Prof. Dr. Christoph Bockisch
MSc Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Companion exercises
Objektorientierte Programmierung: Wintersemester 2021/2022

No. 13, due until 21.02.2022

Attention: This exercise-sheet is pure bonus. Points on this sheet count towards your credit and not your debit.

Task 13.1: Nier: Replicant — 6 Points

Like a DFA (Deterministic Finite Automaton), a NFA (Non-determinstic Finite Automaton) can be described by using the following quintupel: (Q, Σ, δ, q_0, F).

The meaning of these symbols is basically the same as it was with the DFA. The only difference is in the transitionfunction δ. A quick reminder: $\delta : Q \times \Sigma \to Q$ was the definition of $delta$ within a deterministic context. The non-deterministic definition of $delta$ is as follows: $\delta : Q \times \Sigma \to 2^Q$. This means that δ can return a set of states instead of only one state. The empty set is also a possible result. The NFA is in an accepting state if one or more states within the resulting set is an accepting state, so $\exists q_n \in Q^* | q_n \in$ F.

You will need your solutions to the previous exercise sheet 10. If you did not solve that particular sheet, you may download *dfa.zip* from Ilias with needed classes packaged inside.

Attention: You may ignore ϵ-transitions and may assume that only one start-state is valid.

a) Implement a class `NFA` that extends `GenericAutomaton`. Implement a useful constructor and a field for the set of current states. — 1
b) Implement a method to add transitions to your `NFA` by using methods of the super-class. — 1
c) Implement a method `public ArrayList<State> delta(Character symbol)` that returns a new set of current states based on each of the current states and overrides the current states set within your automaton. — 2
d) Implement the required methods from `GenericAutomaton`. — 1
e) Test your implementation of `NFA` using JUnit-Tests. — 1