Project – ANTLRv4

Patryk Kiepas

March 25, 2017

1 Instruction

1.1 General

1. Project is for one person.
2. Read this PDF and perform mentioned tasks.
3. Select 1 out of 4 given codes.
4. Build grammar using ANTLRv4 for selected code example.
5. Send the report and grammar by email: `kiepas@agh.edu.pl` with subject: [**EFREI Linguistique**] **Project ANTLRv4 – *Your Full Name***.
6. **Deadline is 24:00, 1st April 2017 (end of Saturday).**
7. **IMPORTANT:** In your solution you might use proposed tokens/rules but it's not necessary.

1.2 Report

In the report list all of these things about the grammar you created:

- A list of used tokens – with a short description for each one
- A list of used parser rules – with a short description for each one
- Describe shortly how the grammar works
- Give two other code examples that your grammar is able to parse (at least 10 lines of code for each example)
- List a few ideas for extending your grammar (at least 2 ideas)

2 JavaScript + jQuery

Proposed constructs:

- Token: RETURN, ELSE, THIS
- Regular: *object*, *function_def*, *stmt*

```
var compare = {                                    // Declare compare object
  name: function(a, b) {
  a = a.replace(/^the /i, '');
  b = b.replace(/^the /i, '');

  if (a < b) {
    return -1;
  } else {
    return a > b ? 1 : 0;
  }
},

$('.sortable').each(function() {
  var $controls = $table.find('th');
  var rows = $tbody.find('tr').toArray();

  $controls.on('click', function() {  // When user clicks on a header
    var $header = $(this);
    var order = $header.data('sort');

    if ($header.is('.ascending') || $header.is('.descending')) {
      $header.toggleClass('ascending␣descending');
    } else {
      $header.siblings().removeClass('ascending␣descending');
      if (compare.hasOwnProperty(order)) {
        column = $controls.index(this);
        rows.sort(function(a, b) {
          a = $(a).find('td').eq(column).text();
          b = $(b).find('td').eq(column).text();
          return compare[order](a, b);
        });
        $tbody.append(rows);
      }
    }
  });
});
```

3 Scala

Proposed constructs:

- Tokens: DEF, OBJECT, EXTENDS
- Rules: *object*, *class_def*, *function_def*

```
object complexOps extends Application {
    class Complex(val re: Double, val im: Double) {
        def + (that: Complex) =
            new Complex(re + that.re, im + that.im)

        def - (that: Complex) =
            new Complex(re - that.re, im - that.im)

        def * (that: Complex) =
            new Complex(re * that.re - im * that.im,
                        re * that.im + im * that.re)

        def / (that: Complex) = {
            val denom = that.re * that.re + that.im * that.im
            new Complex((re * that.re + im * that.im) / denom,
                        (im * that.re - re * that.im) / denom)
        }

        override def toString =
            re + (if (im < 0) "-" + (-im) else "+" + im) + "*i"
    }
    val x = new Complex(2, 1); val y = new Complex(1, 3)
    println(x + y)
}
```

4 C++

Proposed constructs:

- Token rules: *COMMENT*, *INCLUDE*, *INT*
- Grammar rules: *include*, *expr*, *stmt*

```
#include <iostream>
#include <vector>
#include <stdexcept>

int main() {
    try {
        std::vector<int> vec{3,4,3,1};
        // Throws an exception, std::out_of_range
        // (indexing for vec is from 0-3 not 1-4)
        int i{vec.at(4)};
    }

    // An exception handler, catches std::out_of_range
    catch (std::out_of_range& e) {
        std::cerr << "Accessing␣a␣non-existent␣element:␣" << e.what() << "\n";
    }

    catch (std::exception& e) {
        std::cerr << "Exception␣thrown:␣" << e.what() << "\n";
    }

    catch (...) {
        std::cerr << "Some␣fatal␣error\n";
    }
}
```

5 Python

Proposed constructs:

- Tokens: IMPORT, RETURN, DEF
- Rules: *import*, *function_def*, *assignment*

```
from scipy import ndimage,special,fftpack
import numpy

def filter2B(image, mask):
    (Mx, My) = mask.shape
    # check if Mrow and Mcol are odd
    if Mx % 2 == 0 or My % 2 == 0:
        print "Mask␣width␣and␣height␣must␣be␣odd"
        return array([])
    Nx = (Mx - 1) / 2
    Ny = (My - 1) / 2
    filtered_image = ndimage.filters.convolve(image, mask, mode='nearest')
    return filtered_image

def approxI1_I0(image):
    count = numpy.sum(image < 1.5)
    image_oct = 8 * image
    Mn = 1. - 3. / image_oct - 15. / 2. / (image_oct ** 2) -
            (3. * 5. * 21.) / 6. / (image_oct ** 3)
    Md = 1. + 1. / image_oct + 9. / 2. / (image_oct ** 2) +
            (25. * 9.) / 6. / (image_oct ** 3)
M = Mn / Md
if (count > 1):
     K = numpy.flatnonzero(image < 1.5)
     K = numpy.flatnonzero(image == 0)
return M
```