Prof. Dr. Andreas Podelski
Tanja Schindler

Hand in until November 14th, 2018
15:59 via the post boxes
Discussion: November 19/20th, 2018

Tutorial for Cyber-Physical Systems - Discrete Models

Exercise Sheet 4

The goal of this sheet is to train your ability to construct a transition system that describes the behaviour of a parallel system where the single components communicate via shared variables, in order to be able to reason about such a parallel system.

Exercise 1: Parallelism - Interleaving I

The goal of this exercise is to train your ability to construct the interleaving of program graphs and the underlying transition system in order to be able to reason about parallel systems.

Consider the following parallel system consisting of two processes. The programs for each process are given in a high-level programming language. Each line of the pseudocode reflects an atomic expression.

Process 1:	**Process 2:**
1 $x := x + 1;$	1 $x := 3 * x;$

(a) Draw the program graphs P_1 and P_2 for each process with $Var_1 = Var_2 = \{x\}$. The domain of x is $\{-128, -127, \ldots, -1, 0, 1, \ldots, 127\}$ (signed 8-bit integer). We assume that the initial value of x is 1.

(b) Provide the effect functions Effect_1 and Effect_2.

(c) Draw the interleaving of the program graphs $P_1 ||| P_2$ and provide the effect function.

(d) Draw the reachable part of the transition system $TS(P_1 ||| P_2)$. Use it to determine which values can finally be stored in x.

Exercise 2: Parallelism - Interleaving II

The goal of this exercise is to get an Aha! experience.

Consider the following parallel system consisting of two processes. The programs for each process are given in a low-level programming language. The parallel system seems to do semantically the same as the parallel system in Exercise 1.

Process 1:	**Process 2:**
1 $r_1 := x;$	1 $r_2 := x;$
2 $r_1 := r_1 + 1;$	2 $r_2 := 3 * r_2;$
3 $x := r_1;$	3 $x := r_2;$

(a) Draw the program graphs P_1 and P_2 for each process with $Var_1 = \{x, r_1\}$ and $Var_2 = \{x, r_2\}$. The domain of x, r_1 and r_2 is $\{-128, -127, \ldots, -1, 0, 1, \ldots, 127\}$ (signed 8-bit integer). We assume that the initial value of x is 1 and the initial values of r_1 and r_2 are both 0.

(b) Provide the effect functions Effect_1 and Effect_2.

(c) Draw the interleaving of the program graphs $P_1|||P_2$ and provide the effect function.

(d) Draw the reachable part of the transition system $TS(P_1|||P_2)$. Use it to determine which values can finally be stored in x.

It is not strictly needed for solving the exercise above, but, for those who are interested in the motivation for the two exercises: the above program can be seen as a model for an assembler program, namely the assembler program to which the program in Exercise 1 is compiled.
The two exercises together serve to explain that one cannot assume that an update statement in a high-level language is an atomic action. In fact, in modern multi-core architectures, the system is even more complicated (because you have local registers).

Process 1:	**Process 2:**
1 `LOAD` x;	1 `LOAD` x;
2 `ADDI` 1;	2 `MULTI` 3;
3 `STORE` x;	3 `STORE` x;

The domain of x, ACC_1 and ACC_2 is $\{-128, -127, \ldots, -1, 0, 1, \ldots, 127\}$ (signed 8-bit integer).
Semantics of the assembler commands:

- `LOAD i` : Load the content of address i of the memory into an accumulator register ACC.
- `ADDI i` : Adds i to the content of register ACC and stores the result in ACC.
- `MULTI i` : Multiplies the content of register ACC with i and stores the result in ACC.
- `STORE i` : Store the content of register ACC at memory address i.

Note: A variable is here represented by a memory address which is a typical abstraction.