# 6.1. Expression Template Language

The formal syntax for SNOMED CT expression templates (v1) is shown below. This syntax is derived by combining:

- Compositional Grammar v2.3.1,
- Template Syntax v1, and
- Expression Constraint Language v1.3.

As explained in 6. SNOMED CT Language Templates, references to the Template Syntax are also added into the Compositional Grammar rules to enable slots to be included within an expression template. Please note that rules that appear in both Compositional Grammar and the Expression Constraint Language (e.g. conceptId and term) are removed from the Expression Constraint Language (by commenting out) to avoid duplication.

> ***; Compositional Grammar v2.3.1 with slot references (in blue)***
>
> ***expressionTemplate*** *= ws [ (definitionStatus* ***/ tokenReplacementSlot)*** *ws] subExpression ws*
>
> ***subExpression*** *= focusConcept [ws ":" ws refinement]*
>
> ***definitionStatus*** *= equivalentTo / subtypeOf*
>
> ***equivalentTo*** *= "==="*
>
> ***subtypeOf*** *= "<<<"*
>
> ***focusConcept*** *=* ***[templateInformationSlot ws]*** *conceptReference *(ws "+" ws* ***[templateInformationSlot ws]*** *conceptReference)*
>
> ***conceptReference*** *=* ***conceptReplacementSlot / expressionReplacementSlot / (*** *conceptId [ws "|" ws term ws "|"]* ***)***
>
> ***conceptId*** *= sctId*
>
> ***term*** *= nonwsNonPipe *( *SP nonwsNonPipe )*
>
> ***refinement*** *= (attributeSet / attributeGroup) *( ws [";" ws] attributeGroup )*
>
> ***attributeGroup*** *=* ***[ templateInformationSlot ws ]*** *"{" ws attributeSet ws "}"*
>
> ***attributeSet*** *= attribute *(ws "," ws attribute)*
>
> ***attribute*** *=* ***[ templateInformationSlot ws ]*** *attributeName ws "=" ws attributeValue*
>
> ***attributeName*** *= conceptReference*
>
> ***attributeValue*** *= expressionValue / QM stringValue QM / "#" numericValue* ***/ concreteValueReplacementSlot***
>
> ***expressionValue*** *= conceptReference / "(" ws subExpression ws ")"*
>
> ***stringValue*** *= 1*(anyNonEscapedChar / escapedChar)*
>
> ***numericValue*** *= ["-"/"+"] (decimalValue / integerValue)*
>
> ***integerValue*** *= digitNonZero *digit / zero*
>
> ***decimalValue*** *= integerValue "." 1*digit*
>
> ***sctId*** *= digitNonZero 5*17( digit )*
>
> ***ws*** *= *( SP / HTAB / CR / LF ) ; optional white space*
>
> ***SP*** *= %x20 ; space*
>
> ***HTAB*** *= %x09 ; tab*
>
> ***CR*** *= %x0D ; carriage return*
>
> ***LF*** *= %x0A ; line feed*
>
> ***QM*** *= %x22 ; quotation mark*
>
> ***BS*** *= %x5C ; back slash*
>
> ***digit*** *= %x30-39*
>
> ***zero*** *= %x30*

***digitNonZero*** *= %x31-39*

***nonwsNonPipe*** *= %x21-7B / %x7D-7E / UTF8-2 / UTF8-3 / UTF8-4*

***anyNonEscapedChar*** *= HTAB / CR / LF / %x20-21 / %x23-5B / %x5D-7E / UTF8-2 / UTF8-3 / UTF8-4*

***escapedChar*** *= BS QM / BS BS*

***UTF8-2*** *= %xC2-DF UTF8-tail*

***UTF8-3*** *= %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) / %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )*

***UTF8-4*** *= %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) / %xF4 %x80-8F 2( UTF8-tail )*

***UTF8-tail*** *= %x80-BF*

***; Template Syntax v1***

***templateSlot*** *= templateReplacementSlot / templateInformationSlot*

***templateReplacementSlot*** *= conceptReplacementSlot / expressionReplacementSlot / tokenReplacementSlot / concreteValueReplacementSlot*

***conceptReplacementSlot*** *= "[[" ws "+" ws "id" ws [ "(" ws expressionConstraint ws ")" ws] [slotName ws] "]]"*

***expressionReplacementSlot*** *= "[[" ws "+" ws ["scg" ws] [ "(" ws expressionConstraint ws ")" ws] [slotName ws] "]]"*

