Type safe SPARQL Query Parsing with ANTLR

Bachelor‘s Thesis

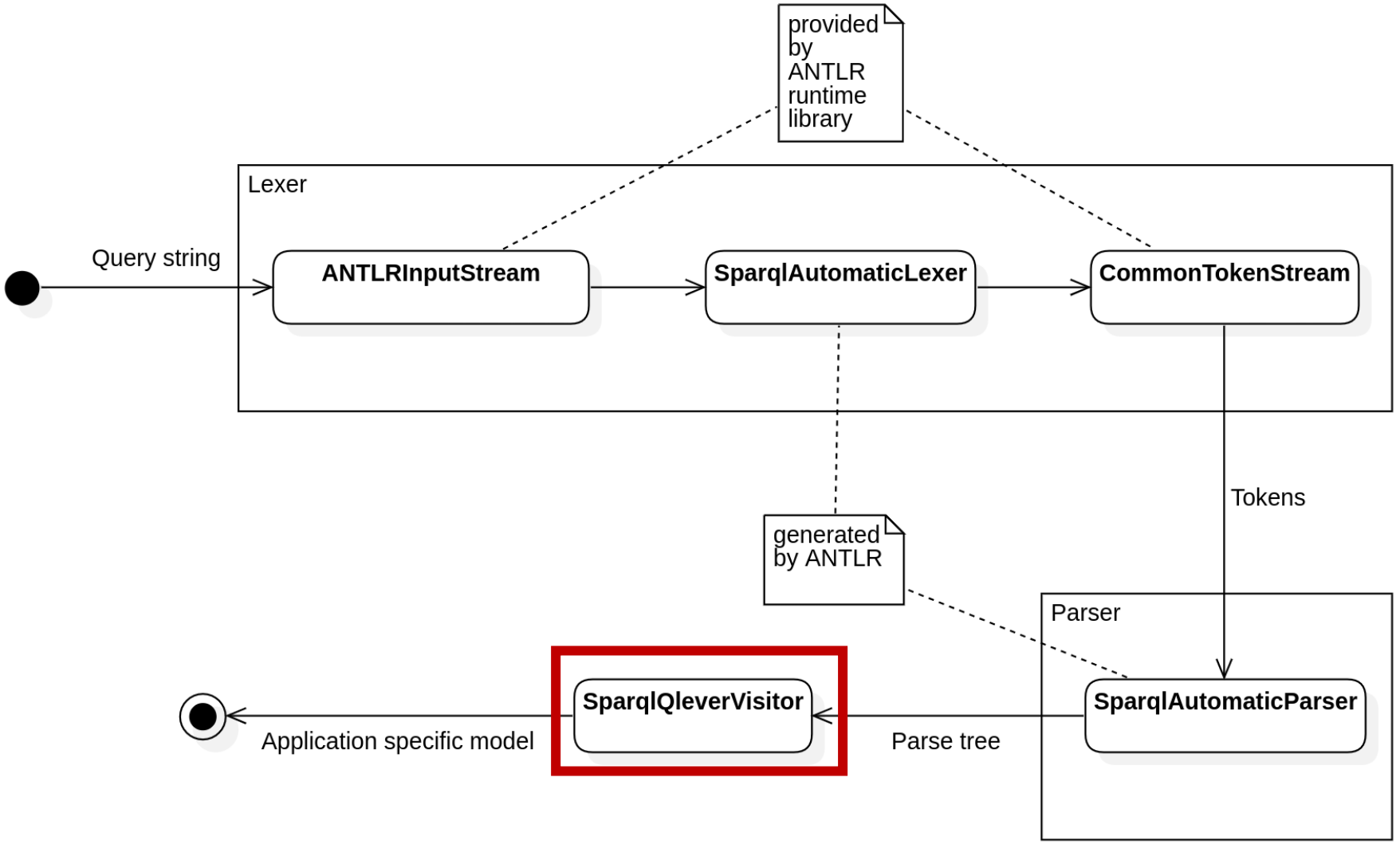
Julian Mundhahs

October 4th 2022

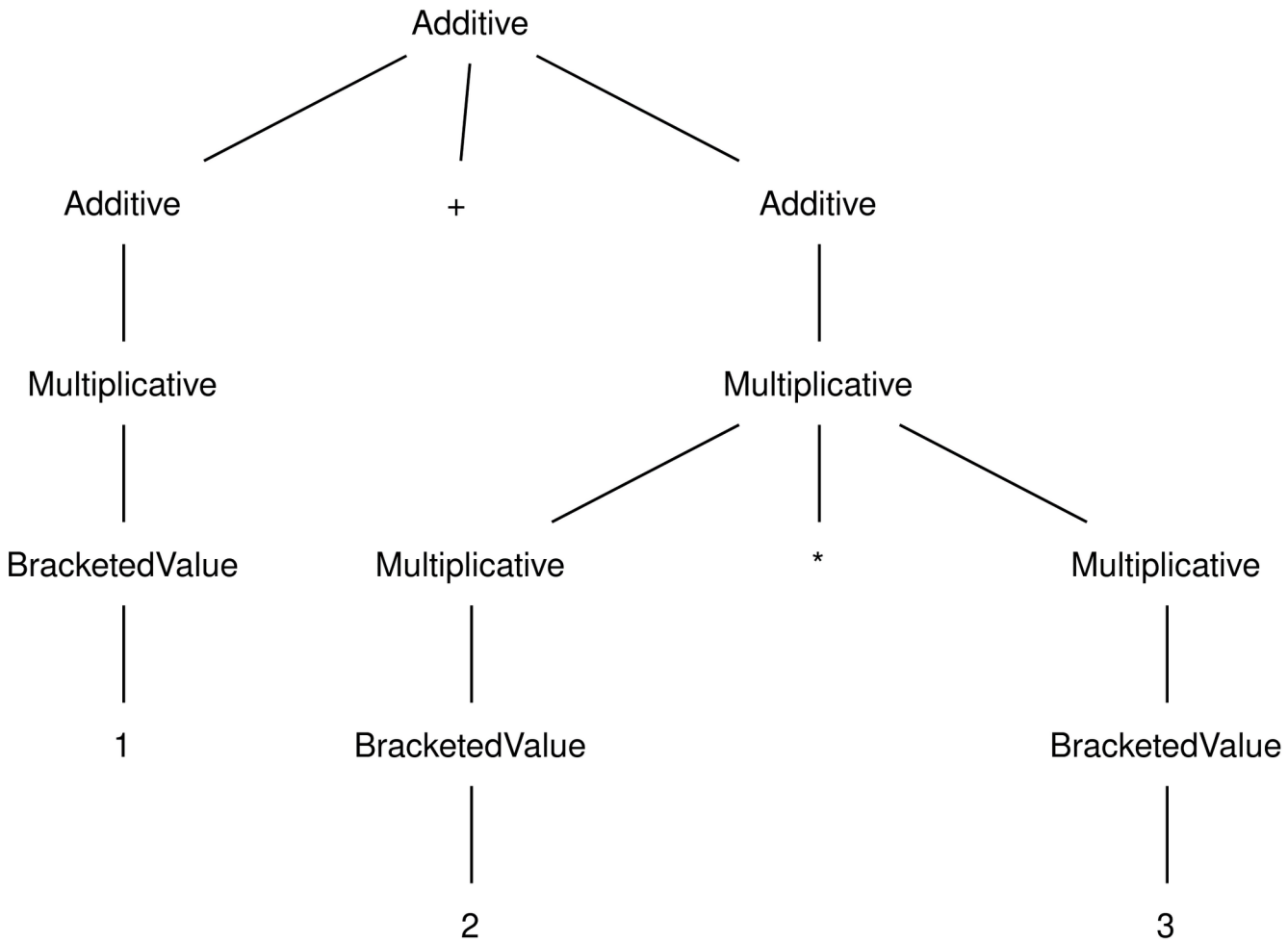
Problem

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX olympics: <http://wallscope.co.uk/ontology/olympics/>
PREFIX medal: <http://wallscope.co.uk/resource/olympics/medal/>
