Compilers

ANTLR overview

2nd Bachelor Computer Science
2023 – 2024

Kasper Engelen
kasper.engelen@uantwerpen.be

1 Introduction

ANTLR is a tool that can be used to automatically generate code from a context-free grammar. This code will include a **parser**, as well as so-called **visitor** classes that can be used to extract information from the parse tree. In this document you will find instructions on how to install ANTLR and how to use it in your compiler project. Note that in this project we will use Python 3, and that you will need to install the Java runtime as well.

2 Installing ANTLR

As far as this course is concerned we will use two tools that are provided by the ANTLR project: the ANTLR tool itself and the ANTLR runtime library. The first is a Java jar-file that can be used out-of-the-box. Note that this requires Java in order to run. The ANTLR runtime library is a Python library that can be used with a package manager such as pip, virtualenv, Anaconda, etc.

The latest version of the ANTLR tool (for example: `antlr-4.12.0-complete.jar`) can be downloaded from `https://www.antlr.org/download`. The ANTLR runtime can be installed using, in the case of Pip, the following command:

```
pip install antlr4-python3-runtime
```

Make sure you do not have a folder called `antlr4` in your code, since this will prevent the library from being successfully imported in Python.

3 Using ANTLR

Making use of ANTLR inside a project can be done by following a number of steps:

1. Create a grammar file (`.g4` extension). This file contains the grammar of the language you want to parse. Multiple examples of grammar can be found on the internet. **You will have to write your own grammar as part of the assignment!**

    - **Tip:** Create some text files that contain examples of the language described by the grammar. These will be useful when debugging your grammar.

    - **Tip:** In order to easily see if your grammar works correctly, go to `http://lab.antlr.org`. There you can paste your grammar, set a start rule, and paste some example code. You will then see a visualised parse tree, as well as any parsing errors that may occur. **Make sure that there is no code present under the "Lexer" tab!**

- **Tip:** Write your grammar in such a way that it is structured and flexible. The grammar will be built incrementally so the structure has to be easy to modify later on. Working in a well-structured manner will also reduce the amount of bugs in your grammar and make it easier to debug.

2. Generate the parser and visitor classes in Python using the following command:

```
java -jar antlr-4.12.0-complete.jar -Dlanguage=Python3 MyGrammar.g4
-visitor
```

   - This will generate Python classes that you can use to parse files written according to your specific grammar. **Do not edit these files, as they will be overwritten everytime you change your grammar.**
   - In the example, `MyGrammar.g4` is the text file containing your grammar. Using the `-Dlanguage` flag, the target language can be chosen, in this case Python 3. Using the `-visitor` flag, a default parse tree visitor base class is generated.

3. You will need a `main.py` that will handle the input and call the parser. You can find an example here: `https://github.com/antlr/antlr4/blob/master/doc/python-target.md`
   - In this file you will have to import the ANTLR-generated files and classes, as well as the ANTLR runtime (`import antlr4`). Make sure you do not have a folder called `antlr4` among your code files. The ANTLR runtime is necessary since the generated ANTLR classes depend on this library.

4. In order to analyse the contents of the parse tree, you will have to implement custom visitor classes. Your custom visitor class can then be used to traverse the parse tree and analyse its contents.
   - **Note:** You will need to create a subclass to the generated visitor class. The reason we use inheritance is to avoid losing your code when generating the ANTLR classes again. Write such classes in your own files.

4 Documentation and debugging

When implementing and debugging your grammar and visitor class, the following references may be useful:

- Don't hesitate the ask the teaching assistant for help.
  You can also e-mail to `kasper.engelen@uantwerpen.be`.
- Visit `http://lab.antlr.org` where you can try out your grammar. **Make sure that there is no code present under the "Lexer" tab!**
- Some information on how to call your parser from Python can be found here: `https://github.com/antlr/antlr4/blob/master/doc/python-target.md`
- Documentation for the ANTLR API can be found at `https://www.antlr3.org/api/Python/index.html`
- For more information on the API, you can use the documentation for the Java API: `http://www.antlr.org/api/Java/index.html`.
- Use the Python built-in `dir()` function with ANTLR objects as parameter to find out what fields and methods are available.