SWI-Prolog Unicode library

Jan Wielemaker
VU University, Amsterdam
The Netherlands
E-mail: J.Wielemaker@cs.vu.nl

August 17, 2012

Abstract

This package wraps utf8proc unicode routines. This library provides the four unicode normalization forms (NFC, NFD, NFKC, NFKD) as well as access to the Unicode character properties.

Contents

1 library(unicode): Unicode string handling

See also `http://www.public-software-group.org/utf8proc`

This library is a wrapper around the utf8proc library, providing information about Unicode code-points and performing operations (mappings) on Unicode atoms. The central predicate is `unicode_map/3`, mapping a Unicode atom to another Unicode atom using a sequence of operations. The predicates `unicode_nfd/2`, `unicode_nfc/2`, `unicode_nfkd/2` and `unicode_nfkc/2` implement the four standard Unicode normalization forms.

Lump handling:

```
U+0020      <-- all space characters (general category Zs)
U+0027  '   <-- left/right single quotation mark U+2018..2019,
                modifier letter apostrophe U+02BC,
                modifier letter vertical line U+02C8
U+002D  -   <-- all dash characters (general category Pd),
                minus U+2212
U+002F  /   <-- fraction slash U+2044,
                division slash U+2215
U+003A  :   <-- ratio U+2236
U+003C  <   <-- single left-pointing angle quotation mark U+2039,
                left-pointing angle bracket U+2329,
                left angle bracket U+3008
U+003E  >   <-- single right-pointing angle quotation mark U+203A,
                right-pointing angle bracket U+232A,
                right angle bracket U+3009
U+005C  \   <-- set minus U+2216
U+005E  ^   <-- modifier letter up arrowhead U+02C4,
                modifier letter circumflex accent U+02C6,
                caret U+2038,
                up arrowhead U+2303
U+005F  _   <-- all connector characters (general category Pc),
                modifier letter low macron U+02CD
U+0060  `   <-- modifier letter grave accent U+02CB
U+007C  |   <-- divides U+2223
U+007E  ~   <-- tilde operator U+223C
```

unicode_map(+*In*, -*Out*, +*Options*) *[det]*

Perform unicode normalization operations. *Options* is a list of operations. Defined operations are:

stable

Unicode Versioning Stability has to be respected.

compat

Compatiblity decomposition (i.e. formatting information is lost)

compose
Return a result with composed characters.

decompose
Return a result with decomposed characters.

ignore
Strip "default ignorable characters"

rejectna
Return an error, if the input contains unassigned code points.

nlf2ls
Indicating that NLF-sequences (LF, CRLF, CR, NEL) are representing a line break, and should be converted to the unicode character for line separation (LS).

nlf2ps
Indicating that NLF-sequences are representing a paragraph break, and should be converted to the unicode character for paragraph separation (PS).

nlf2lf
Indicating that the meaning of NLF-sequences is unknown.

stripcc
Strips and/or convers control characters. NLF-sequences are transformed into space, except if one of the NLF2LS/PS/LF options is given. HorizontalTab (HT) and FormFeed (FF) are treated as a NLF-sequence in this case. All other control characters are simply removed.

casefold
Performs unicode case folding, to be able to do a case-insensitive string comparison.

charbound
Inserts 0xFF bytes at the beginning of each sequence which is representing a single grapheme cluster (see UAX#29).

lump
(e.g. HYPHEN U+2010 and MINUS U+2212 to ASCII "-"). (See module header for details.) If NLF2LF is set, this includes a transformation of paragraph and line separators to ASCII line-feed (LF).

stripmark
Strips all character markings (non-spacing, spacing and enclosing) (i.e. accents) NOTE: this option works only with `compose` or `decompose`.

unicode_nfd(+*In*, -*Out*) *[det]*
Characters are decomposed by canonical equivalence.

unicode_nfc(+*In*, -*Out*) *[det]*
Characters are decomposed and then recomposed by canonical equivalence. It is possible for the result to be a different sequence of characters than the original.

See also `http://en.wikipedia.org/wiki/Unicode_equivalence#Normal_forms`

unicode_nfkd(+*In*, -*Out*) *[det]*
Characters are decomposed by compatibility equivalence.

unicode_nfkc(*+In, -Out*) *[det]*

Characters are decomposed by compatibility equivalence, then recomposed by canonical equivalence.

unicode_property(*?Char, ?Property*) *[nondet]*

True if *Property* is defined for *Char*. *Property* is a term Name(Value). Defined property-names are:

category(*atom*)

Unicode code category of *Char*. This is one of Cc, Cf, Cn, Co, Cs, Ll, Lm, Lo, Lt, Lu, Mc, Me, Mn, Nd, Nl, No, Pc, Pd, Pe, Pf, Pi, Po, Ps, Sc, Sk, Sm, So, Zl, Zp, Zs. When testing, a single letter stands for all its subcategories. E.g. to test form a letter, you can use

```
unicode_property(C, category('L'))
```

combining_class(*integer*)

bidi_class(*atom*)

decomp_type(*atom*)

decomp_mapping(*list(code)*)

bidi_mirrored(*bool*)

uppercase_mapping(*code*)

lowercase_mapping(*code*)

titlecase_mapping(*code*)

comb1st_index(*code*)

comb2nd_index(*code*)

comp_exclusion(*bool*)

ignorable(*bool*)

control_boundary(*bool*)

extend(*bool*)

casefold_mapping(*list(code)***)**

To be done Complete documentation

2 License

Index