L2/17-166

Re: Proposal for v10.1 of UTS #39
From: Mark Davis
Date: 2017-05-10
Draft: link

It has become clear that we need to enhance some of the data and text in UTS #39, especially in light of recent events with whole-script confusable spoofing of domain name labels.

While it is too late for the upcoming version 10.0 of UTS #39, I suggest that we work towards a 10.1 version of #39 , with a target date of this fall.

> **There is one item that we should probably change for v10.0.** Markus pointed out that using the Σ symbol for “the set of all scripts” can be a bit confusing. Although there is no unique symbol for “universal set” it appears that a usable symbol would be U+1D54C (𝕌) MATHEMATICAL DOUBLE-STRUCK CAPITAL U. Alternatively, we could use ALL.

I propose that we produce a proposed update 10.1 for public review after Unicode 10.0 is released. The rest of this document describes the proposed changes.

Issues:

1. The original confusable characters were derived some time ago (in internet time), and extended from there. For example, It would be useful to gather more specific data as to the fonts used for different scripts in address bars and similar environments to tune the confusable data better to the current environment. That can help to reduce the spurious confusables, such as C and ℭ or “rn” and “m”
2. It might also be useful to have two levels of confusability:
    a. **Same**: where the common glyphs for the characters overlap in shape such as {o vs ο}
    b. **Small**: where the common glyphs for the characters should be disjoint in shape, but the difference is small enough that people may overlook them in context {o vs σ}.
    c. Examples of current data: {Arabic=[ه, ﻫ, ﻬ, ﮪ, ﮫ, ﮬ, ﮭ, ﮦ, ﮧ, ﮨ, ﮩ, ہ, ﻩ, ۿ, ﻪ, ە, ھ, ٥, ۵], Armenian=[օ], Chakma=[𑄪], Common=[ℴ, 𝐨, 𝑜, 𝒐, 𝓸, 𝔬, 𝕠, 𝖔, 𝗈, 𝗼, 𝘰, 𝙤, 𝚘, 𝛐, 𝛔, 𝜊, 𝜎, 𝝄, 𝝈, 𝝾, 𝞂, 𝞸, 𝞼], Coptic=[ⲟ], Cyrillic=[о], Deseret=[𐐬], Devanagari=[०], Georgian=[ჿ], Grantha=[𑌃], Greek=[ο, σ], Gujarati=[૦], Gurmukhi=[੦], Hebrew=[ס], Kaithi=[𑂰], Kannada=[ಂ, ೦], Khojki=[𑈰], Lao=[໐], Latin=[o, ᴏ, ᴑ, ꬽ, ⱺ], Mahajani=[𑅐], Malayalam=[ം, ൦, ഠ], Multani=[𑊰], Myanmar=[၀, ိ], Osage=[𐓪], Sinhala=[ං], Tai_Le=[ᥲ], Tamil=[௦], Telugu=[ం, ౦], Thaana=[ް], Thai=[๐], Warang_Citi=[𑣈, 𑣗]}
3. We should add two new sections to the document, based on drafts presented below.
4. We should also encourage browsers and others to apply more techniques to mitigate problems.
    a. Distinguish between reasonable vs. suspect whole-script confusables using the TLDs (see 5.1.1 suggested text below).
    b. Forbid whole-script confusable labels for high-frequency domain names such as apple.com.
    c. Avoid fonts that increase confusability. Consider the table below. The 1 is pretty distinct, except in Georgia where it can be confused with a dotless i. But the Ӏ U+04C0 CYRILLIC LETTER PALOCHKA looks far too much like a U+006C LATIN SMALL LETTER L in Arial and Roboto, while the U+0433 CYRILLIC SMALL LETTER GHE is too close to an U+0072 LATIN SMALL LETTER R in all but Georgia.

Font	Latin	Cyrillic	Cyrillic
Arial	circle	circ1e	circle
Georgia	circle	circıe	circIe
Roboto	circle	circ1e	circle
Noto Sans	circle	circ1e	circIe

New proposed sections for #39.

5.1.1 Scripts for Whole Script Confusables

It may be useful to be able to determine not just whether a string has a whole-script confusable, but also *which* scripts those whole-script confusables can be in.

