5.2 Informative Comments

This section provides a brief description of each rule listed in the normative specification.

expression = ws [definitionStatus ws] subExpression ws	
	An expression often consists only of a subExpression. However, in some cases a definition status is needed to state whether the clinical meaning being represented is equivalent to or a subtype of the subExpression. If no definition status is included, the clinical meaning being represented is assumed to be ‘equivalent to’ the subExpression.
subExpression = focusConcept [ws ":" ws refinement]	
	A subexpression consists of one or more focus concepts, optionally followed by a refinement. The meaning of the expression is a subtype of all the focus concepts constrained by the refinement. Note that where there is a requirement for multiple separately qualified concepts to be present these are expressed in attribute groups within a refinement of a general concept such as " situation with explicit context ".
definitionStatus = equivalentTo / subtypeOf	
	The definition status states whether the clinical meaning being expressed is equivalent to or a subtype of the given expression representation.
equivalentTo = "==="	
	A definition status of equivalentTo (i.e. “===”) indicates that the clinical meaning being represented is semantically equivalent to the given expression. If no definition status is included in the expression, this definition status is assumed.
subtypeOf = "<<<"	
	A definition status of subtypeOf (i.e. “<<<”) indicates that the clinical meaning being represented is a semantic subtype (or subclass) of the given expression.
focusConcept = conceptReference *(ws "+" ws conceptReference)	
	A focusConcept consists of one or more concept references separated by ‘plus’ signs.
conceptReference = conceptId [ws " **|** " ws term ws " **|** "]	
	A conceptReference is represented by a ConceptId optionally followed by a term enclosed by a pair of "|" characters. Whitespace before or after the ConceptId is ignored as is any whitespace between the initial "character" and the first non-whitespace character in the term or between the last non-whitespace character and second "|" character.
conceptId = sctId	
	The ConceptId must be a valid SNOMED CT identifier for a concept. The initial digit may not be zero. The smallest number of digits is six, and the maximum is 18.
term = nonwsNonPipe *(*SP nonwsNonPipe)	
	The term must be the term from any SNOMED CT description that is associated with the concept identified by the preceding concept identifier. For example, the term could be the preferred synonym for a given dialect. The term may include valid UTF-8 characters except for the pipe "|" character. The term begins with the first non-whitespace character following the starting "|" character and ends with the last non-whitespace character preceding the next "|" character.
refinement = (attributeSet / attributeGroup) *(ws ["," ws] attributeGroup)	
	A refinement contains all the grouped and ungrouped attributes that refine the meaning of the containing expression.
attributeGroup = "**{**" ws attributeSet ws "**}**"	
	An attribute group contains a collection of attributes that operate together as part of the refinement of the containing expression.
attributeSet = attribute *(ws " **,** " ws attribute)	
	An attribute set contains one or more attribute name-value pairs, separated by commas.
attribute = attributeName ws "=" ws attributeValue	
	An attribute is a name-value pair expressing a single refinement of the containing expression.
attributeName = conceptReference	
	The attribute name is the name of an attribute (or relationship type) to which a value is applied to refine the meaning of a containing expression. The attribute name is represented in the same way as other concept references.
attributeValue = expressionValue / QM stringValue QM / “#” numericValue / booleanValue	

