

Verification and Integration of Real-Time Control Software

Rajeev Alur
University of Pennsylvania

Realizing the potential of networked control systems will be predicated upon our ability to produce *embedded software* that can effectively and safely harness the functionality of sensors and processors. Embedded software is different, and more demanding, than the typical programming applications in many ways. Modern programming languages abstract away from real time and resources, and do not provide adequate support for embedded applications. Consequently, current development of embedded software requires significant low-level manual effort for scheduling and component assembly. This is inherently error-prone, time-consuming and platform-dependent. Consequently, developing novel programming and implementation methodology for synthesizing portable, predictable embedded software is an important challenge for networked control systems.

Formal Verification

To provide assurance guarantees for control software, a formal approach to design is appealing. Model based design and formal verification have been successful in targeted applications such as microprocessor designs, and we believe that the same success is feasible in the domain of embedded control systems. There are multiple research challenges that need to be addressed to develop the model based approach. We list some of them along with possible emerging directions:

Modeling: The appropriate mathematical model for embedded software systems is *hybrid systems* that combines the traditional state-machine based models for discrete control with classical differential- and algebraic-equations based models for continuously evolving physical activities. For modeling the environment of embedded devices, the stochastic aspects also are essential. Another desirable feature of modeling is *compositionality* so that different components of the system can be described and analyzed in a modular manner. Developing a compositional and mathematically rigorous modeling framework for stochastic hybrid systems will be an important research direction.

Model Checking: Model checking tools can reveal design bugs at early stages by subjecting partial models for compatibility checks against specifications. Impressive progress in symbolic state-space exploration techniques has enhanced the applicability of model checking significantly. This has led to improved reliability for network protocols and device drivers. However, developing similar technology for networked devices needs sustained research in analysis tools for stochastic and hybrid models.

Software generation: Generating embedded software directly from high-level models, such as hybrid systems, is appealing, but challenging due to the wide gap between the two. In current practice, this gap is bridged with significant manual effort by exploiting the run-time support offered by operating systems for managing tasks and interrupts. A key challenge to systematic software synthesis from hybrid models is to ensure that one can infer properties of the software from the properties of the model, and this problem is receiving increasing attention from researchers.

System Integration

Contemporary software development emphasizes components with clearly specified APIs. A static API for a software component such as a Java library class consists of all the (public) methods, along with the types of input parameters and returned values, that the component supports. This promotes a clear separation between the specification of the component and its implementation. Such static APIs can be enforced using type systems, but while indispensable, only offer a partial solution to designing bug-free systems as these APIs do not capture constraints on resources, real-time guarantees, and other quality-of-service aspects. Consequently, they offer little assistance in “system” integration. This is an important issue not only for being able to derive system-level performance and correctness guarantees, but also for being able to assemble components in a cost effective manner.

Interfaces for Embedded Components: The notion of an interface for a control device interacting with its physical environment and other devices must incorporate information about timing delays and continuous parameters such as threshold levels. Capturing the notion of quality-of-service abstractly, and having mechanisms that can enforce the adherence to interfaces as well as check compatibility between components interfaces, is an emerging and challenging trend in embedded systems research.

Biography

Rajeev Alur is Zisman Family Professor and Graduate Group Chair of Computer and Information Science at the University of Pennsylvania. He obtained his bachelor’s degree in computer science from Indian Institute of Technology at Kanpur in 1987, and PhD in computer science from Stanford University in 1991. Before joining Penn in 1997, he was with Computing Science Research Center in Bell Laboratories. His areas of research include formal modeling and analysis of reactive systems, hybrid systems, concurrency theory, and design automation for embedded software. His awards include President of India’s Gold Medal for academic excellence (1987), US National Science Foundation’s CAREER (1997) and ITR (2001) awards, and Alfred P. Sloan Faculty Fellowship (1999). His recent research projects include Mocha (a model checker for compositional analysis of reactive systems using game-based requirements and assume-guarantee reasoning), Hermes (a modeling and verification tool for hierarchical state machines), OpEm (open APIs for embedded devices such as smartcards), JIST (Synthesis of behavioral interfaces for Java classes), and Charon (a modeling and verification tool for embedded control systems based on hybrid systems theory). Prof. Alur has been a coPI in DARPA’s initiative on model-based design of embedded systems, and ARO’s initiative on high confidence embedded systems. He currently serves as the Chair of ACM SIGBED (Special Interest Group on Embedded Systems).