Formal Methods, 2013/14

Homework 2

5th of December 2013

Due date: 12th of December 2013, beginning of the lecture

Problem 1 (3 Points)
Consider the two non-deterministic automata below. For these two automata give a description in English which languages they describe and give a deterministic version of the automata according to the construction from class:

a) [Diagram of the first automaton]

b) [Diagram of the second automaton]

For this automata construct a deterministic version according to the construction from class:

c) [Diagram of the third automaton]

Problem 2 (2 Points)
Answer these questions with yes or no. If no give a counter-example, if yes give an explanation. A regular language is a language that is recognised by an automaton. So the language of palindromes is not regular. As an alphabet you may assume \{a, b\}.

a) Is the universal language regular? The universal language is the language that contains all possible words that can be made of the alphabet. If yes, show an automaton for the alphabet \{a, b\}.
b) Let $L$ be a language such that $|L| = n$ for some fixed $n \in \mathbb{N}$. Is every such $L$ regular?

c) Let $L$ be a regular language. Does there exist a strict subset of $L$, $L'$, which is regular? Is every such subset $L'$ regular?

**Problem 3 (5 Points)**

See below for a semantics of reactive modules.

We want to model a train passing through a gate. The gate must obviously be closed while the train passes through.

- If the train sends approach it takes between 2 and 3 seconds until the train reaches the gate.
- The train stays in the gate at most 2 seconds
- If the gate receives approach it closes within 2 seconds
- If the gate receives exit it opens within 1 second.

a) Give a set of reactive modules (for time, train and gate) that model the above specification. Keep in mind that within $x$ seconds means that every tick up to $x$ there is a choice that the action occurs now or not (once $x$ is reached there is no choice, the action has to occur).

b) Give an automaton that shows all possible transitions through the states of the reactive modules. The edges should be labeled with the event(s) that occur during this transition(s) if any, or otherwise be empty.
Semantics of Reactive Modules

We define the language of Simple Reactive Modules (SRM).

**Syntax** A *module* consists of
- a set $A$ of input events,
- a set $B$ of output events,
- a set $X$ of state variables,
- an initial condition on the state variables,
- a guarded command for each input event,
- a guarded command for each output event,
- a complete guarded command called update command.

A *guarded command* is a set of guarded actions. A *guarded action* consists of a guard, which is an enabling condition on the state variables, and an action, which is a modification of the state variables. A guarded command is *complete* if in every state at least one guard is true. A *system* is a set of modules whose output events are disjoint.

**Semantics** A *state for a system* consists of a state for every module. A *state for a module* is a function that maps every state variable to a value. A guarded command $g$ changes module state $q$ to module state $q'$ if there exists a guarded action in $g$ whose guard is true in $q$ and whose action modifies $q$ to $q'$. Module state $q'$ is an *$e$-successor* of module state $q$ if $e$ is an input or output event and the guarded command for $e$ changes $q$ to $q'$. Module state $q'$ is an *update successor* of module state $q$ if the update command changes $q$ to $q'$. System state $s'$ is an *update successor* of system state $s$ if every module state in $s'$ is an update successor of the respective module state in $s$. System state $s'$ is an *$e$-successor* of system state $s$ if for every module $M$, if $e$ is an input or output event of $M$, then the $M$-state in $s'$ is an $e$-successor of the $M$-state in $s$, else the $M$-state in $s'$ is an update successor of the $M$-state in $s$. 