An Actor-oriented Approach to ARGESIM Benchmark C05 'Two State Model' using Berkeley Ptolemy II

Patrick Einzinger

dwh Simulation Services, Neustiftgasse 57-59, 1070 Vienna, Austria; patrick.einzinger@drahtwarenhandlung,at

Simulator. The Ptolemy II project (from the Center for Hybrid and Embedded Software Systems in the Department of Electrical Engineering and Computer Sciences of the University of California in Berkeley) studies modelling and simulation of concurrent, real-time, embedded systems which combine the continuous dynamics of physical systems with discrete mode changes (for example from a digital controller). Models in Ptolemy II consist of so-called actors - visually represented in block diagrams - which are able to communicate with each other by sending messages over ports and channels.

Actors and their connections describe an abstract syntax of the model; additionally one has to specify a 'model of computation' (like continuous time or discrete event) which gives operational rules for the execution of the model. Ptolemy II supports heterogeneous and hierarchical mixtures of different models of computation. It is constructed in Java and the source code is freely available (*http://ptolemy.eecs.berkeley.edu/*).

Modelling. The top-level of the model (Figure 1) consists of one 'Modal Model' actor called 'TwoStateModel' with output ports for y1, y2 and state changes (the icon for the modal model hides the port for y2). A modal model implements a finite state machine whose states (or modes) can have refinements, i.e. submodels that execute when the state machine is in the corresponding state. Therefore this actor type is clearly well suited for the Two State Model.

The modal model actor has connections to various sinks: a timed plotter for plotting the output signal y1, a periodic sampler (period 5) with a display to show the final value for y1 on screen and an expression writer with a clock (triggered by the output port for state changes) for writing the time values of the state events to a file. With these sinks all necessary information for accomplishment of the tasks is available.

A 'director' block defines the model of computation. In the top-level a continuous director governs the execution. It implements a differential equation solver, however as there is no differential equation at the top-level just the continuous directors in the refinements of the states control its step size.

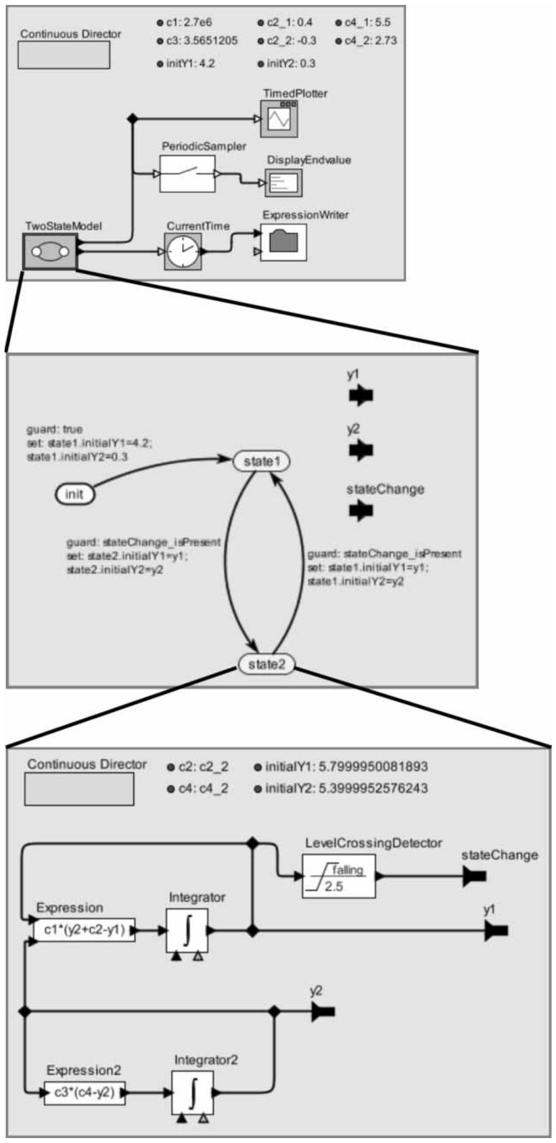

Figure 1. Three hierarchical levels of the model: The top level with the modal model actor 'TwoStateModel' inside of which there is a finite state machine with 'init' state for initialization and the two states of the system. Each state has a refinement. The refinement for state 2 is shown here. Note the 'LevelCrossingDetector' actor which triggers the event of hitting the pin.

