Funcons-beta: Characters *

The PLanCompS Project

`Characters.cbs` | PLAIN | PRETTY

OUTLINE

---

*Suggestions for improvement: `plancomps@gmail.com`.
Reports of issues: `https://github.com/plancomps/CBS-beta/issues`.

Characters

[*Type* characters
Alias chars
Datatype unicode-characters
Alias unicode-chars
Type unicode-points
Funcon unicode-character
Alias unicode-char
Funcon unicode-point
Alias unicode
Type basic-multilingual-plane-characters
Alias bmp-chars
Type basic-multilingual-plane-points
Type iso-latin-1-characters
Alias latin-1-chars
Type iso-latin-1-points
Type ascii-characters
Alias ascii-chars
Type ascii-points
Funcon ascii-character
Alias ascii-char
Funcon utf-8
Funcon utf-16
Funcon utf-32
Funcon backspace
Funcon horizontal-tab
Funcon line-feed
Funcon form-feed
Funcon carriage-return
Funcon double-quote
Funcon single-quote
Funcon backslash]

Built-in Type characters <: values

Literal characters can be written '`C`' where *C* is any visible character other than a single-quote or backslash character, which need to be escaped as '\'' and '\\'.

Alias chars = characters

Unicode character set The set of Unicode characters and allocated points is open to extension. See https://en.wikipedia.org/wiki/Plane_(Unicode)

Built-in Datatype unicode-characters <: characters

Alias unicode-chars = unicode-characters

Built-in Type unicode-points <: bounded-integers(0, unsigned-bit-vector-maximum(21))

Built-in Funcon unicode-character(_ : unicode-points) : unicode-characters

Alias unicode-char = unicode-character

The values in unicode-characters are the values of unicode-character(*UP* : unicode-points).

Funcon unicode-point(_ : unicode-characters) : ⇒ unicode-points

Alias unicode = unicode-point

Rule unicode-point(unicode-character(*UP* : unicode-points)) ⇝ *UP*

Unicode basic multilingual plane The set of Unicode BMP characters and allocated points is open to extension.

Built-in Datatype basic-multilingual-plane-characters <: unicode-characters

Alias bmp-chars = basic-multilingual-plane-characters

Built-in Type basic-multilingual-plane-points <: bounded-integers(0, unsigned-bit-vector-maximum(17))

The values in basic-multilingual-plane-characters are the values of unicode-character(*BMPP* : basic-multilingual-plane-poin

ISO Latin-1 character set

Built-in Datatype iso-latin-1-characters <: basic-multilingual-plane-characters

Alias latin-1-chars = iso-latin-1-characters

Type iso-latin-1-points ⇝ bounded-integers(0, unsigned-bit-vector-maximum(8))

The values in iso-latin-1-characters are the values of unicode-character(*ILP* : iso-latin-1-points).

ASCII character set

Built-in Type ascii-characters <: iso-latin-1-characters

Alias ascii-chars = ascii-characters

Type ascii-points ⇝ bounded-integers(0, unsigned-bit-vector-maximum(7))

The values in ascii-characters are the values of unicode-character(*AP* : ascii-points).

Funcon ascii-character(_ : strings) : ⇒ ascii-characters?

Alias ascii-char = ascii-character

ascii-character "C" takes a string. When it consists of a single ASCII character *C* it gives the character, otherwise ( ).

Rule ascii-character [*C* : ascii-characters] ⇝ *C*

Rule $\dfrac{C : \sim \text{ascii-characters}}{\text{ascii-character } [C : \text{characters}] \rightsquigarrow ( )}$

Rule $\dfrac{\text{length}(C^*) \neq 1}{\text{ascii-character } [C^* : \text{characters}^*] \rightsquigarrow ( )}$

Character point encodings See https://en.wikipedia.org/wiki/Character_encoding

Built-in Funcon utf-8(_ : unicode-points) : ⇒(bytes, (bytes, (bytes, bytes?)?)?)

Built-in Funcon utf-16(_ : unicode-points) : ⇒(bit-vectors(16), (bit-vectors(16))?)

Built-in Funcon utf-32(_ : unicode-points) : ⇒ bit-vectors(32)

Control characters

Funcon backspace : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "0008")

Funcon horizontal-tab : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "0009")

Funcon line-feed : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "000a")

Funcon form-feed : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "000c")

Funcon carriage-return : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "000d")

Funcon double-quote : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "0022")

Funcon single-quote : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "0027")

Funcon backslash : ⇒ ascii-characters
⇝ unicode-character(hexadecimal-natural "005c")