Runtime Verification on Robotics Systems

Zhijiang Dong, Middle Tennessee State University, Murfreesboro, TN, USA
Yujian Fu, Alabama A&M University, Huntsville, AL, USA
Yue Fu, Fuyuan High Tech Co., Tianjin, China

ABSTRACT

Runtime verification is a technique for generating monitors from formal specification of expected behaviors for the underlying system. It can be applied to automatically evaluate system execution, either on-line or off-line, analyzing extracted execution traces; or it can be used online during operation, potentially steering the application back to a safety region if a property is violated. As a so-called light-weighted formal method, runtime verification bridges the gap between system design and implementation and shortens the distance of software quality assurance between the software testing and model checking and theorem proving. Runtime verification is considered as a highly scalable and automatic technique. Most of current runtime verification research are endeavored on the program context, in other words, on the program side and falls in the implementation level. These applications limited the benefits of runtime verification that bridges the gap among types of applications. With the proliferation of embedded systems and mobile device, dynamically verifying the firmware and mobile apps becomes a new emerging area. Due to the characteristics of runtime verification technique and limitations of the robotics systems, so far, very few research and project are located in the runtime verification on the firmware of embedded systems, which appear in most of robotics systems. Robotics systems are programmed on the firmware and only observed on device. In this paper, the authors first discussed the current runtime verifications on the embedded systems with limitations. After that, a layered runtime verification framework will be presented for the firmware verification. The case study is applied on the commonly recognized educational toolkit – LEGO Mindstorm robotics systems.

Keywords: Autonomous Systems, Embedded Systems, Formal Methods, Quality, Software Assurance, Testing, Trust

INTRODUCTION

To ensure the trusted and confident system in both design and implementation levels, to make the practice of designing from high level system specification a reality, verification methods must accompany every step in the design flow. Formal specification languages with formal definitions of systems requirements at the design level make formal verification possible. System properties that are specified by logic in the most cases can be proved through various formal verification tools (such as SPIN, SMV, etc). Two main limitations are widely observed in current research.
of computer systems – one is the limitations of current model checking and testing techniques, another is the application domains of verification techniques.

Although model checking has been widely applied in the system model verification for decades with its elegant algorithms and various of reduction mechanisms on the state explosion, it still suffers the problem of scalability on large complicated system. On the other hand, at the lower level of the system development process, the complexity can quickly overwhelm the automatic tools with its tons of implementation issues. Testing scales well, and is by far the most used technique in practice to validate software systems. The merge of testing and temporal logic specification is an attempt to achieve the benefits of both approaches, while avoiding some of the pitfalls of ad hoc testing and the complexity of full-blown theorem proving and model checking. Since evaluating test case executions manually is time consuming, it is necessary to generate test oracles in an easy and automated manner from high level specifications. Furthermore, in the context of highly reliable and/or safety critical systems, one would actually want to monitor a program execution during operation and to determine whether it conforms to its specification. Thus runtime verification become the workhorse for system verification and validation.

Runtime verification is a technique for generating monitors from abstract specification of expected behaviors for the underlying system. It can be applied to automatically evaluate system execution, either on-line or off-line, analyzing extracted execution traces; or it can be used online during operation, potentially steering the application back to a safety region if a property is violated. As a so-called light-weighted formal method, runtime verification bridges the gap between system design and implementation. Runtime verification is considered as a highly scalable technique. Several runtime verification systems have been developed, of which some were presented at recent international workshops on the runtime verification.

Most of current runtime verification research are endeavored on the program context, in other words, on the program side and falls in the implementation level. These applications limited the benefits of runtime verification that bridges the gap among types of applications. With the proliferation of embedded systems and mobile device, dynamically verifying the firmware and mobile apps becomes a new emerging area. So far, very few research and project are located in the runtime verification on the firmware of embedded systems. Robotics systems are typical systems that are programmed on the firmware and only observed on device. In this paper, we propose a layered runtime verification framework for the firmware verification on the LEGO Mindstorm robotics systems.

This framework monitors system model in terms of behaviors in the source code and binary code online. The monitored properties fit into the properties that can be observed in both source code and binary code. The integration of monitored results from both levels will be analyzed and synthesized to gain the collected behavior. Similar as those works on the runtime verification, in this framework we adopts LTL as the property specification language. A translation algorithm is presented to obtain the consistent implementation of the architecture in the programming language context.

**Current Situations of Runtime Verification on Embedded Systems**

Quality assurance of embedded system brings more challenge in the current academic society due to the complexity of the systems, which mainly comes from the multiple interface, various platforms and adaptation to the environment. The multiple factors of embedded systems increase the concurrency of the system functions and behaviors which in turn increase the complexity. Traditional verification approaches are mainly rooted on the formal specification languages, which are good at software system modeling and representation. With the networking, system-of-chips
Just Doesn’t Look Right: Exploring the Impact of Humanoid Robot Integration into Explosive Ordnance Disposal Teams
www.igi-global.com/chapter/just-doesnt-look-right/84954?camid=4v1a