This capability can be used, for example, to distinguish between reasonable vs. suspect whole-script confusables. Take the Latin-script domain-name label "circle". It would be fine to see that used in the domain name "circle.com". It would also be fine to see the Cyrillic confusable "circIe" used in the Cyrillic domain name "circIe.рф". But a browser may want to alert, as possible spoofs, "conflicts" like the use of the Cyrillic "circIe" with .com and the Latin "circle" with .рф. To perform such assessments, one needs to be able to determine the scripts of possible whole-script confusables.

> The process of determining such potential conflicts is more complicated than simply looking at the scripts of the labels. It can be perfectly legitimate to have SLD scripts not be the same as TLD scripts, such as:
> - Cyrillic labels in a domain name with a TLD of .ru or .рф
> - Chinese labels in a domain name with a TLD of .com.au or .com
> - Cyrillic labels *that aren't confusable with Latin* with a TLD of .com.au or .com

If we look at the various confusables for the string "circle" (in Latin characters) we find that it has whole-script confusables in 3 other scripts: *Cherokee*, *Coptic*, and *Cyrillic*. The full data is the following, where each row represents a successive character:

1. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Latin=[c, ᴄ]}
2. {Cherokee=[i, □], Common=[⍳], Cyrillic=[і, Ӏ, ι], Latin=[i, ı, ɩ, ɪ]}
3. {Cherokee=[□], Coptic=[ⲅ], Cyrillic=[г], Latin=[r, □, □]}
4. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Latin=[c, ᴄ]}
5. {Common=[1, |, ∣], Coptic=[ⳑ], Cyrillic=[Ӏ, Ӏ], Latin=[l, I, ɭ, ǀ]}
6. {Common=[℮], Cyrillic=[е, ҽ], Latin=[e, □]}

The above results after applying two filters:

1. Mixed-script confusables are eliminated. That is, each must have a character in each row or there must be a Common character.
2. All characters must be NFKC (that is, characters with Identifier_Type=Not_NFKC are not permitted). Allowing non-NFKC characters would result in many more Common characters, and thus many more whole-script confusables. Stronger filters on allowable characters, such as Identifier_Status=Recommended, will result in fewer whole-script confusables.

The following high-level algorithm can be used to determine all scripts that contain a whole-script confusable with a string X:

1. Consider Q, the set of all strings confusable with X.
2. Remove all strings from Q whose resolved script set is ∅ or Σ (that is, keep only single-script strings except for those with characters only in Common).
3. Take the union of the resolved script sets of all strings remaining in Q.

As usual, this algorithm is intended only as a definition; implementations should use an optimized routine that produces the same result.

The set of allowed characters is crucial to the determination of WSCs. Here for example is the set for C.

Allowed Characters	**Results for “C”**
All characters	{Carian=[□], Cherokee=[Ꮯ], Common=[ℂ, ℭ, 𝐂, 𝐶, 𝑪, 𝒞, 𝓒, 𝕮, 𝖢, 𝗖, 𝘊, 𝘾, 𝙲, □], Coptic=[Ⲥ], Cyrillic=[С], Deseret=[□], Elbasan=[□], Greek=[Ϲ], Latin=[C, Ⅽ, Ｃ], Lisu=[□], Old_Italic=[□], Warang_Citi=[□, □]}
NFKC	{Carian=[□], Cherokee=[Ꮯ], Common=[□], Coptic=[Ⲥ], Cyrillic=[С], Deseret=[□], Elbasan=[□], Latin=[C], Lisu=[□], Old_Italic=[□], Warang_Citi=[□, □]}
Identifier_Status=Allowed	{Cyrillic=[С], Latin=[C]}

Casing also makes a difference: uppercase Latin characters tend to have many more confusables. For example, here are sets for ‘c’:

Allowed Characters	**Results for “c”**
All characters	{Cherokee=[□], Common=[𝐜, 𝑐, 𝒄, 𝒸, 𝓬, 𝔠, 𝕔, 𝖈, 𝖼, 𝗰, 𝘤, 𝙘, 𝚌], Coptic=[ⲥ], Cyrillic=[с], Deseret=[□], Greek=[ϲ], Latin=[c, ᴄ, ⅽ, ｃ]}
NFKC	{Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Deseret=[□], Latin=[c, ᴄ]}
Identifier_Status=Allowed	{Cyrillic=[с], Latin=[c]}

8 Sample Implementations

8.1 Scripts for Whole Script Confusables

The algorithm in Section 5.1.1 Scripts for Whole Script Confusables would need to be optimized for use in implementations. Below is a sample optimization in pseudocode.