***tokenReplacementSlot*** *= "[[" ws "+" ws "tok" ws [ "(" ws slotTokenSet ws ")" ws] [slotName ws] "]]"*

***concreteValueReplacementSlot*** *= stringReplacementSlot / integerReplacementSlot / decimalReplacementSlot*

***stringReplacementSlot*** *= "[[" ws "+" ws "str" ws [ "(" ws slotStringSet ws ")" ws] [slotName ws] "]]"*

***integerReplacementSlot*** *= "[[" ws "+" ws "int" ws [ "(" ws slotIntegerSet ws ")" ws] [slotName ws] "]]"*

***decimalReplacementSlot*** *= "[[" ws "+" ws "dec" ws [ "(" ws slotDecimalSet ws ")" ws] [slotName ws] "]]"*

***slotTokenSet*** *= slotToken *(mws slotToken)*

***slotStringSet*** *= slotString *(mws slotString)*

***slotIntegerSet*** *= ( "#" integerValue / slotIntegerRange) *(mws ("#" integerValue / slotIntegerRange))*

***slotDecimalSet*** *= ( "#" decimalValue / slotDecimalRange) *(mws ("#" decimalValue / slotDecimalRange))*

***slotIntegerRange*** *= ( slotIntegerMinimum to [ slotIntegerMaximum ] ) / ( to slotIntegerMaximum )*

***slotIntegerMinimum*** *= [ exclusiveMinimum ] "#" integerValue*

***slotIntegerMaximum*** *= [ exclusiveMaximum ] "#" integerValue*

***slotDecimalRange*** *= ( slotDecimalMinimum to [ slotDecimalMaximum ] ) / ( to slotDecimalMaximum )*

***slotDecimalMinimum*** *= [ exclusiveMinimum ] "#" DecimalValue*

***slotDecimalMaximum*** *= [ exclusiveMaximum ] "#" DecimalValue*

***exclusiveMinimum*** *= ">"*

***exclusiveMaximum*** *= "<"*

***slotName*** *= "@" (nonQuoteStringValue / slotString)*

***slotToken*** *= definitionStatus / memberOf / constraintOperator / conjunction / disjunction / exclusion / reverseFlag / expressionComparisonOperator / numericComparisonOperator / stringComparisonOperator*

***slotString*** *= QM stringValue QM*

***nonQuoteStringValue*** *= *(%x21 / %x23-26 / %x28-3F / %x41-5A / %x5C / %x5E-7E) ; string with no ws, quotes, at or square brackets*

***templateInformationSlot =*** *"[[" ws slotInformation ws "]]"*

***slotInformation*** *= [cardinality ws] [slotName ws]*

***; Expression Constraint Language v1.3 - Note that some rules are commented out because they are repeated in the Compositional Grammar rules above.***

***expressionConstraint*** *= ws ( refinedExpressionConstraint / compoundExpressionConstraint / dottedExpressionConstraint / subExpressionConstraint ) ws*

***refinedExpressionConstraint*** *= subExpressionConstraint ws ":" ws eclRefinement*

***compoundExpressionConstraint*** *= conjunctionExpressionConstraint / disjunctionExpressionConstraint / exclusionExpressionConstraint*

***conjunctionExpressionConstraint*** *= subExpressionConstraint 1*(ws conjunction ws subExpressionConstraint)*

***disjunctionExpressionConstraint*** *= subExpressionConstraint 1*(ws disjunction ws subExpressionConstraint)*

***exclusionExpressionConstraint*** *= subExpressionConstraint ws exclusion ws subExpressionConstraint*

***dottedExpressionConstraint*** *= subExpressionConstraint 1*(ws dottedExpressionAttribute)*

***dottedExpressionAttribute*** *= dot ws eclAttributeName*

***subExpressionConstraint*** *= [constraintOperator ws] [memberOf ws] (eclFocusConcept / "(" ws expressionConstraint ws ")")*

