Homework for lecture
Automata and Formal Languages II

TU München
Institut für Informatik
Dr. Peter Lammich

SS 2015 | Homework Sheet 3 | 21.04.2014

Submission: May 6

Aufgabe 3.1. [**Linear Time Emptiness Check**] (10 points)
Given an NFTA $\mathcal{A}$. Specify and prove correct an $O(|\mathcal{A}|)$-time algorithm to decide $L(\mathcal{A}) = \emptyset$. See TATA, Ex 1.18, for additional hints.

Aufgabe 3.2. [**Application: Executions of Parallel Programs**] (10 points)
Consider the alphabet $fork/2, write/1, other/1$, $lock/1$, $unlock/1$, $nil/0$. Intuitively, a tree describes an execution of parallel processes that write to a resource. $fork$ is a step that creates a new process, *write* accesses the resource, *other* abstracts from other steps, e.g., modification of local variable. *lock* and *unlock* describe locking and unlocking of the resource. *nil* is used to indicate the end of the steps for a process.

1. Specify a tree automaton that captures the executions of the following program, for all numbers of iterations of the while loop.

```
proc1:
  local variable x;
  x = 5;
  while * do {
    lock;
    write x;
    unlock;
  }

main:
  local variable y;
  fork proc1;
  y = 7;
  lock
  write y;
  unlock;
```

2. Specify a tree homomorphism that deletes *other*-nodes, preserving the remaining structure of the tree.
3. Specify a tree automaton that characterizes executions where locks are not used reentrantly. I.e., if a process has already locked the resource, it must not execute *lock* again, before it has unlocked the resource.
4. Specify a tree automaton that characterizes executions that contain *write* operations not protected by a lock

For 3 and 4, it is enough to characterize executions without *other*-nodes.