	An attribute value is either an expression, a string-based concrete value enclosed in quotation marks, a numeric concrete value (i.e. integer or decimal) preceded by a “#”, or a boolean.
expressionValue = conceptReference / "**(**" ws subExpression ws "**)**"	
	An expression value is either a single concept reference without brackets, or a more complex expression enclosed in round brackets.
stringValue = 1*(anyNonEscapedChar / escapedChar)	
	A string value includes one or more printable ASCII characters (these are also valid UTF8 characters encoded as one octet) and/or UTF8 characters encoded as 2- 3- or 4-octet sequences. Quotes and backslash characters must be preceded by the escape character (“\”).
numericValue = ["-"/"+"] (decimalValue / integerValue)	
	A numeric value is either an integer or a decimal. Positive numbers optionally start with a plus sign ("+"), while negative integers begin with a minus sign ("-").
integerValue = digitNonZero *digit / zero	
	An integer value is either starts with a non-zero digit followed by zero to many additional digits, or is the integer zero itself.
decimalValue = integerValue "." 1*digit	
	A decimal value starts with an integer. This is followed by a decimal point and one to many digits.
booleanValue = true / false	
	A boolean value is either true or false.
true = ("t"/"T") ("r"/"R") ("u"/"U") ("e"/"E")	
	The boolean value of true is represented as "TRUE" using any combination of upper or lower characters.
false = ("f"/"F") ("a"/"A") ("l"/"L") ("s"/"S") ("e"/"E")	
	The boolean value of false is represented as "FALSE" using any combination of upper and lower characters.
sctId = digitNonZero 5*17(digit)	
	A SNOMED CT id is used to represent an attribute id or a concept id. The initial digit may not be zero. The smallest number of digits is six, and the maximum is 18.
ws = *(SP | HTAB | CR | LF)	
	Optional whitespace characters (space, tab, carriage return and linefeed) are ignored everywhere in the expression except: 1. Whitespace within a conceptId is an error. (**Note:** Whitespace before or after the last digit of a valid Identifier is ignored.) 2. Non-consecutive spaces within a term are treated as a significant character of the term. (**Note:** Whitespace before the first or after the last non-whitespace character of a term is ignored.) 3. Whitespace within the quotation marks of a concrete value is treated as a significant character.
SP = %x20	
	Space character.
HTAB = %x09	
	Tab character.
CR = %x0D	
	Carriage return character.
LF = %x0A	
	Line feed character.
QM = %x22 ; quotation mark	
	Quotation mark character.
BS = %x5C	
	Back slash character.
digit = %x30-39	

	Any digit 0 through 9.
zero = %x30	
	The digit 0.
digitNonZero = %x31-39	
	Digits 1 through 9, but excluding 0. The first character of a concept identifier is constrained to a digit other than zero.
nonwsNonPipe= %x21-7B / %x7D-7E / UTF8-2 / UTF8-3 / UTF8-4	
	Non whitespace (and non pipe) includes printable ASCII characters (these are also valid UTF8 characters encoded as one octet) and also includes all UTF8 characters encoded as 2- 3- or 4-octet sequences. It excludes space (which is %x20) and the pipe character "|" (which is % x7C), and excludes CR, LF, HTAB and other ASCII control codes. See RFC 3629 (UTF-8, a transformation format of ISO 10646 authored by the Network Working Group).
anyNonEscapedChar = HTAB / CR / LF / %x20-21 / %x23-5B / %x5D-7E / UTF8-2 / UTF8-3 / UTF8-4	
	anyNonEscapedChar includes any printable ASCII characters which do not need to be preceded by an escape character (i.e. “\”). This includes valid UTF8 characters encoded as one octet and all UTF8 characters encoded as 2, 3 or 4 octet sequences. It does, however, exclude the quotation mark (“) and the backslash (\). See RFC 3629 (UTF-8, a transformation format of ISO 10646 authored by the Network Working Group).
escapedChar = BS QM / BS BS	
	The double quotation mark and the back slash character must both be escaped within a string-based concrete value by preceding them with a back slash.
UTF8-2 = %xC2-DF UTF8-tail	
	UTF8 characters encoded as 2-octet sequences.
UTF8-3 = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2(UTF8-tail) / %xED %x80-9F UTF8-tail / %xEE-EF 2(UTF8-tail)	
	UTF8 characters encoded as 3-octet sequences.
UTF8-4 = %xF0 %x90-BF 2(UTF8-tail) / %xF1-F3 3(UTF8-tail) / %xF4 %x80-8F 2(UTF8-tail)	
	UTF8 characters encoded as 4-octet sequences.
UTF8-tail = %x80-BF	
	UTF8 characters encoded as 8-octet sequences.