Define

- **allowedSet** is the set of characters that are allowed by the protocol (such as Identifier_Status=Recommended)
- **toScripts** is a map from each character in **allowedSet** to its augmented script set (see Section 5.1).

- **allScripts** is the union of all scripts resulting from toScripts. Note that because of allowedSet, this may be a subset of Σ.
- **toConfusables** is a map from each character to the set of all characters in **allowedSet** having the same prototype (see Section 4).
- **input** is the input string
- **result** is a list of multimaps from Script to Strings.

Processing

1. **Basic Map:** For each character C in skeleton(**input)**
    a. Create a multimap M from Script to Strings
    b. For each character CC in toConfusables(C)
        i. Let scripts = toScripts(CC)
        ii. If scripts = **allScripts**, let scripts = {Common}
        iii. For each script in scripts
            1. Add {script → CC} to M
    c. *TBD: need slightly more complicated data structure to allow for multiple characters, eg rn → {m…}*
    d. Add M to **result**.
2. **Filter**
    a. Let commonScripts = the union of all scripts that are in elements of **result** that do not contain Common
    b. Add Common to commonScripts.
    c. For each M in result, remove any mappings from scripts that aren't in commonScripts.

Example

For **input** = "circle", the basic map is:

1. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Deseret=[□], Latin=[c, ᴄ]}
2. {Cherokee=[i, □], Common=[⍳], Cyrillic=[і, Ӏ, ӏ], Greek=[ι], Latin=[i, ı, ɩ, ɪ], Warang_Citi=[□]}
3. {Cherokee=[□], Coptic=[ⲅ], Cyrillic=[г], Greek=[ᴦ], Latin=[r, □, □]}
4. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Deseret=[□], Latin=[c, ᴄ]}
5. {Arabic=[١ ,۱ ,ا], Common=[1, |, ∣, ?], Coptic=[ⳑ], Cyrillic=[Ӏ, Ӏ], Greek=[Ι], Hebrew=[ן ,ו ,ו], Latin=[I, l, ɭ, ǀ], Lisu=[□], Lycian=[□], Mende_Kikakui=[□], Miao=[□], Nko=[□], Old_Italic=[□, □], Runic=[ᛁ], Thaana=[١], Tifinagh=[□]}
6. {Common=[℮], Cyrillic=[е, ҽ], Latin=[e, □]}

Rows 1, 3 and 4 don't contain Common. The intersection of the scripts from those rows yields {Cherokee, Coptic, Cyrillic, Latin}. So commonScripts = {Cherokee, Common, Coptic, Cyrillic, Latin}. All other scripts are filtered out, leaving:

1. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Latin=[c, ᴄ]}
2. {Cherokee=[i, □], Common=[⍳], Cyrillic=[і, Ӏ, ӏ], Latin=[i, ı, ɩ, ɪ]}
3. {Cherokee=[□], Coptic=[ⲅ], Cyrillic=[г], Latin=[r, □, □]}
4. {Cherokee=[□], Coptic=[ⲥ], Cyrillic=[с], Latin=[c, ᴄ]}
5. {Common=[1, |, ∣], Coptic=[ⳑ], Cyrillic=[Ӏ, Ӏ], Latin=[I, l, ɭ, ǀ]}
6. {Common=[℮], Cyrillic=[е, ҽ], Latin=[e, □]}

As always, the above is a logical description: algorithms can be optimized as long as they produce the same results.

For many applications, only the commonScripts are necessary. For that, the algorithm can be simplified to only retain the scripts that are keys for each M.

The additional information about the characters can be used to construct the actual whole-script confusables, where that is useful. That process simply takes each script + Common and generates the strings that correspond. For example, for Cherokee it would be:

1. Cherokee=[□]
2. Cherokee+Common=[i, □, ι]
3. Cherokee=[□]
4. Cherokee=[□]
5. Common=[1, |, ǀ]
6. Common=[e]

This would result in 9 strings formed by taking each of the possible paths through these sets of characters: “□i□□1e”, ...

Source for table above

Font	Latin	Cyrillic	Cyrillic
Arial	circle	circ1e	circIe
Georgia	circle	circ1e	circIe
Roboto	circle	ciгc1e	ciгcIe
Noto Sans	circle	ciгc1e	ciгcIe