SIG on Formal Methods

NEWS LETTER VOLUME 9, SEPTEMBER 2015

SIG on Formal Methods:

Formal Methods (FM) has been in existence since 1940's, when Alan Turing proved the logical analysis of sequential programs using the properties of program states; and Floyd, Hoare and Naur used axiomatic techniques to prove program correctness against the specifications in 1960's.

These initial successes helped inculcate an interest in applying FM to the field of computer science.

Academia has been instrumental in bringing this field to the forefront, through continued research and development. The use of Formal Methods requires an expert skill-set expertise and therefore, its use is limited to those trained in the field.

Mission Statement:

Computer Society of India wants the field of Formal Methods to have a wider audience and more people to benefit from the application of these methods to all spheres of life. There is a need to use effective, correct and reliable approaches to design, develop and qualify complex, high assurance system software with the rigid schedules and budget. For this we need advanced tools, techniques and methods. Industry standards like RTCA DO-178C (Civil Aerospace), ISO 26262 (Automotive), IEC 61513(Nuclear), EN50126 (Railways) have recommended the usage of formal –method based approach to be used in the various phases of engineering process to achieve the required levels of safety and security.

Today there are proven techniques and tools that can be used in specification, design and verification & validation phases to assure correct requirement-capture, implementation, software functionality and security. This helps in developing high assurance software for applications such as cyber-physical systems, net-centric warfare systems, autonomous robots and Next Generation Air Transportation.

- **One day workshop on FM was conducted during April 2013**

Objectives of the Special Interest Group (SIG) are:

- To bring together scientists, academicians active in the field of formal methods and willing to exchange their experience in the industrial usage of formal methods
- To coordinate efforts in the transfer of formal methods technology and knowledge to industry
- To promote research and development for the improvement of formal methods and tools with respect to their usage in industry.
- To bring out practical engineering methods where formal methods will be integrated with current engineering methods

Some of the known applications of formal methods are:

- Formal verification, including theorem proving, model checking, and static analysis
- Techniques and algorithms for scaling formal methods
- Use of formal methods in automated software engineering and testing
- Model-based formal development
- Formal program synthesis
- Formal approaches to fault tolerance
- Use of formal methods in safety cases
- Use of formal methods in human-machine interaction analysis
- Use of formal methods in compiler validation and object code verification

3. Committee Members

1. Ms. Bhanumathi K S, Convener
2. Mr.Chander Mannar
3. Prof.Anirban basu
4. Mr. Suman Kumar
5. Prof. Shyam Sundar
6. Ms. Manju Nanda
7. Ms. J. Jayanthi
8. Ms. Saroja Devi
9. Prof. Meenakshi D'Souza
10. Dr. Yoganand Jeepu
11. Dr. Swatnalatha Rao
12. Dr. Aditya Kanade
13. Dr. A. Indira
14. Mr. Dhinakaran Pillai

Convener:

Bhanumathi K S
“Ganadhakshya” #406, 8 C Main,
H R B R First Block
Kalyan Nagar
Bangalore 560043
Email:bhanushekar@gmail.com
Mobile:+91 95350 92589

Generating Test Cases from Formal Specifications

Compiled by Bhanumathi K S

Formal verification techniques depend on mathematically precise specifications. But developing rigorous system tests also requires a precise, complete description of system functions, and practical system assurance always requires testing, even when formal methods are used. One of the most interesting applications of formal methods has been the development of tools that can generate complete test cases from formal specifications. Although a large number of "automated testing" tools are available on the market, most of these tools automate the more mundane aspects of software testing: generating test data, passing input data to the system under test, and recording results. Defining the correct system response for a given set of input data is the hard task that most tools cannot accomplish when system behavior is defined only with natural language specifications. Because the expected system response can only be determined by reading the specification, programmers are expected to provide this critical missing link in most automated testing systems. The great advantage of test generation tools based on formal methods is that a formal specification describes system behavior mathematically, so expected system responses for particular inputs can be generated; i.e., the tool can generate complete test cases, not just test data input or test scaffolding.

From a cost-benefit standpoint, generating tests from specifications can be one of the most productive uses of formal methods. Approximately half the staff time in a typical commercial software development effort is spent on testing. As computer users have recognized, even this level of effort only removes the most obvious flaws. Much of the software industry operates under a marketing strategy that gives feature richness and time to market a higher priority than quality, because users have demonstrated a willingness to accept bugs in return for more features. Some of the newer test generation tools hold the promise of improving quality while simultaneously reducing time to market, because less developer time is spent on programming test cases. These tools can also provide benefits for custom software, such as most M&S systems, by reducing the time spent on test development and thus allowing more time for other tasks. Some empirical measurements have

shown that tests generated by these tools provide test coverage as good or better than that achieved by manually generated tests, so developers can choose between producing more tests in the same number of staff hours, or reducing the number of hours required for testing.

References:

- Cost Effective Use of Formal Methods in Verification and Validation, D. Richard Kuhn, Ramaswamy Chandramouli and Ricky W. Butler, http://csrc.nist.gov/groups/SNS/asft/documents/Foundations_2002.pdf