SELECT ?athlete ?athlete_label (COUNT(?medal) as ?count) WHERE {
  ?medal olympics:medal medal:Gold .
  ?medal olympics:athlete ?athlete .
  ?athlete rdfs:label ?athlete_label .
}
GROUP BY ?athlete ?athlete_label
ORDER BY DESC(?count)
```

New Parser

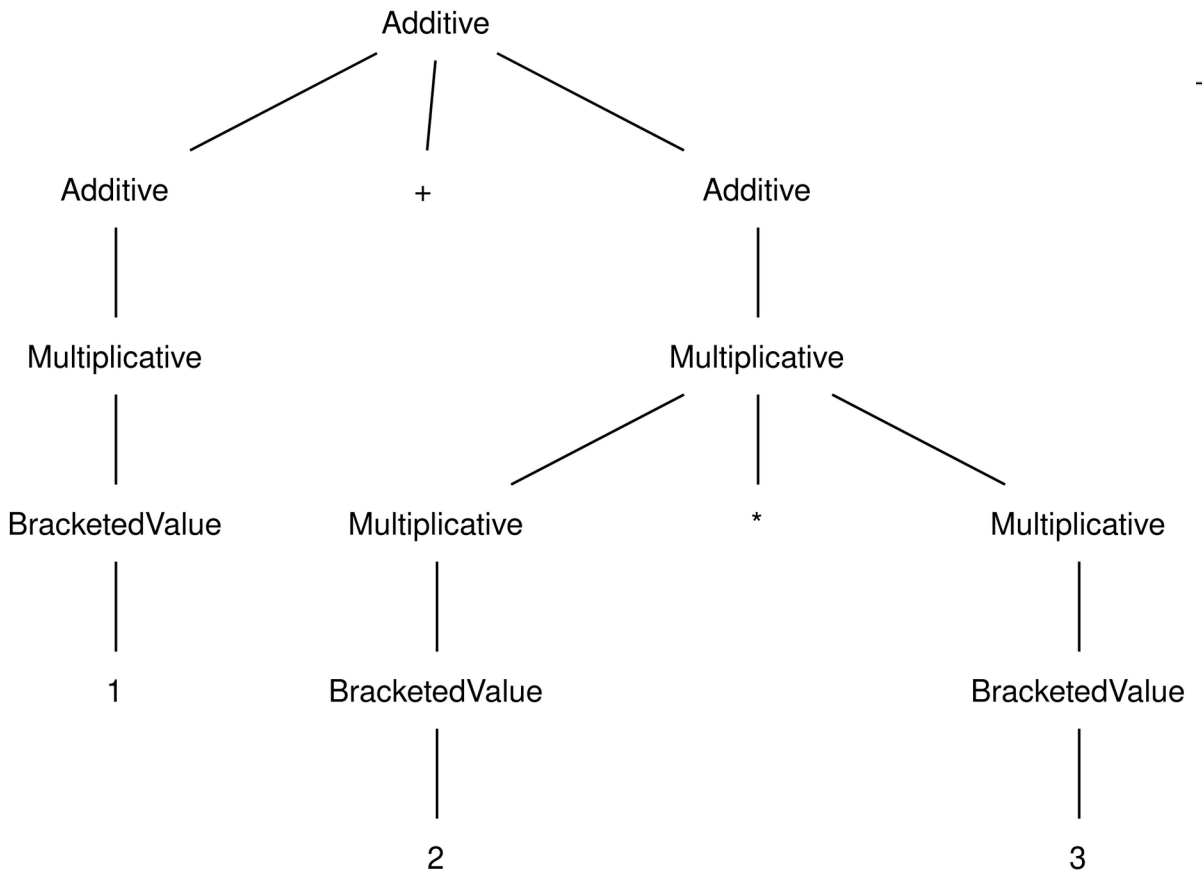

provided by ANTLR runtime library
Lexer
Query string
ANTLRInputStream
SparqlAutomaticLexer
CommonTokenStream
Tokens
generated by ANTLR
Parser
SparqlQleverVisitor
SparqlAutomaticParser
Application specific model
Parse tree

```
1 Additive: Additive "+" Additive | Multiplicative;
2 Multiplicative: Multiplicative "*" Multiplicative | BracketedValue;
3 BracketedValue: ("0".."9")+ | "(" Additive ")";
```



Questions?

