Proceedings of the American Control Conference
Arlington, VA June 25-27, 2001

Modeling Distributed Hybrid Systems in Ptolemy II

Jie Liu, Xiaojun Liu, and Edward A. Lee
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720, USA
{liuj, liuxj, eal}@eecs.berkeley.edu

1. Introduction

We present Ptolemy II as a modeling and simulation environment for distributed hybrid systems. In Ptolemy II, a distributed hybrid system is specified as a hierarchy of models: an event-based top level and distributed islands of hybrid systems. Each hybrid system is in turn a hierarchy of continuous-time models and finite state machines.

Ptolemy II [2] supports the modeling of heterogeneous systems by using a hierarchical component-based architecture and well-defined models of computation. The basic component in Ptolemy II is called an *actor*. Actors have *ports*, which are their communication interface to other actors. An aggregation of actors and their connections is called a *composite actor*. A composite actor can have it own ports, which mediate the communication between inside actors and outside actors. Note that a composite actor is also an actor, so that components can be arbitrarily nested.

The interaction among actors within a composite actor is defined by a model of computation and implemented by a *director* of that composite actor. In particular, a director defines the communication mechanism and the execution order of the actors under its control. The hierarchical composition of actors allows directors to be hierarchically nested. Thus, heterogeneous models can co-exist in a system, and within each level, the model of computation is well-defined. A variety of models of computation has been implemented in Ptolemy II. We only present the models that are needed for modeling distributed hybrid systems.

2. Hierarchical Hybrid Systems

In Ptolemy II, hybrid systems are modeled by hierarchically nesting finite state machines (FSM) with continuous-time (CT) models, as shown in Figure 1.

Finite state machines are modeled using a special actor called an *FSMActor*, shown as actor B in Figure 1. An FSM actor contains states and transitions and exposes an actor interface. States in an FSM actor can refine into other components, which means that when the state machine is in that state, the component that refines the state will replace the FSM actor. Transitions have guards and actions. A guard on a transition is a predicate that indicate when the transition will be taken. For example, in Figure 1, the guard on the transition from q_1 to q_2 could be expressed in terms of the input of B and the output of B_1. When the transition is taken, the actions on the transition will be performed. Actions may reset variables in the refinement of the destination state, or produce outputs for the FSM actor.

In hybrid systems, an FSM state can refine into continuous-time differential equations, like

$$\begin{aligned} \dot{x} &= f(x, u, t) \\ y &= g(x, u, t) \end{aligned} \quad , \tag{1}$$

where x is the continuous state variable, u is the input, and y is the output. Such differential equations are modeled using integrators with feedback, where f and g are built using actors, which may internally implement other models, such as another hybrid subsystem. In order to interact with discrete models, event generators and waveform generators are introduced in the CT model [3]. These generators convert between continuous signals and discrete events. FSM actors can use the generated events in their guards.

3. Hybrid Simulation

The simulation of hybrid systems alternates between solving differential equations numerically and making discrete FSM transitions. Two challenges in the simulation are:

- managing the discretization of numerical integration steps to achieve both accuracy of continuous-time trajectories and precise timing of discrete events;

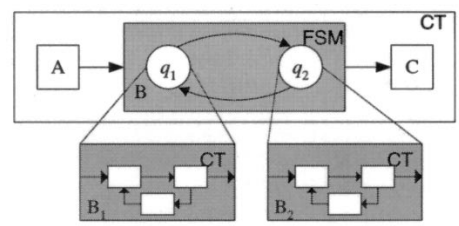

Figure 1. A hierarchical hybrid system.

This work is part of the Ptolemy project, which is supported by DARPA/ITO, the State of California MICRO program, and the following companies: Cadence Design Systems, Hewlett Packard, Hitachi, Hughes Space and Communications, Motorola, NEC, and Philips.

- managing the control flow across levels of hierarchy.

These issues are tackled in Ptolemy II by using variable step size numerical methods enhanced with event handling and coordinating CT and FSM directors [4]. The step size control mechanism in [4] handles both timed events (whose occurrence time is known beforehand) and state events (whose occurrence time depends on the value of the state variables).

