Formal methods in LARGE-SCALE computing systems

Saif U. R. Malik FBCS and Samee U. Khan FBCS, from the Department of Electrical and Computer Engineering, North Dakota State University, talk about the significance and role of formal methods in delivering reliable and correct software.

Computer programming is difficult and error-prone. In critical and large-scale computing systems, such as real-time systems and clouds, the errors are hazardous and expensive. An example of such errors was reported by Mars Orbiter Mishap Investigation Board in September 1999, when the Mars Climate Orbiter crashed because of a 'silly mistake' caused by the incorrect distance units in a program. The distance unit miscalculation caused the aircraft to go too close to Mars, which eventually caused the crash.

Formal method (FM) techniques have matured considerably as a verification discipline in the past few decades and have become a mainstream technology in industrial design, verification methodologies, and processes. The increasing criticality and complexity of applications and the role of software and hardware in those applications has led to the maturity of FM techniques. The aim of such techniques is to increase the quality of software by mathematically proving program correctness as opposed to using test cases.

The Mars Climate Orbiter crashed because of a 'silly mistake' caused by the incorrect distance units in a program.

Large-scale computing systems, such as clouds, have an order of tens of thousands or more computer servers. According to a research done by Huan Liu, Accenture Analyst and Research Manager in Silicon Valley, the Amazon Cloud is backed up by 454,400 computer servers.

Massive data analysis applications are executed over the cloud that requires huge amounts of memory, enormous numbers of processor cycles, and communication bandwidth. The model implemented over the cloud is pay-per-usage, which means that the end-users will pay only for the services usage. Therefore, the cloud service provider (CSP) has to maintain a certain level of quality of service to keep up the reputation.

If the CSP does not adhere to the specified quality of services, then the users may not use those services, and this is where the FM techniques come in handy, by providing a certain level of reliability and correctness to the system.

Software horror stories

According to Greg Linden, Principal at Microsoft Live Labs, Google reported a 20 per cent revenue loss due to a delay of 500msecs in response time. Moreover, Amazon reported a sales decrease of 1 per cent due to an additional response time of 100msecs. These examples indicate the importance and impact of the slightest inaccuracy in large-scale computing systems.

According to the New York Times in August 2012, the Knight Capital Group lost US$440 million in just 45 minutes, when newly installed trading software went haywire. In another incident on New Year's Eve in 1997, in Southland, New Zealand, a software failure to account for a leap year and considering only 365 days in a year as valid, caused more than AU$1 million of loss.

Roger Sessions, CTO of an online newsletter ObjectWatch and Lead Architect for Object Persistence in IBM, states in his report published in 2009 that, 'We are already losing over US$500 billion per month on IT failure, and the problem is getting worse.'

Determining correctness

The correctness of a program is based on a specific standard, which is called specification.

The specification provides a complete description of the behavior of a system to be developed and also includes use cases to describe user interactions with the software. Two techniques are most widely used to eliminate the presence of failures in software: testing and FMs (which includes formal verification).

In testing, the program is executed with a set of inputs to evaluate the differences between given input and expected output. The goal in testing is to reduce the frequency of failures.

Edsger Dijkstra (May 1930 – August 2002) said, 'Testing can be used to show the presence of bugs, but not their absence.' The other technique is FMs that mathematically prove that the software has no defects. The goal in FM techniques is to detect and eliminate failures.

Because FMs presuppose program semantics that are not considered in testing, FM techniques are considered more powerful than testing. Moreover, through FMs, users can logically analyze the system to prove properties for any possible inputs.

However, in testing the set of inputs assumes to cover both the range of normal uses and exceptional circumstances, which is unrealistic. Furthermore, the size of the computing systems is so huge that testing becomes unfeasible.

The FM technique provides a firm foundation for consistency amongst related activities by introducing rigor in all phases of software.

Bill Gates, Co-founder and Chairman of Microsoft, on his address at WinHEC in 2002 stated that, 'Things like software verification, have been the Holy Grail of computer science for many decades and now in some very key areas, for example, driver verification, we are building tools that can do actual proof about the software and how it works to guarantee the reliability.

Applications

The FMs are used in almost every field of computers from fine-grained circuit design verification to coarse-grained NASA extravehicular activity system verification. In large-scale computing systems FMs are applied in different areas of hardware and software, including routers, Ethernet switches, routing protocols, and security applications.

We applied FM tools to verify data center routing protocol in [1] and [2]. IBM used ACL2, a FMs theorem proving tool, in AMD x86 processor development process. Intel uses FMs to verify its hardware and firmware (permanent software programmed into a read-only memory).

Because FMs presuppose program semantics not considered in testing, FM techniques are considered more powerful than testing.

According to Jim Woodcock FBCS the broadest application of FM techniques is in the legacy code maintenance of Microsoft products.

There are several projects at NASA in which FMs are applied, such as next generation air traffic system (NextGen), unmanned aircraft system integration in the national airspace system and airborne coordinated conflict resolution and detection (ACCoRD).

Benefits

There are several advantages of using FMs for the specification and analysis of large-scale computing and real-time systems. The FM techniques help early identification of incompleteness, ambiguities, and fallacies.

Moreover, through FM techniques, analysis and verification of large-scale computing systems can be performed using automatic or machine-assisted tools that are unfeasible in a testing scenario. Some techniques of FMs generate a counter-example if a model of the system is incorrect. This allows the evaluation of design alternatives, without expensive prototyping.

Cliff B. Jones FBCS stated that, 'all engineering disciplines make progress by employing mathematically based notations and methods.' Moreover, he calls FM techniques 'thinking tools' for computer systems.

People would be scared if they knew that an aircraft they use to travel, a bridge they cross every day, or a software system is not designed using a sound methodology. In large-scale computing systems, such as clouds, if the users have fears on the reliability or security applications, then they are not going to use it. Therefore, the use of FMs must be adopted to ensure the correctness, reliability, robustness, and safety of the systems.

References

[1] S. U. R. Malik, S. K. Srinivasan, S. U. Khan, and L. Wang, 'A Methodology for OSPF Routing Protocol Verification', 12th Intlernational Conference on Scalable Computing and Communications (ScalCom), China, December 2012.

[2] S. U. R. Malik, S. K. Srinivasan, and S. U. Khan, 'Convergence Time Analysis of Open Shortest Path First Routing Protocol in Internet Scale Networks', IET Electronics Letters, vol. 48, no. 19, pp. 1188-1190, 2012.

More software failure stories can be found on: www.cs.tau.ac.il/~nachumd/horror.html

Image: iStockPhoto/133495516

doi:10.1093/itnow/bwt025 ©2013 The British Computer Society