APPLICATION OF FORMAL METHODS IN FACTORY AUTOMATION TO REDUCE CONFIGURATION-INDUCED MISPROCESSING RISKS

—EXTENDED ABSTRACT—

Levent Erkok (levent.erkok@intel.com)
Intel Corporation, Ronler Acres 3, MS:250, Hillsboro OR

An essential aspect of deploying reliable software architectures is the verification of safety and liveness properties. Briefly, safety refers to freedom from erroneous states, and liveness refers to freedom from dead-locks. In this paper, we describe a formal methods based approach to verify certain such properties of Intel's station controller architecture. Our approach represents a significant paradigm shift from the current simulation/testing based methodologies. We introduce a technique that automatically constructs a formal model of Intel's station controller lot-processing logic, which is then used in a model checking environment to verify several desired properties automatically. The automation aspect is important: Once we set-up our verification environment, no further manual intervention is necessary. Hence, the return-on-investment of the described verification framework is quite high.

Intel's station controllers lie at the heart of our factory automation control systems. Station controllers manage the interaction between manufacturing equipments and the remainder of factory automation; relieving the users from error prone task of manual interaction with a distributed set of applications. Station controllers are designed to reduce misprocessing risks. Misprocessing can result from a number of issues such as user errors, inconsistent data, tool errors, etc. Our work focuses on addressing misprocessing risks resulting from incorrect configuration changes. Basically, we try to automatically identify and prevent configuration changes that may have an unexpected adverse effect on other components. This class of problems is seen over and over on the factory floor. A significant amount of engineering time is spent in analyzing and troubleshooting configuration issues, and such bugs cause loss of production time, involve potential misprocessing risk, and inevitably result in loss of revenue.

How can we verify that configuration changes will not cause any issues? Verification attempts can take many forms, stretching the spectrum from peer reviews to full-fledged formal modeling and computerized verification. Clearly, neither extreme is satisfactory. Peer reviews are known to be extremely prone to error, and full formal modeling is almost always insurmountable. Obviously, it is not feasible to formally verify every piece of code running in a complex network of distributed applications. Driven by practical needs and constraints, the current state of-the-art is to formally verify "core" portions of the underlying system. To this end, we have chosen to formally model and verify properties about CT (configurable transaction) scripting facilities associated with station controllers. A CT script is a dynamically loaded piece of code that wraps around the usual lot processing logic. Different CT scripts can be specified for different kinds of tools, and also for different kinds of jobs on the same tool. The name "configurable" indicates that tool engineers are free to configure these scripts as they see fit, and the associated station controller will execute the specified commands as required. This flexibility comes at a price. It is implicitly assumed that CT scripts behave properly; i.e., they neither cause the system to go into an undesirable indeterminate state, nor cause any dead-locks. Our work provides a methodology for automatic verification of these properties, augmenting the current techniques based on simulation and manual tests.

The formal verification technology we use is based on "Computation Tree Logic," a.k.a. CTL. CTL is a temporal logic designed in early 1980's.The most distinguishing feature of temporal logics is the existence of connectives that explicitly refer to time, allowing one to express time varying truths. From a computational viewpoint, this expressive power allows us to assert conditions that will hold at a future state of a computer program. It is precisely this facility that enables us to formulate and verify properties like the ones we want to verify about CT scripts. Using an in-house developed formal model of Intel's lot processing logic, we employ a model checking based approach to prove several assertions about CT scripts. Once verification is complete, the user is given feedback on the validity of the assertions. In case of failure, a counter-example execution path is also provided. Clearly, these counter-examples are extremely valuable in identifying root causes of issues.

We have used our verification system on CT scripts that are used in Intel's production fabs. Our system was able to identify several problems that were missed during the testing/deployment phases in a completely automated manner. The counter-examples produced by the system pin-pointed the problems in the underlying scripts, saving valuable engineering time.

Formal verification techniques have been used in the industry in hardware verification for quite some time. Our work provides a proof-of-concept that formal verification is also applicable to real-life software architectures that are in widespread use in semi-conductor factory automation. Future applications of this methodology could be quite far-fetching. An ultimate goal could be the automatic generation of production software from formal models, which would in turn imply major changes and quality improvements in automation software architecture. Clearly, we are still quite far away from this last point, but the prospects are encouraging and are well worth exploring.