Formal Verification Methods

Osman Hasan
School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Pakistan

Sofiène Tahar
Concordia University, Canada

BACKGROUND

On June 1, 2009, Air France Flight 447 from Rio de Janeiro to Paris crashed into the Atlantic Ocean, killing all 216 passengers and 12 crew members. Prior to the disappearance of the A330 aircraft, the automatic reporting system sent messages indicating disagreement in the airspeed readings, which led investigators to believe that the pilot probe sensors did not "accurately" measure airspeed and the autopilot may have automatically disengaged. Like the above probes, many systems are increasingly being used in safety-critical domains, such as medicine, transportation systems, chemical plants, etc. There is hence a dire need to ensure the accuracy of such systems as a system bug, or error, may endanger human life or lead to a significant financial loss.

In order to ensure error-free systems, the system design process is usually accompanied by a rigorous system analysis to check if the designed system would exhibit the desired behavior. The core of system analysis is based on mathematics and the fundamental idea is to create a mathematical model of the system and then use logical or mathematical reasoning to verify that the desired properties hold for this model. Traditionally, system analysis is done by paper-and-pencil proof methods. However, considering the complexity of present age systems, such kind of analysis is notoriously difficult, if not impossible, and is quite error prone due to the human error factor. Moreover, it is quite often the case that mathematicians forget to pen down all the assumptions that are required for the validity of their analysis. This fact may also result in designing erroneous systems. With the advent of computers, many computer based software tools based on the principle of testing or simulation have been introduced for system verification. Due to the reliable and efficient bookkeeping characteristic of computers, large systems can be analyzed and thus one of the limitations of paper-and-pencil proof methods can be overcome. However, the main problem with such kind of analysis is its completeness as the system is checked only for a subset of possible inputs since exhaustive testing is not possible for any system with a significant number of inputs due to the exponential growth of the test patterns. In Dijkstra's words "*Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence.*" Moreover, testing or simulation cannot be used to precisely verify properties about continuous systems due to the usage of computer arithmetic, like floating-point or fixed-point representations of real numbers. The above mentioned inaccuracy limitations of commonly used system analysis techniques can be held responsible for many unfortunate incidents that happened due to an erroneous system deployed in a safety-critical domain For example, the Therac-25 software bug and the Intel Pentium's floating-point division unit error).

INTRODUCTION

In order to raise the reliability of system analysis, a system analysis technique is required that can have the precision of paper-and-pencil based mathematical proofs, and thus does not rely upon computer-arithmetic, and utilizes the computers for bookkeeping, to be able to handle complex systems without having to worry about human-errors. Formal verification methods, which are primarily based on theoretical computer science fundamentals like logic calculi, automata theory and strongly type systems, fulfill these requirements. The main principle behind formal analysis of a system is

DOI: 10.4018/978-1-4666-5888-2.ch705

to construct a computer based mathematical model of the given system and formally verify, within a computer, that this model meets rigorous specifications of intended behavior. Due to the mathematical nature of the analysis, 100% accuracy can be guaranteed.

The history of formal methods dates back to Knuth and Dijkstra as both of them advocated the topic. Formal verification methods started to be investigated as computer-aided design (CAD) tools in the 1970s for software verification. However, the interest was marred by the fact that the software bugs can be easily fixed by releasing a software patch and thus the added reliability of software is not worth the rigorous exercise of formal verification. There was some research activity related to the formal verification of security systems funded by the US National Security Agency in the 1980s but the real catalyst for the active research interest in formal verification was their usage in verifying digital hardware systems in late 1980s. This is mainly because hardware descriptions are often more regular and hierarchical than software ones, hardware primitives are less obscure than the ones used in software and the cost of an uncaught design bug in hardware is much more profound than software since the hardware silicon chip once fabricated cannot be fixed by releasing a patch but instead has to be re-designed and re-fabricated, which costs considerable amount of time and money. The Intel floating-point division bug in 1994 further enhanced the interest in formal hardware verification and the industry started to adopt formal hardware verification tools in their design flows in late 1990s (Kropf, 1999). With the success of formal verification in hardware and due to some interesting developments in the underlying technologies, it started to be used again for software, transportation and security system analysis domains. Moreover, researchers started to explore the formal verification of physical systems, such as control systems, robotics and analog circuits, and biological systems by using powerful abstraction techniques to reduce the complexity of observable phenomena to what is relevant for a particular purpose. Recently, formal verification methods have also been used to verify complete system models, along with their continuous and unpredictable physical realities. The future of formal methods seems to be quite promising and besides academia, industry giants, like Intel and Microsoft, are also actively involved in formal methods related research.

The added benefits of formal verification methods come mainly at the cost of extreme rigor. Generally speaking, the expressiveness of a formal verification method is in direct proportion with the amount of required user intervention. Thus, formal verification of complex systems is more challenging and time consuming. Therefore, the general trend is to use a lightweight approach, i.e., use traditional verification methods, like simulation or testing, where accuracy of the analysis is not a big concern while using formal verification methods for the critical sections of the systems. On similar lines, hybrid formal verification methods are also being developed which allow us to partition the overall system model based on its complexity levels and thus facilitate using automatic formal verification methods for the rather simpler sections of the system while using the interactive methods with the complex sections.

Generally, formal verification methods are classified based on their underlying logic, expressiveness and decidability. The most commonly used formal verification methods include *theorem proving*, *symbolic simulation* and *model checking*. All of these have their own strengths and weaknesses. They have been used successfully to verify a variety of real-world systems. In the rest of this article, we provide a brief introduction to these widely used formal verification methods along with some of their practical applications. Finally, the article ends with some discussions and conclusions.

THEOREM PROVING

Theorem proving or automated reasoning is one of the most generic and widely used formal verification method. The system that needs to be analyzed is mathematically modeled in an appropriate logic and the properties of interest are verified using computer based software tools usually called theorem provers. The use of formal logics as a modeling medium makes theorem proving a very flexible verification method as it is possible to formally verify any system that can be described mathematically. The core of theorem provers usually consists of some well-known axioms and primitive inference rules. Soundness is assured as every new theorem must be created from these basic axioms and primitive inference rules or any other already proven theorems or inference rules. A question that may arise here is that why do we need logic to model the system and why natural languages