***eclFocusConcept*** *= eclConceptReference / wildCard*

***dot = "."***

***memberOf*** *= "^"*

***eclConceptReference*** *= conceptId [ws "|" ws term ws "|"]*

***; conceptId*** *= sctId*

***; term*** *= 1*nonwsNonPipe *( 1*SP 1*nonwsNonPipe )*

***wildCard*** *= "*"*

***constraintOperator*** *= childOf / descendantOrSelfOf / descendantOf / parentOf / ancestorOrSelfOf / ancestorOf*

***descendantOf*** *= "<"*

***descendantOrSelfOf*** *= "<<"*

***childOf*** *= "<!"*

***ancestorOf*** *= ">"*

***ancestorOrSelfOf*** *= ">>"*

***parentOf*** *= ">!"*

***conjunction*** *= (("a"/"A") ("n"/"N") ("d"/"D") mws) / ","*

***disjunction*** *= ("o"/"O") ("r"/"R") mws*

***exclusion*** *= ("m"/"M") ("i"/"I") ("n"/"N") ("u"/"U") ("s"/"S") mws*

***eclRefinement*** *= subRefinement ws [conjunctionRefinementSet / disjunctionRefinementSet]*

***conjunctionRefinementSet*** *= 1*(ws conjunction ws subRefinement)*

***disjunctionRefinementSet*** *= 1*(ws disjunction ws subRefinement)*

***subRefinement*** *= eclAttributeSet / eclAttributeGroup / "(" ws eclRefinement ws ")"*

***eclAttributeSet*** *= subAttributeSet ws [conjunctionAttributeSet / disjunctionAttributeSet]*

***conjunctionAttributeSet*** *= 1*(ws conjunction ws subAttributeSet)*

***disjunctionAttributeSet*** *= 1*(ws disjunction ws subAttributeSet)*

***subAttributeSet*** *= eclAttribute / "(" ws eclAttributeSet ws ")"*

***eclAttributeGroup*** *= ["[" cardinality "]" ws] "{" ws eclAttributeSet ws "}"*

***eclAttribute*** *= ["[" cardinality "]" ws] [reverseFlag ws] eclAttributeName ws (expressionComparisonOperator ws subExpressionConstraint / numericComparisonOperator ws "#" numericValue / stringComparisonOperator ws QM stringValue QM)*

***cardinality*** *= minValue to maxValue*

***minValue*** *= nonNegativeIntegerValue*

***to*** *= ".."*

***maxValue*** *= nonNegativeIntegerValue / many*

***many*** *= "*"*

***reverseFlag*** *= "R"*

***eclAttributeName*** *= subExpressionConstraint*

***expressionComparisonOperator*** *= "=" / "!="*

***numericComparisonOperator*** *= "=" / "!=" / "<=" / "<" / ">=" / ">"*

***stringComparisonOperator*** *= "=" / "!="*

***; numericValue*** *= ["-"/"+"] (decimalValue / integerValue)*

***; stringValue*** *= 1*(anyNonEscapedChar / escapedChar)*

***; integerValue*** *= digitNonZero *digit / zero*

***; decimalValue*** *= integerValue "." 1*digit*

***nonNegativeIntegerValue*** *= (digitNonZero *digit ) / zero*

***; sctId*** *= digitNonZero 5*17( digit )*

***; ws*** *= *( SP / HTAB / CR / LF / comment ) ; optional white space*

***mws*** *= 1*( SP / HTAB / CR / LF / comment ) ; mandatory white space*

***comment*** *= "/*" *(nonStarChar / starWithNonFSlash) "*/"*

***nonStarChar*** *= SP / HTAB / CR / LF / %x21-29 / %x2B-7E /UTF8-2 / UTF8-3 / UTF8-4*

***starWithNonFSlash*** *= %x2A nonFSlash*

***nonFSlash*** *= SP / HTAB / CR / LF / %x21-2E / %x30-7E /UTF8-2 / UTF8-3 / UTF8-4*

***; SP*** *= %x20 ; space*

***; HTAB*** *= %x09 ; tab*

***; CR*** *= %x0D ; carriage return*

***; LF*** *= %x0A ; line feed*

***; QM*** *= %x22 ; quotation mark*

***; BS*** *= %x5C ; back slash*

***; digit*** *= %x30-39*

***; zero*** *= %x30*

***; digitNonZero*** *= %x31-39*

***; nonwsNonPipe*** *= %x21-7B / %x7D-7E / UTF8-2 / UTF8-3 / UTF8-4*

***; anyNonEscapedChar*** *= SP / HTAB / CR / LF / %x20-21 / %x23-5B / %x5D-7E / UTF8-2 / UTF8-3 / UTF8-4*

***; escapedChar*** *= BS QM / BS BS*

***; UTF8-2*** *= %xC2-DF UTF8-tail*

***; UTF8-3*** *= %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) / %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )*

```
; UTF8-4 = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) / %xF4 %x80-8F 2( UTF8-tail )

; UTF8-tail = %x80-BF
```