Hierarchical Hybrid Modeling of Embedded Systems*


University of Pennsylvania
http://www.seas.upenn.edu/hybrid/

Abstract. This paper describes the modeling language Charon for modular design of interacting hybrid systems. The language allows specification of architectural as well as behavioral hierarchy, and discrete as well as continuous activities. The modular structure of the language is not merely syntactic, but is exploited by analysis tools, and is supported by a formal semantics with an accompanying compositional theory of refinement. We illustrate the benefits of Charon in design of embedded control software using examples from automated highways concerning vehicle coordination.

1 Introduction

An embedded system typically consists of a collection of digital programs that interact with each other and with an analog environment. Examples of embedded systems include manufacturing controllers, automotive controllers, engine controllers, avionic systems, medical devices, micro-electromechanical systems, and robots. As computing tasks performed by embedded devices become more sophisticated, the need for a sound discipline for writing embedded software becomes more apparent (c.f [23]). Model-based design paradigm, with its promise for greater design automation and formal guarantees of reliability, is particularly attractive given the following trends.

Software Design Notations. Modern object-oriented design paradigms such as Unified Modeling Language (UML) allow specification of the architecture and control at high levels of abstraction in a modular fashion, and bear great promise as a solution to managing the complexity at all stages of the software design cycle [7]. There are emerging tools such as RationalRose (see www.rational.com) that support modeling, simulation, and code generation, and are increasingly becoming popular in domains such as automotive software and avionics.

Control Engineering. Traditionally control engineers have used tools for continuous differential equations such as MATLAB (see www.mathworks.com) for modeling of the plant behavior, for deriving and optimizing control laws, and for validating functionality and performance of the model through analysis and

* Supported by DARPA MoBIES grant F33615-00-C-1707
simulation. Tools such as SIMULINK recently augmented the continuous modeling with state-machine-based modeling of discrete control.

**Formal Verification Tools.** Model checking is emerging as an effective technique for debugging of high-level models (see [10] for a survey). Model checkers such as SMV [26] and SPIN [20] have been successful in revealing subtle errors in cache coherency protocols in multiprocessors and communication protocols in computer networks. In recent years, the model checking paradigm has been successfully extended to models with continuous variables leading to tools such as UPPAAL [22], HyTech [18], and CheckMate [8].

This paper describes our modeling language, CHARON, that is suitable for high-level specification of interacting embedded systems. We proceed to discuss the three distinguishing aspects of CHARON.

**Hybrid Modeling.** Traditionally, control theory and related engineering disciplines, have addressed the problem of designing robust control laws to ensure optimal performance of processes with continuous dynamics. This approach to system design largely ignores the problem of implementing control laws as a piece of software and issues related to concurrency and communication. Computer science and software engineering, on the other hand, have an entirely discrete view of the world, which abstracts from the physical characteristics of the environment to which the software is reacting to, and is typically unable to guarantee safety and/or performance of the embedded device as a whole. An embedded system consisting of sensors, actuators, plant, and control software is best viewed as a hybrid system. The relevance of hybrid modeling has been demonstrated in various applications such as coordinating robot systems [2], automobiles [6], aircrafts [29], and chemical process control systems [13].

Early formal models for hybrid systems include phase transition systems [25] and hybrid automata [1]. While modularity in hybrid specifications has been addressed in languages such as hybrid I/O automata [24], CHARON allows richer specifications. Discrete updates in CHARON are specified by guarded actions labeling transitions connecting the modes. Some of the variables in CHARON can be declared analog, and they flow continuously during continuous updates that model passage of time. The evolution of analog variables can be constrained in three ways: differential constraints (e.g. by equations such as $\dot{x} = f(x, u)$), algebraic constraints (e.g. by equations such as $y = g(x, u)$), and invariants (e.g. $|x - y| \leq \varepsilon$) which limit the allowed durations of flows.

**Hierarchical Modeling.** Modern software design paradigms promote hierarchy as one of the key constructs for structuring complex specifications. We are concerned with two distinct notions of hierarchy. In architectural hierarchy, a system with a collection of communicating agents is constructed by parallel composition of atomic agents, and in behavioral hierarchy, the behavior of an individual agent is described by hierarchical sequential composition. The former hierarchy is present in almost all concurrency formalisms, and the latter, while present in all block-structured programming languages, was introduced for state-machine-based modeling in STATECHARTS [17].