# XPATH+ EBNF

LocationPath ::= RelativeLocationPath
| AbsoluteLocationPath

AbsoluteLocationPath ::= '/' RelativeLocationPath?
| AbbreviatedAbsoluteLocationPath

RelativeLocationPath ::= Step
| RelativeLocationPath '/' Step
| AbbreviatedRelativeLocationPath

Step ::= AxisSpecifier NodeTest Predicate*
| AbbreviatedStep

AxisSpecifier ::= AxisName '::'
| AbbreviatedAxisSpecifier

AxisName ::= 'ancestor'
| 'ancestor-or-self'
| 'attribute'
| 'child'
| 'descendant'
| 'descendant-or-self'
| 'following'
| 'following-sibling'
| 'namespace'
| 'parent'
| 'preceding'
| 'preceding-sibling'
| 'self'
| **'link-destination'**
| **'link-source'**

NodeTest ::= NameTest
| NodeType '(' ')'
| 'processing-instruction' '(' Literal ')'

Predicate ::= '[' PredicateExpr ']'
| **‘[[‘ PredicateExpr ']]'**

PredicateExpr ::= Expr

AbbreviatedAbsoluteLocationPath ::= '//' RelativeLocationPath
| **'///' RelativeLocationPath**

AbbreviatedRelativeLocationPath ::= RelativeLocationPath '//' Step

AbbreviatedStep ::= '.'
| '..'
| **'...'**

AbbreviatedAxisSpecifier ::= '@'?

Expr ::= OrExpr

PrimaryExpr ::= VariableReference
| '(' Expr ')'
| Literal
| Number
| FunctionCall

FunctionCall ::= FunctionName '(' ( Argument ( ',' Argument )* )? ')'

Argument ::= Expr

UnionExpr ::= PathExpr
| UnionExpr '|' PathExpr

PathExpr ::= LocationPath
| FilterExpr
| FilterExpr '/' RelativeLocationPath
| FilterExpr '//' RelativeLocationPath

FilterExpr ::= PrimaryExpr
| FilterExpr Predicate

OrExpr ::= AndExpr

```
                                    | OrExpr 'or' AndExpr
AndExpr                             ::= EqualityExpr
                                    | AndExpr 'and' EqualityExpr
EqualityExpr                        ::= RelationalExpr
                                    | EqualityExpr '=' RelationalExpr
                                    | EqualityExpr '!=' RelationalExpr
RelationalExpr                      ::= AdditiveExpr
                                    | RelationalExpr '<' AdditiveExpr
                                    | RelationalExpr '>' AdditiveExpr
                                    | RelationalExpr '<=' AdditiveExpr
                                    | RelationalExpr '>=' AdditiveExpr
AdditiveExpr                        ::= MultiplicativeExpr
                                    | AdditiveExpr '+' MultiplicativeExpr
                                    | AdditiveExpr '-' MultiplicativeExpr
MultiplicativeExpr                  ::= UnaryExpr
                                    | MultiplicativeExpr MultiplyOperator UnaryExpr
                                    | MultiplicativeExpr 'div' UnaryExpr
                                    | MultiplicativeExpr 'mod' UnaryExpr
UnaryExpr                           ::= UnionExpr
                                    | '-' UnaryExpr
ExprToken                           ::= '(' | ')' | '[' | ']' | '.' | '..' | '@' | ',' | '::' | ‘…’ | ‘[[‘ | ’]]’
                | NameTest
                | NodeType
                | Operator
                | FunctionName
                | AxisName
                | Literal
                | Number
                | VariableReference
Literal             ::=             '"' [^"]* '"'
                | "'" [^']* "'"
Number              ::=             Digits ('.' Digits?)?
                | '.' Digits
Digits              ::=             [0-9]+
Operator            ::=             OperatorName
                | MultiplyOperator
                | '/' | '//' | '///' | '|' | '+' | '-' | '=' | '!=' | '<' | '<='
                | '>' | '>='
OperatorName                        ::= 'and' | 'or' | 'mod' | 'div'
MultiplyOperator                    ::= '*'
FunctionName                        ::= QName - NodeType
VariableReference                   ::= '$' QName
NameTest            ::=             '*'
                | NCName ':' '*'
                | QName
NodeType            ::=             'comment'
                | 'text'
                | 'processing-instruction'
                | 'node'
ExprWhitespace                      ::= S
```