Automatic Design Debugging

Heinz Riener[1]

Institute of Computer Science, University Bremen, Germany,
{hriener,fey}@informatik.uni-bremen.de,
http://www.zesy.uni-bremen.de

Cyber-Physical Systems (CPS) [4] integrate computation with physical processes and emerge naturally in many safety critical feedback control applications from the automotive and aerospace fields. CPS consist of *cybernetic parts* and *physical parts* interfaced together with actuators and sensors. The sensors observe the physical world and send feedback in form of signals to the cybernetic parts. The cybernetic parts analyze their input signals and influence the environment by stimulating actuators depending on their logical state. In contrast to embedded systems, the behavior of a CPS is determined by both, its physical and its logical state. As examples, consider the automatic correction of driver actions or the autonomous maneuver of satellites. In both applications precisely capturing the state of the environment in real-time is critical to guarantee the correct functioning of the system.

For safety critical systems guaranteeing the correctness (*design validation*) is a predominant problem. The veteran techniques for design validation of hardware and software systems, namely simulation and testing, are inadequate for CPS because of their continuous state space. For the same reason, static methods render undecidable [3] for hybrid models that capture the logical and the physical state of a CPS. Thus, new techniques [1,5] have been developed tailored to the requirements of CPS. These techniques combine static and dynamic approaches and concentrate on a subset of "interesting" CPS with robust dynamics for which effective decision procedures can be established.

Other important problems like *design debugging* have hardly been investigated for CPS. Design debugging deals with localizing and correcting faults in a CPS that refutes its specification. In localization, potentially faulty components of the system are determined which have to be manually inspected. To keep the manual effort tractable, the number of potentially faulty components should be kept as low as possible but without excluding the faulty components. The resolution of debugging depends on the granularity of the system models. For instance, a faulty component may refer to an expression in a high-level programming on the system-level or a gate in hardware on the gate-level.

Formal approaches to design debugging divide into model-based [6] and explanation-based approaches [2]. Model-based approaches determine potentially faulty components by logical reasoning about a model of the system and the refuted specification from first principle. Systematically the influence of individual components onto the system's correctness is examined. A component is marked as potential faulty if and only if the system becomes correct when this component is assumed to behave non-deterministic. Explanation-based approaches characterize faults by comparing similar faulty and correct execution traces of the system.

The differences of the execution traces give a possible explanation of a symptom of a faulty system behavior. These approach have been investigated for hardware and software systems [7,8,9] but not yet for CPS.

Beyond localization, correction (closely related to synthesis) deals with finding replacements for potentially faulty components of the system such that the specification of the system is satisfied. In contrast to synthesis, correcting starts with an almost correct system that needs slight adaptions to become correct. Especially, the overall architecture of the system structure is already available which makes it in practice simpler than synthesis the system from scratch.

Neither fault localization nor fault correction has been studied for CPS. Similar to the problem of design validation, existing methods are inadequate when dealing with the continuous state space of a CPS. We imagine that effective debugging approaches are valuable for reducing the development time and development costs.

References

1. Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *Computer Science Logic*, pages 126–140, 1999.
2. Alex Groce, Sagar Chaki, Daniel Kroening, and Ofer Strichman. Error explanation with distance metrics. *International Journal on Software Tools for Technology Transfer*, 8(3):229–247, 2006.
3. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.
4. Edwared A. Lee and Anjit A. Seshia. *Introduction to Embedded Systems — A Cyber Physical Systems Approach.* LeeSeshia.org, 2011.
5. Stefan Ratschan. Safety verification of non-linear hybrid systems is quasi-decidable. *Formal Methods in System Design*, 40(1):71–90, 2013.
6. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
7. Heinz Riener and Görschwin Fey. Model-based diagnosis versus error explanation. In *International Workshop on System Level-Design of Automotive Electronics/Software in conjunction with Design Automation Conference*, 2012.
8. Heinz Riener and Görschwin Fey. Model-based diagnosis versus error explanation. In *International Conference on Formal Methods and Models for Codesign*, pages 43–52, 2012.
9. Heinz Riener and Görschwin Fey. Yet a better error explanation algorithm. In *16. ITG/GMM/GI-Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, pages 193–194, 2013.