञ्‍	091E 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ण्‌ा	ण	ण्‌ा	ण्‍	0923 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ण्‌ा	ण	ण्‌ा	ण	0923 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
त्‌ा	त	त्‌ा	त्‍	0924 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
त्‌ा	त	त्‌ा	त	0924 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
थ्‌ा	थ	थ्‌ा	थ्‍	0925 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
थ्‌ा	थ	थ्‌ा	थ	0925 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ध्‌ा	ध	ध्‌ा	ध्‍	0927 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ध्‌ा	ध	ध्‌ा	ध	0927 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
न्‌ा	न	न्‌ा	न्‍	0928 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
न्‌ा	न	न्‌ा	न	0928 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ऩ्‌ा	ऩ	ऩ्‌ा	ऩ्‍	0929 094D	○	○	○	○	○	○	○	○	✓	○	✓

				093E											
ऩा	ऩा	ऩा	ऩा	0929 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ऩा	ऩ	ऩा	ऩा	0928 093C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ऩा	ऩा	ऩा	ऩा	0928 093C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
प़ा	प	प़ा	प़ा	092A 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
प़ा	प़ा	प़ा	प	092A 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ब़ा	ब	ब़ा	ब़ा	092C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ब़ा	ब़ा	ब़ा	ब	092C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
भ़ा	भ	भ़ा	भ़ा	092D 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
भ़ा	भ़ा	भ़ा	भ	092D 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
म़ा	म	म़ा	म़ा	092E 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
म़ा	म़ा	म़ा	म	092E 094D 200D	○	○	○	○	○	○	○	✓	✓	○	✓

				093E										
य्‌ा	य	य्‌ा	या	092F 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
य॒ा	य॒	य॒ा	य॒	092F 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ल्‌ा	ल	ल्‌ा	ल्‌ा	0932 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ल॒ा	ल॒	ल॒ा	ल॒	0932 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
व्‌ा	व	व्‌ा	व्‌ा	0935 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
व॒ा	व॒	व॒ा	व॒	0935 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
श्‌ा	श	श्‌ा	श्‌ा	0936 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
श॒ा	श॒	श॒ा	श॒	0936 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ष्‌ा	ष	ष्‌ा	ष्‌ा	0937 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ष॒ा	ष॒	ष॒ा	ष॒	0937 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
स्‌ा	स	स्‌ा	स्‌ा	0938 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
स॒ा	स॒	स॒ा	स॒	0938 094D	○	○	○	○	○	○	○	✓	✓	○	✓

				200D 093E											
ख़्‌ा	ख़	ख़्‌ा	ख़्‌ा	0959 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ख़ा	ख़ा	ख़ा	ख़ा	0959 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ख़्‌ा	ख़	ख़्‌ा	ख़्‌ा	0916 093C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ख़ा	ख़ा	ख़ा	ख़ा	0916 093C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ग़्‌ा	ग़	ग़्‌ा	ग़्‌ा	095A 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ग़ा	ग़ा	ग़ा	ग़ा	095A 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ग़्‌ा	ग़	ग़्‌ा	ग़्‌ा	0917 093C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ग़ा	ग़ा	ग़ा	ग़ा	0917 093C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ज़्‌ा	ज़	ज़्‌ा	ज़्‌ा	095B 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ज़ा	ज़ा	ज़ा	ज़ा	095B 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓

ज़्া	ज़ा	ज़ा	ज़ा	091C 093C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ज़ा	ज़ा	ज़ा	ज़	091C 093C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
य़्ा	य़	य़ा	य़ा	095F 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
य़ा	य़ा	य़ा	य़ा	095F 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
य़्ा	य़	य़ा	य़ा	092F 093C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
य़ा	य़ा	य़ा	य़ा	092F 093C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ज़्ा	ज़ा	ज़ा		0979 094D 093E	○	○	○	○	○	○	○	○	✓	○	
ज़ा	ज़ा	ज़ा		0979 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	
ष्ा	ष	ष्ा		097A 094D 093E	○	○	○	○	○	○	○	○	✓	○	
ष्ा	ष्ा	ष्ा		097A 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	
ग़्ा	ग़ा	ग़ा	ग़ा	097B 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓

ग्◌ा	ग्◌ा	ग्◌ा	ग्ा	097B 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ज्◌ा	ज्ा	ज्◌ा	ज्ा	097C 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ज्◌ा	ज्◌ा	ज्◌ा	ज्ा	097C 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ड्◌ा	ड्ा	ड्◌ा	ड्ा	097E 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ड्◌ा	ड्◌ा	ड्◌ा	ड्ा	097E 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
ब्◌ा	ब्ा	ब्◌ा	ब्ा	097F 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
ब्◌ा	ब्◌ा	ब्◌ा	ब्ा	097F 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
क्च्◌ा	क्च	क्च्◌ा	क्च्ा	0915 094D 091A 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
क्च◌ा	क्च◌ा	क्च◌ा	क्च	0915 094D 091A 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
क्ष्◌ा	क्ष	क्ष्◌ा	क्ष्ा	0915 094D 0937 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓

