L2/21-091

Stability policy: Property of characters must not become property of strings

Mathias Bynens, Markus Scherer, Mark Davis
2021-apr-23

Proposal

Add a new Unicode Character Encoding Stability Policy (https://www.unicode.org/policies/stability_policy.html).

Summary: A property of characters must not become a property of strings, or vice versa. Applies to normative and informative properties.

Rough suggested text:

Property Domain Stability

Applicable Version: Unicode 14.0+

The domain of a normative or informative property will never change. In particular, a property of characters will never be changed into a property of strings, and vice versa.

A property that is not explicitly documented as being a property of strings (or having an entirely different domain) is assumed to be a property of characters and must remain that way.

It is theoretically possible (although unlikely) that in some versions of Unicode a property of strings only applies to characters. Whether the property actually applies to multi-character strings (or, unusually, the empty string) might change from version to version while it remains documented as a property of strings.

This stability guarantee does not apply to Contributory properties (such as "Other_Alphabetic") nor to Provisional properties. For a list of which properties are Normative or Informative, see UAX #44, Unicode Character Database.

Rationale

Properties that apply to different domains are processed differently. Implementers have assumed, and must continue to be able to assume, that when using a property of characters they can iterate over the code points of a string and look up all of the relevant values.

Using a property of strings requires substring matches, which are implemented differently.

“Upgrading” the domain of an existing property to a superset domain (e.g., characters to strings) would make all existing implementations miss all of the new domain elements (e.g., multi-character substring elements) and fail to find and process their values.

“Downgrading” the domain of an existing property to a subset domain would not break any existing implementations. They would merely be unnecessarily complex, and implementers may want to optimize their code to take the simpler subset domain into account.

However, even a “downgrade” can cause problems; for example, if regular expressions allow properties of strings only in certain contexts, then a “downgraded” property (now of only characters) would validate with the newer Unicode version, while other implementations that use an older Unicode version’s property metadata would reject what they still see as a property of strings.

We do not see any real use case for allowing “downgrades”.

Background

We already proposed a stability policy against “upgrades” in L2/19-168 “Supporting string properties in regular expressions” (https://www.unicode.org/L2/L2019/19168-regex-string-prop.pdf) and discussed it at UTC #159 (https://www.unicode.org/L2/L2019/19122.htm), but no formal action was recorded for this part of that proposal.

The different domains of properties are especially important for processing of regular expressions. This is why we have formalized this distinction first in UTS #18. See https://www.unicode.org/reports/tr18/#domain_of_properties for the definition of properties of characters (=properties of code points) vs. properties of strings.

See also the related changes in the Proposed Update UTR #23, The Unicode Character Property Model (https://www.unicode.org/review/pri415/ → https://www.unicode.org/reports/tr23/tr23-12.html).

The specification of which properties may apply to strings is important to standards that make use of Unicode properties. For example, for the ECMAScript (JavaScript) standard we propose to support certain properties of strings in character classes, together with set operations (e.g., intersection). In the pending ECMAScript proposal we are including a validation mechanism whose stability (that is, not breaking the validation of existing regular expressions due to changes in future versions of Unicode) relies on not changing the domains of properties. (The proposal forbids a set-complement if it may apply to multi-character strings.) See https://github.com/tc39/proposal-regexp-set-notation/issues/19

If we find that a “domain upgrade” makes sense, we would need to create a new property. This is somewhat similar to the “codomain upgrade” from the Script property to the Script_Extensions property.

If an “upgraded” property is created and the original property of characters is no longer useful, then it could be deprecated. However, the original property may well remain useful, just like the Script property.