### 6.1.1 Basic string types and values

TTCN‑3 supports the following basic string types:

NOTE 1: The general term string or string type in TTCN‑3 refers to **bitstring**, **hexstring**, **octetstring**, **charstring** and **universal charstring**.

a) **bitstring:** a type whose distinguished values are the ordered sequences of zero, one, or more bits.

Values of type **bitstring** shall be denoted by an arbitrary number (possibly zero) of the bitdigits: 0 1, preceded by a single quote ( ' ) and followed by the pair of characters 'B.

Within the quotes any number of whitespaces or any of the following C0 control characters: LF(10), VT(11), FF(12), CR(13) (see Recommendation ITU‑T T.50 [4]) (jointly called newline characters, see clause A.1.5.1) may be included. The control character shall be preceded by a backslash ("\"). Such whitespaces, control characters and backslash will be ignored for the value and length calculation.

EXAMPLE 1: '01101'B  
'0110 1001'B  
'0110\  
 1001'B

b) **hexstring:** a type whose distinguished values are the ordered sequences of zero, one, or more hexadecimal digits, each corresponding to an ordered sequence of four bits.

Values of type **hexstring** shall be denoted by an arbitrary number (possibly zero) of the hexadecimal digits (uppercase and lowercase letters can equally be used as hex digits):

0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

preceded by a single quote ( ' ) and followed by the pair of characters'H; each hexadecimal digit is used to denote the value of a semi-octet using a hexadecimal representation.

Within the quotes any number of whitespaces or any of the following C0 control characters: LF(10), VT(11), FF(12), CR(13) (see Recommendation ITU T T.50 [4]) (jointly called newline characters, see clause A.1.5.1) may be included. The control character shall be preceded by a backslash ("\"). Such whitespaces, control characters and backslash will be ignored for the value and length calculation.EXAMPLE 2: 'AB01D'H  
'ab01d'H  
'Ab01D'H  
'Ab 01 D'H  
'Ab\  
 01\  
 D'H

c) **octetstring:** a type whose distinguished values are the ordered sequences of zero or a positive even number of hexadecimal digits (every pair of digits corresponding to an ordered sequence of eight bits).

Values of type **octetstring** shall be denoted by an arbitrary, but even, number (possibly zero) of the hexadecimal digits (uppercase and lowercase letters can equally be used as hex digits):

0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

preceded by a single quote ( ' ) and followed by the pair of characters 'O; each hexadecimal digit is used to denote the value of a semi-octet using a hexadecimal representation.

Within the quotes any number of whitespaces or any of the following C0 control characters: LF(10), VT(11), FF(12), CR(13) (see Recommendation ITU T T.50 [4]) (jointly called newline characters, see clause A.1.5.1) may be included. The control character shall be preceded by a backslash ("\"). Such whitespaces, control characters and backslash will be ignored for the value and length calculation.EXAMPLE 3: 'FF96'O  
'ff96'O  
'Ff96'O  
'Ff 96'O  
'Ff\  
 96'O

d) **charstring:** are types whose distinguished values are zero, one, or more characters of the version of Recommendation ITU‑T T.50 [4] complying with the International Reference Version (IRV) as specified in clause 8.2 of Recommendation ITU‑T T.50 [4].

NOTE 2: The IRV version of Recommendation ITU‑T T.50 [4] is equivalent to the IRV version of the International Reference Alphabet (former International Alphabet No.5 - IA5), described in Recommendation ITU‑T T.50 [4].

Values of **charstring** type shall be denoted by an arbitrary number (possibly zero) of non-control characters from the relevant character set, preceded and followed by double quote ("). Graphical characters include the range from SP(32) to TILDE (126). Values of **charstring** type can also be calculated using the predefined conversion function int2char with the positive integer value of their encoding as argument (see clause C.1).

NOTE 3: The predefined conversion function is able to return single-character-length values only.

In cases where it is necessary to define strings that include the character double quote (") the character is represented by a pair of double quotes on the same line with no intervening space characters.

