Better default values in the UCD

Markus Scherer, Mark Davis, 2021-may-05

Intro

The Unicode Character Database files specify both the default values of properties, and the specific values for specific code points and characters. Some properties have different default values for different blocks, or for certain ranges set aside for various types of future allocations. Default values are specified in special comments at the head of the file containing an @missing directive (see the “Status Quo” below for details).

Proposal

We propose to improve Unicode data files for easier maintenance and better self-documentation.

1. Explicitly specify behavior with multiple @missing lines

Additions to https://unicode.org/reports/tr44/#Missing_Conventions:

> Whenever there is an @missing line whose code point range overlaps with an earlier @missing line, the code points in the overlap are given the value of the later line. For example, consider the following:
>
> ```
> # @missing: 0000..10FFFF; Left_To_Right
> # @missing: 0600..07BF; Arabic_Letter
> ```
>
> Any missing values for code points in the range 0600..07BF are given the default value Arabic_Letter, while missing values in 0000..05FF and 07C0..10FFFF are given the default value Left_To_Right.

2. Allow block values in the place of code point ranges of @missing lines

Additions to https://unicode.org/reports/tr44/#Missing_Conventions:

> A code point range may also be specified by a Block property value to significantly improve readability, as in the following example:
>
> ```
> # @missing: 0000..10FFFF; Left_To_Right
> # @missing: Block=Arabic; Arabic_Letter
> ```

Status quo

Spec: tr44/#Default_Values + tr44/#Missing_Conventions

Default values are provided via pseudo-syntax, in the form of comment lines like this in PropertyValueAliases.txt:

```
# @missing: 0000..10FFFF; General_Category; Unassigned
```

or like this in DerivedBidiClass.txt:

```
# @missing: 0000..10FFFF; Left_To_Right
```

UCD file parsers need to either read these comment lines or hardcode these defaults.

Other than being in comments, and the @missing prefix, the syntax of these lines follows the conventions of the regular data lines.

Parsers override these defaults with per-character and per-range values:
UnicodeData.txt

```
004C;LATIN CAPITAL LETTER L;Lu;0;L;;;;;N;;;;006C;
```

DerivedBidiClass.txt

```
0620..063F    ; AL # Lo  [32] ARABIC LETTER KASHMIRI YEH..
0660..0669    ; AN # Nd  [10] ARABIC-INDIC DIGIT ZERO..ARABIC-INDIC DIGIT NINE
```

Several properties have alternate default values for certain blocks and no-block ranges that are reserved for future characters/scripts/types of symbols including future RTL scripts, unified ideographs, and default ignorable code points.

In UAX #44 and in the data files, these are documented in text and comments, and the data is provided via regular syntax but in this case for unassigned code points.

DerivedBidiClass.txt

```
# Bidi Class (listing UnicodeData.txt, field 4: see UAX #44: http://www.unicode.org/reports/tr44/)
# Unlike other properties, unassigned code points in blocks
# reserved for right-to-left scripts are given either types R or AL.
#
# The unassigned code points that default to AL are in the ranges:
#     [\u0600-\u07BF \u0860-\u086F \u08A0-\u08FF \uFB50-\uFDCF \uFDF0-\uFDFF \uFE70-\uFEFF
#      \U00010D00-\U00010D3F \U00010F30-\U00010F6F
#      \U0001EC70-\U0001ECBF \U0001ED00-\U0001ED4F \U0001EE00-\U0001EEFF]
#
#     This includes code points in the Arabic, Syriac, and Thaana blocks, among others.
...
061D          ; AL # Cn       <reserved-061D>
070E          ; AL # Cn       <reserved-070E>
074B..074C    ; AL # Cn   [2] <reserved-074B>..<reserved-074C>
...
```

Multiple @missing lines for multiple default values

UCD maintainers need to carefully edit the text and comments about alternate default in sync with changing the data file generation code and thus the generated data.

This could be simpler and less fragile if we used multiple @missing lines for the different ranges and their default values.

tr44/#Missing_Conventions documents the syntax but does not say whether there must be only one @missing line per file or property. All such lines in the UCD currently cover the entire code point range.

A natural extension of the default-and-overrides model of UCD files is to allow another level of @missing line overrides, with “last one wins” parsing.

We propose to first provide an @missing line for all of Unicode, and then one line per alternate-default range. Each @missing line or data line sets the value for its range, overriding whatever was set before for any code points in that range.

For example:

```
...
# @missing: 0000..10FFFF; Left_To_Right
...
# The unassigned code points that default to AL are in the following ranges.
# This includes code points in the Arabic, Syriac, and Thaana blocks, among others.
# @missing: 0600..07BF; Arabic_Letter
# @missing: 0860..086F; Arabic_Letter
# @missing: 08A0..08FF; Arabic_Letter
# @missing: FB50..FDCF; Arabic_Letter
...
# The unassigned code points that default to R are in the following ranges.
# This includes code points in the Hebrew, NKo, and Phoenician blocks, among others.
# @missing: 0590..05FF; Right_To_Left
# @missing: 07C0..085F; Right_To_Left
# @missing: 0870..089F; Right_To_Left
...
# The unassigned code points that default to ET are in the range:
# @missing: 20A0..20CF; European_Terminator
...
```

We would remove the data lines for unassigned code points.

(We could in principle also remove all of the data lines for characters with the same value as their range default.)

Parsers that expect and look for @missing lines only for the whole code point range would have to actually parse the range of the @missing line, and treat each @missing line as an override for its entire range, just like they already do for each data line.

Block defaults

Since we often define default values for whole Unicode blocks, we could use those directly in the syntax, making it more self-documenting and easier to understand. When there is no block for a range, we would continue to specify the range itself.

For example:

```
...
# @missing: 0000..10FFFF; Left_To_Right
```

```
...
# The unassigned code points that default to AL are in the following ranges.
# @missing: Block=Arabic; Arabic_Letter
# @missing: Block=Syriac_Supplement; Arabic_Letter
# @missing: Block=Arabic_Extended-A; Arabic_Letter
# @missing: Block=Arabic_Presentation_Forms-A; Arabic_Letter
...
# The unassigned code points that default to R are in the following ranges.
# @missing: Block=Hebrew; Right_To_Left
# @missing: Block=NKo; Right_To_Left
# @missing: Block=Samaritan; Right_To_Left
# @missing: Block=Mandaic; Right_To_Left
# @missing: 0870..089F; Right_To_Left
...
# The unassigned code points that default to ET are in the range:
# @missing: Block=Currency_Symbols; European_Terminator
...
```

Before parsing data files that use this syntax, Blocks.txt would have to be parsed first.

A successful parser of a non-trivial subset of the UCD already has to parse files in this order, in order to be able to handle names and aliases of properties and property values:

1. PropertyAliases.txt
2. PropertyValueAliases.txt
3. everything else

This proposal would insert Blocks.txt into the parse-early set.

Issue: Relying on full block ranges becomes awkward when a Block includes noncharacter code points, which often have a different default value. We would either need to exclude those (easy for parser implementers to overlook), or fall back to an explicit range as above (probably best).

```
...
100000..10FFFD; L # Co [65534] <private-use-100000>..<private-use-10FFFD>
10FFFE..10FFFF; BN # Cn   [2] <noncharacter-10FFFE>..<noncharacter-10FFFF>
```