Ruzica Piskac, Ph.D.
Leander Tentrup, M.Sc.
Michael Gerke, M.Sc.
Felix Klein, B.Sc.

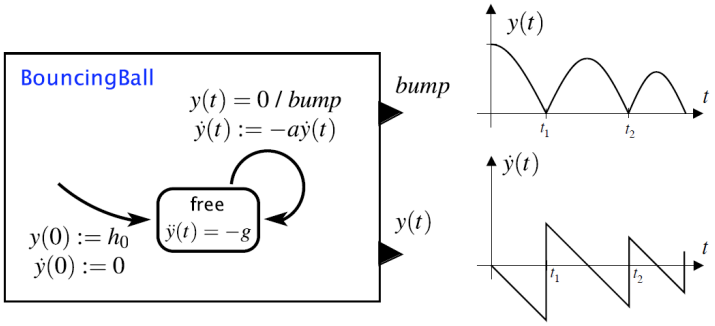
Summer term 2014
Problem Set 2
Due: Tuesday, 20th May 2014

Embedded Systems

Please indicate your **name**, **matr. number**, **email address**, and which **discussion session** you have been allocated to. Only one submission per group is necessary.

Problem 1: Bouncing Ball

The following FSM from the book "Introduction to Embedded Systems" models a bouncing ball. It has a number of limitations and even a serious flaw in the sense that it allows for clearly unintended behavior. Discuss and repair.



Problem 2: Geiger-Müller Counter

You are supposed to model the counter unit for a Geiger-Müller counter that counts how many *events* occur during one second as an extended FSM. An event is a discharge of the device due to ionizing radiation. As can be seen in the drawing below, the input to the counter is a voltage level. Whenever an event is detected there is a significant peak in the measured voltage level. Peaks over a given threshold V_{min} should be counted by the counter. After each second the display should be updated to show the number of events that occurred during the last second.

If you are not familiar with the working principle of Geiger-Müller counters you might want to read the Wikipedia article.[1]

[1] http://en.wikipedia.org/wiki/Geiger-Müller_tube

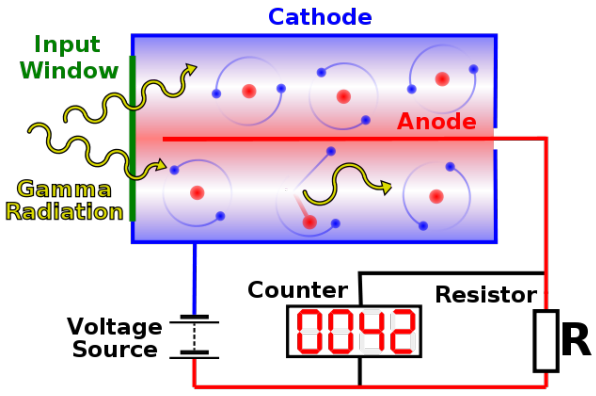


Figure 1: Schematic drawing of a Geiger-Müller counter. Source: Wikipedia.

Problem 3: Modeling with StateCharts

Figure 2 shows the control of a simple vending machine (in the StateCharts formalism). Figure 3 lists all occurring events together with their meaning. A typical interaction of the vending machine with the environment is:

- Initially the system is in the states $\boxed{0}$ and $\boxed{A}$.
- The user inserts a coin, the environment generates the event COIN_IN, A_1 moves to state $\boxed{1}$, and the event OK is generated.
- A_2 consumes the event OK and moves to state $\boxed{B}$.
- The user presses the cancel-button, A_1 moves back to state $\boxed{0}$, the events RESET and COIN_OUT are generated.
- A_2 consumes the RESET event and moves back to state $\boxed{A}$.

(a) Describe the trace of transitions occurring when the user inserts a coin and orders coffee.

(b) The control of the vending machine has a bug that allows the user to cheat. Find it.

(c) Fix the bug.

(d) Now, construct an equivalent automaton Q where no parallelism is involved. The initial state should be $\boxed{0A}$. When the event COIN_IN occurs, Q moves to state $\boxed{1A}$ and the event OK is generated. This causes Q to move from state $\boxed{1A}$ to state $\boxed{1B}$. Now continue yourself.