EXAMPLE 4: The charstring "ab"cd" is written in TTCN-3 code as in the following constant declaration. Each of the 3 quote characters that are part of the string is preceded by an extra quote character and the whole character string is delimited by quote characters, e.g.  
**var** **charstring** vl\_char:= """ab""cd""";

e) The character string type preceded by the keyword **universal** denotes types whose distinguished values are zero, one, or more characters from ISO/IEC 10646 [2].

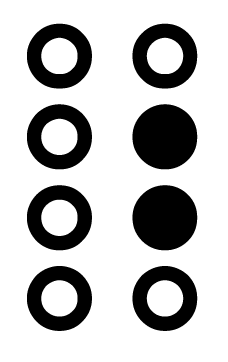
**universal** **charstring** values can also be denoted by an arbitrary number (possibly zero) of characters from the relevant character set, preceded and followed by double quote ("), calculated using a predefined conversion function (see clause C.1.2) with the positive integer value of their encoding as argument, by a "quadruple" or using the USI-like notation.

NOTE 4: If applying the double quote format all characters from any character set defined in ISO/IEC 10646 [2] are allowed. Users should be aware of the character set capabilities of their editing tool and the TTCN‑3 module transfer syntax UTF-8 (see clause 8).

NOTE 5: The predefined conversion function is able to return single-character-length values only.

In cases where it is necessary to define strings that include the character double quote (") the character is represented by a pair of double quotes on the same line with no intervening space characters.

The "quadruple" is only capable to denote a single character and denotes the character by the decimal values of its group, plane, row and cell according to ISO/IEC 10646 [2], preceded by the keyword **char** included into a pair of brackets and separated by commas (e.g. **char** ( 0, 0, 1, 113) denotes the Latin small letter u with double acute: "ű"). In cases where it is necessary to denote the character double quote (") in a string assigned according to the first method (within double quotes), the character is represented by a pair of double quotes on the same line with no intervening space characters. The two methods may be mixed within a single notation for a string value by using the concatenation operator.

EXAMPLE 5: The expression: "the Braille character" & **char** (0, 0, 40, 48) & "looks like this" represents the literal string: the Braille character  looks like this.

The UCS sequence identifier-like (USI-like) notation (see also clause 6.6 of ISO/IEC 10646 [2]) can be used to denote 1..N characters, using their short identifiers of code point (similar to UIDs described in clause 6.5 of ISO/IEC 10646 [2]). The USI-like notation is composed of the keyword **char** followed by parentheses. The parentheses enclose a comma-separated list of short identifiers . Each short identifier represents a single character and it shall be composed of a letter **U** or **u** followed by an optional "+" PLUS SIGN character, followed by 1..8 hexadecimal digits. The hexadecimal digits represent the numeric code point of the character. (e.g. char(U0171) denotes the Latin small letter u with double acute: "ű"). In the USI-like notation, the leading zeroes can be omitted, (i.e. char(U171)is equal to char(U0171)).

EXAMPLE 6: The expression: **char** (U4E2D, U56FD) represents the literal string: 中国.

NOTE 6: Control characters can be denoted by using the predefined conversion function, the quadruple form or the USI-like notation.

By default, **universal charstring** shall conform to the UTF-32 encoding specified in clause 9.3 of ISO/IEC 10646 [2].

NOTE 7: UTF-32 is an encoding format, which represents any UCS character on a fixed, 32 bits‑length field.

This default encoding can be overridden using the defined variant attributes (see clause 27.5). The useful character string types utf8string, bmpstring, utf16string and iso8859string using these attributes are defined in annex E.