क्◌ा	क्◌ा	क्◌ा	क्ा	0915 094D 0937 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
त्त्◌ा	त्त	त्त्◌ा	त्त्ा	0924 094D 0924 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
त्त◌ा	त्त◌ा	त्त◌ा	त्त	0924 094D 0924 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓
न्त्◌ा	न्त	न्त्◌ा	न्त्ा	0928 094D 0924 094D 093E	○	○	○	○	○	○	○	○	✓	○	✓
न्त◌ा	न्त◌ा	न्त◌ा	न्त	0928 094D 0924 094D 200D 093E	○	○	○	○	○	○	○	✓	✓	○	✓

Observations: HarfBuzz reliably rejects “do not use” sequences. DirectWrite, CoreText, and Sangam allow some of them, without much recognizable logic.

Recommendations: These sequences should be reliably rejected.

Marks without bases

Marks without bases are not valid clusters, and OpenType recommends inserting dotted circles to indicate that.

Noto	Dev3	Sangam	MT	Code	Uni-code	Open-Type dev2	Open-Type dev3	Edge	Fire-fox dev2	Safari dev2	Fire-fox dev3	Safari dev3	Anna-purna	Sangam	MT

◌़	◌़	◌़	़	093C	○	○	○	○	○	○	○	○	✓	○	✓
◌ि	◌ि	◌ि	ि	093F	○	○	○	○	○	○	○	○	✓	○	✓
◌ं	◌ं	◌ं	ं	0902	○	○	○	○	○	○	○	○	✓	○	✓
◌:	◌:	◌:	:	0903	○	○	○	○	○	○	○	○	✓	○	✓
-	-			1CE2	○	○	○	✓	✓	✓	✓	✓			

Observations: The shaping engines and the Sangam font insert dotted circles before the marks that have the script property value Devanagari. U+1CE2, which has the script property value Common, doesn't get one – it's possible that it gets redirected to the default shaping engine, which does not validate.

Recommendations: OpenType specifications should be created that clarify how text is broken into script runs and clusters, and provide a validation model that makes sense across all scripts.

References

Annapurna: Peter Martin: Annapurna SIL. Font version 1.204. Part of The Annapurna Font Family. SIL International, 2019.

Edge: Microsoft Edge. Browser version 44.19041.1.0; EdgeHTML 18.19041. Included in Microsoft Windows 10 version 2004. Microsoft, 2020.

Everson et al. 2007: Michael Everson and Peter Scharf (editors), Michel Angot, R. Chandrashekar, Malcolm Hyman, Susan Rosenfield, B. V. Venkatakrishna Sastry, Michael Witzel: Proposal to encode 55 characters for Vedic Sanskrit in the BMP of the UCS. Unicode Consortium, 2007.

Firefox: Firefox. Browser version 79.0. Mozilla, 2020.

MT: Monotype: Devanagari MT. Font version 13.0d1e3. Included in macOS 10.15.6. Apple, 2020.

Noto: Jelle Bosma: Noto Sans Devanagari Regular. Font version 2.001. Google, 2020.

OpenType Devanagari: Developing OpenType Fonts for Devanagari Script. Microsoft, dated 02/08/2018, accessed 2020-07-12.

OpenType USE: Creating and supporting OpenType fonts for the Universal Shaping Engine. Microsoft, dated 07/31/2020, accessed 2020-08-07.

Safari: Safari. Browser version 13.1.2. Included in macOS 10.15.6. Apple, 2020.

Sangam: Muthu Nedumaran: Devanagari Sangam MN. Font version 14.0d1e12. Included in macOS 10.15.6. Apple, 2020.

Sharma 2009: Shriramana Sharma: Request for encoding 1CF4 VEDIC TONE CANDRA ABOVE. Unicode Consortium, 2009.

Sharma 2011a: Shriramana Sharma: Request to annotate 1CD8 VEDIC TONE CANDRA BELOW. Unicode Consortium, 2011.

Sharma 2011b: Shriramana Sharma: Proposal to encode svara markers for the Jaiminiya Archika. Unicode Consortium, 2011.

SIL: Devanagari (Nagari). SIL International. Accessed 2020-08-19.

South Asia subcommittee 2008: South Asia subcommittee: South Asia Subcommittee Report. Unicode Consortium, 2008.

Srinidhi and Sridatta 2017: Srinidhi A and Sridatta A: Request to change the glyphs of Vedic signs Jihvamuliya and Upadhmaniya. Unicode Consortium, 2017.

Unicode: The Unicode Consortium: The Unicode Standard, Version 13.0. The Unicode Consortium, 2020. For Devanagari, in particular section 12.1 Devanagari, pages 447-472.

Unicode Normalization: The Unicode Consortium: Unicode Character Encoding Stability Policies. The Unicode Consortium. Accessed 2020-08-19.