Appendix A

The Grammar of approXQL

For the specification of the grammar of approXQL, shown in Figure A.1, we use the Extended Backus-Naur Form (EBNF) proposed by N. Wirth [Wir77]. In EBNF, [ ] brackets indicate zero or one occurrences of the enclosed expression; { } brackets indicate zero or more occurrences. Additionally, ( ) parentheses can be used to group expressions.

The signs enclosed by quotation marks and the capitalized words are the terminals of the grammar. The structure of a `NAME` is defined by the production for the non-terminal `Name` in the XML specification [BPSM00]. A `WORD` is a character sequence that starts with a letter and continues with letters and digits. The structure of a `NUMBER` follows the common rules for integers and real numbers. A `TOKEN` is any sequence of characters, including names, words, and numbers. Generic tokens are one basis for extending the language by new data types.

The dots at the end of the productions for the non-terminals `StructType`, `DataType`, and `Operator` indicate that these productions can be extended. For example, a new data type `person_name` and a new operator `sounds_like` may be added. A back-end module of the parser ensures that each operator is defined for the data type it is used with, and that the parsed `TOKEN`s fulfill the syntax requirements specified by the type.

```
          Query ::= StructSelection [ Containment ]
    Containment ::= '/' PathExpression | '[' Expression ']'
 PathExpression ::= Query | DataSelection | '(' Expression ')'
     Expression ::= PathExpression | Disjunction
    Disjunction ::= Conjunction { 'or' Conjunction }
    Conjunction ::= Expression { 'and' Expression }

StructSelection ::= [ InsModifier ] TypedSelector [ ValModifier ] [ DelModifier ]
  TypedSelector ::= [ StructType ':' ] StructSelector
     StructType ::= 'struct' | 'attribute' | 'element' | ...
 StructSelector ::= NAME | '(' NAME { '|' NAME } ')'

  DataSelection ::= [ InsModifier ] Predicate [ ValModifier ] [ DelModifier ]
      Predicate ::= [ [ DataType ] Operator ] DataSelector
       DataType ::= 'data' | 'text' | ...
       Operator ::= '=' | '<' | '<=' | '>' | '=>' | ...
   DataSelector ::= Data | '(' Data { '|' Data } ')'
           Data ::= Phrase | NUMBER | TOKEN
         Phrase ::= '"' AlphaNum { AlphaNum } '"'
       AlphaNum ::= WORD | NUMBER

    InsModifier ::= '*' | '!'
    ValModifier ::= '*' | '!'
    DelModifier ::= ':' ( '*' | '!' )
```

Figure A.1: The grammar of approXQL.