Formal Languages

Task Instructions

Goal of the task

- Practice building and running DFAs.
- Practice building parsers.

1 Organization and management

All practical assessment tasks will be carried out in teams of three members. Exceptionally, we may allow a team of two. 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 DFA. The DFA 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 DFA 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. For simplicity, we shall specify DFAs without the requirement that the transition function is total. In other words, we will let the programmer specify DFAs with *missing* transitions, then assume those transitions all go to a trap state.

5. Check that your implementation works by running the test at `/test/java/Test.java`. The test loads a specification file at `/test/resources/testcases/dfa1.txt`, build a DFA 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/output1.`

6. 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 DFA specification and an input file, outputs a binary array indicating which strings were accepted.

7. You can add more specification files, test cases, etc. We recommend you check your implementation thoroughly.

8. The next step is for each team member to provide a DFA 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.

   (a) Team member $\{i\}$ will provide his/her specification within the file test/resources/testcases/dfa-member$\{i\}$.txt, where $\{i\}$ is a place holder for $1, 2, 3, ...$

   (b) Each specification file has instructions on the language that the DFA 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 DFA specification at `test/resources/testcases/dfa-member1.txt` and, after doing so, she should add her name to the file within a comment.

9. Feel free to test your specifications thoroughly.

10. The submission consists of a single zip file that contains the following files:

    (a) DFA.java

    (b) dfa-member1.txt, dfa-member2.txt, etc.

    (c) 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. All test cases contain valid inputs. This means you don't need to worry about input validation, as no erroneous input will be given.