

Formal Languages

Task Instructions

Goal of the task

- Practice building and running Regular Expressions.
- Practice building parsers.

1 Organization and management

All practical assessment tasks will be carried out in teams of four members. Exceptionally, we may allow a team of three. Teams will remain unchanged throughout the duration of the course.

2 Instructions

1. Download the Java project template from Moodle.
2. Study its structure based on what we have learned in class. It is a simple interpreter/parser based on antlr4.
3. Implement the logic of the class `RegularExpression`. The `RegularExpression` class already comes with a signature, all you have to do is to implement its methods and constructors, which are currently empty. The logic of the `RegularExpression` class is given in comments. You can also check what it is expected from the class by looking at how it is used by the parser and by the main method in the `App` class.
4. Check that your implementation works by running the test at `/test/java/Test.java`. The test loads a specification file at `/test/resources/testcases/re1.txt`, build a `RegularExpression` based on that specification, iterate over all strings in the input file at `/test/resources/testcases/input1.txt`, and check whether the output matches that given in the file `src/test/resources/testcases/`

5. In addition to the `Test` class, we have provided an `App` class for you to run the parser. Both can be used for testing purposes. The `App` class is a simple interface that, given a `RegularExpression` specification and an input file, outputs a binary array indicating which strings were accepted.
6. You can add more specification files, test cases, etc. We recommend you check your implementation thoroughly.
7. The next step is for each team member to provide a `RegularExpression` specification. Consider a team formed by Alice, Bob and Charley. Take any order for the team members, for example, Alice is team member 1, Bob is team member 2, and Charley is team member 3. Team member $\{i\}$ will provide his/her specification within the file `test/resources/testcases/re-member $\{i\}$.txt`, where $\{i\}$ is a place holder for 1, 2, 3, Each specification has instructions on the language that the Regular Expression should recognize. It is important that you, i.e., team member i , add your name at the end of the file. For example, Alice would be in charge of completing the `RegularExpression` specification at `test/resources/testcases/re-member1.txt` and, after doing so, she should add her name to the file within a comment.
8. Feel free to test your specifications thoroughly.
9. The submission consists of a single zip file that contains the following files:
 - (a) `DFA.java`, which was already provided in the previous submission.
 - (b) `NFA.java`, which was already provided in the previous submission.
 - (c) `RegularExpression.java`
 - (d) `re-member1.txt`, `re-member2.txt`, etc.
 - (e) A `Readme.md` file with the following information:
 - Title
 - Project overview
 - A statement on the contribution of each member.

3 Evaluation

The task either *passes* or *fails*. The task passes if all test cases ran by the marker passes, otherwise it fails.