4. Coordinating Hybrid Systems

Many complex systems consist of multiple hybrid subsystems that communicate through events. In Ptolemy II, the model that coordinates hybrid subsystems does not have to be a CT model. Instead, we can choose the best models that capture the properties of interest. In many cases, these models only need to manage a discrete set of events.

One advantage of using an event-based model to coordinate hybrid subsystems is that integration step sizes in the subsystems are decoupled. Other than detecting events, individual subsystems can adjust their step sizes solely according to their local continuous dynamics and independent of each other. This allows a subsystem with slow dynamics to execute with relatively large step sizes, without unnecessary synchronization with its fast peers.

We introduce two models to coordinate hybrid systems discretely – a discrete event (DE) model and a publish/subscribe (P/S) model. DE models feature a global notion of time, so it is possible to study the impact of communication delays and timing behaviors. The P/S model decouples the senders and the receivers of messages, so it is easy to model federations of subsystems with dynamic configurations.

4.1. Discrete-event model

In a discrete-event model, time is global to all components. An event has a value and a time stamp. Components, which could be internally implemented as a hybrid system, communicate via a set of events located discretely on the time line. A component, when executing, consumes input events and produces output events. In particular, components can increase the input time stamps to compute output time stamps. For example, a communication channel can be modeled as a component that delays its input packets for an (uncertain) amount of time, and may occasionally drop packets.

Additional subtlety has to be taken care of when using discrete event model to coordinate multiple continuous-time subsystems. That is the continuous subsystems have to run ahead of the discrete-event time and prepare for rolling back [3].

4.2. Publish and subscribe model

Publish and subscribe is a model for three-tier communication [1], [5], [6]. In this model, an event channel, also known as a persistent object space, mediates the communication between senders and receivers. Since no direct channel needs to be established between senders and receivers, P/S models are good at managing communicating peers that join and leave a federation.

Many P/S models are untimed. Time stamps on an event produced by one subsystem may not make sense in another subsystem. In such cases, systems usually need to synchronize with real time to exhibit reasonable relative execution progress. In Ptolemy II, special actors have been developed that use JavaSpaces [6] as the event channel. These actors can be integrated with hybrid system models to achieve coordination.

5. Conclusion

Ptolemy II provides a component-based architecture for modeling heterogeneous systems. Distributed hybrid systems can be modeled in this architecture using event-based high-level models and hybrid subsystems. The continuous-time simulation engine in Ptolemy II manages variable step sizes and handles event generation. Event-based high-level models allow the hybrid subsystems to adjust their step sizes independently. Among the event-based models, discrete-event models are good at studying timing properties, and publish/subscribe models are good at managing join-and-leave peers.

Reference

[1] T.H. Harrison, D.L. Levine, and D.C. Schmidt, "The Design and Performance of a Real-time CORBA Event Service," in Proceedings of OOPSLA '97, Atlanta, GA, October 1997, ACM.

[2] J.W. Janneck, E.A. Lee, J. Liu, X. Liu, S. Neuendorffer, S. Sachs, and Y. Xiong, "Disciplining Heterogeneity: the Ptolemy Approach," to appear in *ACM SIGPLAN 2001 Workshop on Languages, Compilers, and Tools for Embedded Systems* (LCTES 2001), Snowbird, Utah, June 22 - 23, 2001.

[3] J. Liu and E.A. Lee, "Component-based Hierarchical Modeling of Systems with Continuous and Discrete Dynamics," *Proc. of 2000 IEEE Symposium on Computer-Aided Control System Design* (CACSD'00), Anchorage, AK, Sept. 2000.

[4] J. Liu, X. Liu, T.J. Koo, B. Sinopoli, S.S. Sastry, and E.A. Lee, "A Hierarchical Hybrid System Model and Its Simulation," *38th IEEE Conference on Decision and Control* (CDC'99), Phoenix, Arizona, Dec. 1999.

[5] R. Rajkumar, M. Gagliardi, and L. Sha, "The Real-Time Publisher/Subscriber Inter-Process Communication Model for Distributed Real-Time Systems: Design and Implementation," in *First IEEE Real-Time Technology and Applications Symposium*, May 1995.

[6] E. Freeman, S. Hupfer, and K. Arnold, *JavaSpaces: Principles, Patterns and Practice*, Addison-Wesley, 1999.