A.1.6.6 Value

Value ::= [PredefinedValue](#TPredefinedValue) | [ReferencedValue](#TReferencedValue)

PredefinedValue ::= [Bstring](#TBstring) |

[BooleanValue](#TBooleanValue) |

[CharStringValue](#TCharStringValue) |

[Number](#TNumber) | /\* IntegerValue \*/

[Ostring](#TOstring) |

[Hstring](#THstring) |

[VerdictTypeValue](#TVerdictTypeValue) |

[Identifier](#TIdentifier) | /\* EnumeratedValue \*/

[FloatValue](#TFloatValue) |

[AddressValue](#TAddressValue) |

[OmitKeyword](#TOmitKeyword)

BooleanValue ::= "true" | "false"

VerdictTypeValue ::= "pass" |

"fail" |

"inconc" |

"none" |

"error"

CharStringValue ::= [Cstring](#TCstring) | [Quadruple](#TQuadruple) | USIlikeNotation

Quadruple ::= [CharKeyword](#TCharKeyword) "(" [Number](#TNumber) "," [Number](#TNumber) "," [Number](#TNumber) "," [Number](#TNumber) ")"

USIlikeNotation ::= [CharKeyword](#TCharKeyword) "(" UIDlike { "," [UID](#TNumber)like } ")"

UIDlike ::= (U|u) {"+"} {Hex}#(1,8)

CharKeyword ::= "char"

FloatValue ::= [FloatDotNotation](#TFloatDotNotation) |

[FloatENotation](#TFloatENotation) |

[NaNKeyword](#TNaNKeyword)

NaNKeyword ::= "not\_a\_number"

FloatDotNotation ::= [Number](#TNumber) [Dot](#TDot) [DecimalNumber](#TDecimalNumber)

FloatENotation ::= [Number](#TNumber) [[Dot](#TDot) [DecimalNumber](#TDecimalNumber)] [Exponential](#TExponential) [[Minus](#TMinus)]

[Number](#TNumber)

Exponential ::= "E"

ReferencedValue ::= [ExtendedIdentifier](#TExtendedIdentifier) [[ExtendedFieldReference](#TExtendedFieldReference)]

Number ::= ([NonZeroNum](#TNonZeroNum) {[Num](#TNum)}) | "0"

NonZeroNum ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

DecimalNumber ::= {[Num](#TNum)}+

Num ::= "0" | [NonZeroNum](#TNonZeroNum)

Bstring ::= "'" {[Bin](#TBin) | BinSpace } "'" "B"

Bin ::= "0" | "1"

Hstring ::= "'" {[Hex](#THex) | BinSpace } "'" "H"

Hex ::= [Num](#TNum) | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" |

"d" | "e" | "f"

Ostring ::= "'" {[Oct](#TOct) | BinSpace } "'" "O"

Oct ::= [Hex](#THex) [Hex](#THex)

Cstring ::= """ {[Char](#TChar)} """

Char ::= /\* REFERENCE - A character defined by the relevant CharacterString type. For charstring a character from the character set defined in ITU-T T.50. For universal charstring a character from any character set defined in ISO/IEC 10646 \*/

Identifier ::= [Alpha](#TAlpha) {[AlphaNum](#TAlphaNum) | [Underscore](#TUnderscore)}

Alpha ::= [UpperAlpha](#TUpperAlpha) | [LowerAlpha](#TLowerAlpha)

AlphaNum ::= [Alpha](#TAlpha) | [Num](#TNum)

UpperAlpha ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |

"J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |

"S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

LowerAlpha ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |

"j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |

"s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

ExtendedAlphaNum ::= /\* REFERENCE - A graphical character from the BASIC LATIN or from the LATIN-1 SUPPLEMENT character sets defined in ISO/IEC 10646 (characters from char (0,0,0,32) to char (0,0,0,126), from char (0,0,0,161) to char (0,0,0,172) and from char (0,0,0,174) to char (0,0,0,255) \*/

FreeText ::= """ {[ExtendedAlphaNum](#TExtendedAlphaNum)} """

AddressValue ::= "null"

OmitKeyword ::= "omit"

BinSpace ::= " " | "\" NLChar NLChar ::= /\* REFERENCE - Any of the following C0 control characters: LF(10), VT(11), FF(12), CR(13) (see Recommendation ITU T T.50 [4]) (jointly called newline characters, see clause A.1.5.1) from the character set defined in ITU-T T.50.