```
1 class AdditiveAutomaticVisitor : public antlr4::tree::
     AbstractParseTreeVisitor {
2    antlrcpp::Any visitAdditive(AdditiveContext* context);
3    antlrcpp::Any visitMultiplicative(MultiplicativeContext* context);
4    antlrcpp::Any visitBracketedValue(BracketedValueContext* context);
5 }
```



```
1 class AdditiveAutomaticVisitor : public antlr4::tree::
      AbstractParseTreeVisitor {
2   antlrcpp::Any visitAdditive(AdditiveContext* context);
3   antlrcpp::Any visitMultiplicative(MultiplicativeContext* context);
4   antlrcpp::Any visitBracketedValue(BracketedValueContext* context);
5 }
```

```
1 class AdditiveVisitor {
2   Expression visit(AdditiveContext* context);
3   Expression visit(MultiplicativeContext* context);
4   Expression visit(BracketedValueContext* context);
5 }
```

Contributions

- Finish visitor implementation
- Improvement of the visitor pattern
  - Type safety
  - Less code duplication
- Improved parser tests
- Better error messages

Questions?

Evaluation: Criteria

- Correctness
- Standard compliance
- Resilience against errors
- Maintainability

Evaluation

- Standard compliance
  - Recognizes whole language, actual support varies
- Coverage

Component	Coverage	
	Old Test	New Test
Old Parser	60%	n.a.
New Parser	67%	84%

- Speed
  - Parsing has no significant impact on execution time.

Questions?

Demo Error Messages

- Error: https://qlever.cs.uni-freiburg.de/olympics/m7QTpD?exec=true
- Unsupported Feature: https://qlever.cs.uni-freiburg.de/olympics/9tWCrn?exec=true

```
1  TEST(ParserTest, testParse) {
2       [...]
3       auto pq = SparqlParser("SELECT ?x WHERE {?x ?y ?z}").parse();
4       ASSERT_TRUE(pq.hasSelectClause());
5       const auto& selectClause = pq.selectClause();
6       ASSERT_GT(pq.asString().size(), 0u);
7       ASSERT_EQ(0u, pq._prefixes.size());
8       ASSERT_EQ(1u, selectClause._varsOrAsterisk.getSelectedVariables()
            .size());
9       ASSERT_EQ(1u, pq._rootGraphPattern._children.size());
10      ASSERT_EQ(1u, pq._rootGraphPattern._children[0]
11                      .getBasic()
12                      ._whereClauseTriples.size());
13      [...]
14 }
```

```
1 MATCHER_P3(IsLimitOffset, limit, textLimit, offset, "") {
2   return (arg._limit == limit) && (arg._textLimit == textLimit) &&
3          (arg._offset == offset);
4 }
```

```
1 TEST(SparqlParser, LimitOffsetClause) {
2   string input = "LIMIT 10";
3   ParserAndVisitor p{input};
4
5   auto limitOffset = p.parser.limitOffsetClauses()->accept(&p.visitor).
        as<LimitOffsetClause>();
6   EXPECT_THAT(limitOffset, IsLimitOffset(10, 1, 0));
7 }
```

```
1 auto LimitOffset = [](uint64_t limit, uint64_t textLimit,
2                       uint64_t offset) -> Matcher<const
                           LimitOffsetClause&> {
3   return testing::AllOf(
4       AD_FIELD(LimitOffsetClause, _limit, testing::Eq(limit)),
5       AD_FIELD(LimitOffsetClause, _textLimit, testing::Eq(textLimit)),
6       AD_FIELD(LimitOffsetClause, _offset, testing::Eq(offset)));
7 };
```

```
1  TEST(SparqlParser, LimitOffsetClause) {
2    auto expectLimitOffset = ExpectCompleteParse<&Parser::
         limitOffsetClauses>{};
3    [...]
4    expectLimitOffset("LIMIT 10", m::LimitOffset(10, 1, 0));
5    expectLimitOffset("OFFSET 31 LIMIT 12 TEXTLIMIT 14",
6                      m::LimitOffset(12, 14, 31));
7    [...]
8  }
```

```
1 class AdditiveAutomaticVisitor : public antlr4::tree::
      AbstractParseTreeVisitor {
2   antlrcpp::Any visitAdditive(AdditiveContext* context);
3   antlrcpp::Any visitMultiplicative(MultiplicativeContext* context);
4   antlrcpp::Any visitBracketedValue(BracketedValueContext* context);
5 }
```

```
1 class AdditiveVisitor {
2   Expression visit(AdditiveContext* context);
3   Expression visit(MultiplicativeContext* context);
4   Expression visit(BracketedValueContext* context);
5 }
```