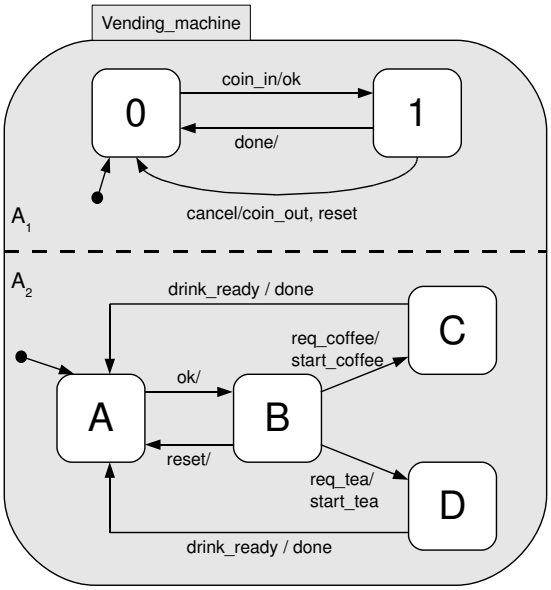


Figure 2: A vending machine.

Event	**Generated by**	**Consumed by**	**Meaning**
COIN_IN	environment	A_1	user inserts coin
CANCEL	environment	A_1	user presses cancel-button
REQ_COFFEE	environment	A_2	user presses coffee-button
REQ_TEA	environment	A_2	user presses tea-button
DRINK_READY	environment	A_2	drink is ready
COIN_OUT	A_1	environment	coin returned to user
START_COFFEE	A_2	environment	start preparation of coffee
START_TEA	A_2	environment	start preparation of tea
OK	A_1	A_2	enough coins inserted
RESET	A_1	A_2	coins back to user
DONE	A_2	A_1	drink delivered

Figure 3: Events for the vending machine in Figure 2.

Problem 4: Timed StateCharts

Consider the AND-state in Figure 4 that models a system with two concurrent processes `P1` and `P2` accessing a shared resource. The StateChart comprises the global variable `id` that is initially set to 0, and the external events `try1`, `try2`, `set1`, `set2`, `retry1`, `retry2`, `enter1`, `enter2`, `exit1`, and `exit2`. Furthermore, the timing behavior is parameterized by the integer constants `D` and `T`. The system is considered as *safe* if it is never the case that `P1` is in `crit1` and `P2` is in `crit2` at the same time.

In case of conflicting update assignments we assume an *interleaving semantics*. For instance, if the assignments id:=1 and id:=2 are to be executed concurrently, then the semantics nondeterministically determines whether (1) first id:=1 then id:=2 is executed, or (2) first id:=2 then id:=1 is executed.

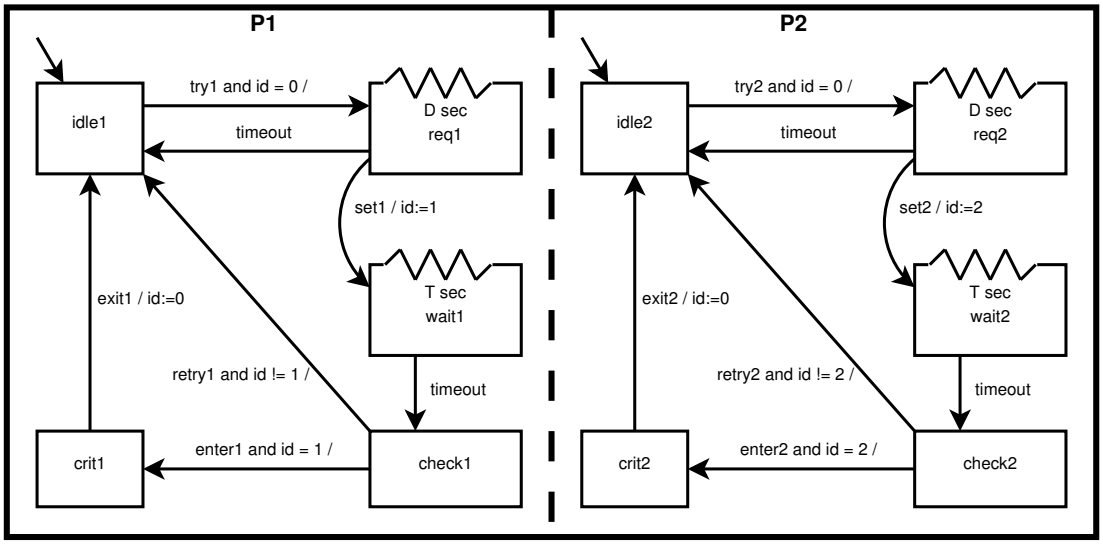


Figure 4: A timed mutual exclusion protocol.

(a) Assume `D` = 10 and `T` = 5. Is the system safe? Justify your answer either by giving an argument why it is safe, or by providing a (short) trace (i.e., a scenario of delays and events) leading to an unsafe state, where `P1` is in `crit1` and `P2` is in `crit2` at the same time.

(b) Give the most general characterization how `D` and `T` must be chosen such that for *any* occurring of events, the system will always remain safe.