Inside the modal model actor 'TwoStateModel' is a finite state machine (therefore it implicitly has an FSM director) with an initial state 'init' used for initializing the model with the corrector initial values for y1 and y2 and the two states 'state1' and 'state2'. The port 'stateChange' triggers the transitions between the states, as it receives an output token each time y1 crosses the specific thresholds. At the time of a state change the output ports for y1 and y2 give the new initial values of the state the model transits to.

The refinements of both state1 and state2 implement the differential equations. The block structure is similar to well-known block diagrams (as for example in Simulink) and has two integrator actors, one for y1 and one for y2. Both are connected to the corresponding output ports. A 'LevelCrossingDetector' actor detects the crossing of the threshold (in rising direction in the case of state1 and in falling direction in state2). At a crossing it sends a token to the output port 'stateChange', activating the guard of the state transition. Both refinements have a continuous director and get the values of c2 and c4 from the corresponding parameters defined at top-level.

A-Task: Simulation Time Domain. The version of Ptolemy II used does not include a solver appropriate for stiff systems. The available solvers are explicit RK23 and explicit RK45. The simulation with RK45 was inefficient: Even with an error tolerance (for the solver and for the level crossing detectors) of 10^{-6} it took several hours (on a laptop with Intel Core 2 Duo T8300, 2GB of Ram and Windows Vista), because the solver had to choose very small step sizes. Figure 2 shows the plot for y1 with error tolerance set to 10^{-10}.

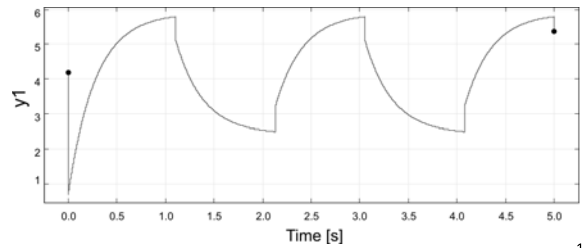

Figure 2. Plot of y1 over time, all error tolerances set to 10^{-10}

B & C-Tasks: Event Times and Final Values. Table 1 shows the time values of the state transitions for all error tolerances set to 10^{-6}, 10^{-10} and 10^{-14}. With error tolerance 10^{-6} an additional sixth event takes place and the final value of y1 is completely wrong. With more stringent error tolerances all events and the final value are detected nearly exactly.

tol	**1.E-06**	**1.E-10**	**1.E-14**
t_1	1.1082628547607	1.1083061678763	1.1083061677712
t_2	2.091871335981	2.1296853555014	2.1296853551551
t_3	2.7319550555404	3.0541529075528	3.0541529069963
t_4	3.2977498326525	4.0755320954016	4.0755320943802
t_5	4.1092310397903	4.9999996474117	4.9999996462213
t_6	4.9718927771635		
y_f	3.5041594536956	5.3701798572355	5.3693128433754

Table 1. Time values of the state events with error tolerances set to 1.E-6, 1.E-10 and 1.E-14, final values y_f .

D-Task: Frequent Events. With new parameter values, the system exhibits high frequent oscillatory behaviour after the first state transition from state1 to state2.

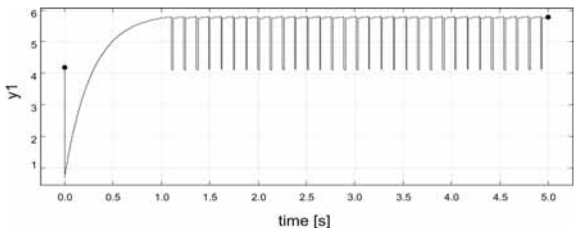

Figure 3. Plot of y1 over time for the parameter values of task d (error tolerances set to 10^{-11}).

t_1 =1.1083061677768	t_{61} = 4.9230401079777
t_2 = 1.1217299678991	t_{62} = 4.9364639081008
t_{60} = 4.8093061100968	y_f = 5.7804025201735

Table 1. First two and last three time values of the state events and final state, error tolerances; Task d.

Altogether there are 62 state events. Figure 3 shows the time plot of y1, and Table 2 gives the time values of the first and last discontinuities as well as the final value for an error tolerance of 10^{-11}.

Resume. The philosophy of the Ptolemy Project is appealing, as it does not express heterogeneous models in terms of one general model of computation. Instead of that it uses an abstract semantics which every actor has to fulfil, and a director block in each actor implements the concrete rules of execution. This allows a rigorous coupling in hybrid models.

Furthermore Ptolemy II is an open source software and freely available. Therefore it is possible to study the internals of the simulator and to do research on hybrid systems without expensive software. Still missing are advanced ODE or DAE solvers (2011).

Submitted: August 2011
Revised: October 2011
Accepted: February 1, 2012