جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila
University of Mohamed Boudiaf - M'sila

1985

Formal Verification and Specification

Chapter 01: Introduction to Formal Methods

Dr. Hichem Debbi

hichem.debbi@gmail.com

March 7, 2022

Software is everywhere

S

oftware is being used more and more in almost all aspects of daily life, e.g. in transportation, finance, health care, government, and telecommunications.

Safety critical based systems

reliability of such software is critical for us, especially when failures may lead to catastrophes where people die or values/money are lost.

Critical Examples

- A software controlling the trains collides
- A pacemaker cease to function
- A rocket my explode due to false data interpretation
- A malfunction of a vehicle's airbag

Critical events

- Pentium bug Intel Pentium chip, released in 1994 produced error in floating point division
  Cost : $475 million
- ARIANE Failure In December 1996, the Ariane 5 rocket exploded 40 seconds after take off . A software component threw an exception
  Cost : $400 million.
- Therac-25 Accident : A software failure caused wrong dosages of x-rays in 1987.
  Cost: Human Loss.

Critical events

Figure: Y2K problem, Millennium bug, Y2K bug

Traditional Solutions

Testing & Simulation
Reviews & Walkthroughs

- Inadequate for safety-critical systems
- Detects presence of bugs not absence
- Late Detection of bugs

Software Development

T

he aim of developing software is to reason about properties of what is being developed.

Definition

Formal methods are a particular kind of mathematically rigorous techniques for the specification, development and verification of software and hardware systems.

Why

The use of formal methods is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical modelling and analysis can contribute to the correctness of the resulting product.

Formal methods

A formal method (FM) whose techniques and tools can be explained in **mathematics**.
FM if includes, as a tool, a **specification language**, then that language has a **formal syntax**, a **formal semantics**, and a **formal proof** system.
The techniques of a formal method help construct a specification, and/or analyse a specification, and/or transform (refine) one (or more) specification(s) into a program.

Formal specification methods - History

The advent of the first formal specification languages, VDM and Z, were not "accompanied" by any tool support: no syntax checkers, nothing ! Academic programming was done by individuals.
It seems to be a fact that industry will not use a formal method unless it is standardized and “supported” by extensive tools. Most formal method specification languages are conceived and developed by small groups of usually university researchers. This basically stands in the way of preparing for standards and for developing and later maintaining tools.

Model-oriented Specification Methods:

- **Specification and model-oriented:** Focus on convenient and expressive specification languages and their semantics. The main challenge is considered to be how to write simple, easy to understand: VDM, Z, B, RAISE/RSL
- **Analysis and model-oriented**: Focus on analysis: Event-B, Alloy, SPIN, SMV

Algebraic Methods

- **CafeOBJ** and **Maude**: use equational logic by rewriting and can be used as a powerful interactive theorem proving systems. They are originally inherited from OBJ3.

- Tools that support validation through **simulation**
- Tools supporting validation via **test generation**
- Tools that support **formal verification**

Formal Specification

We need formal specification to prove a set of properties that the system should satisfy:
Describing "what" the system should do without "how" to do it.
Reduce faults in systems:

- Invest more effort is early stage of system development
- Requirement errors can be discovered as early as possible and resolved

Formal Specification Parts

Statical aspect

- The states it can occupy
- Invariants which will always hold

Dynamic or behavioral aspect

- All possible operations
- The relations of inputs to outputs
- Changes of state that can occur (actions/events)

Functional/non-Functional properties

Functional properties

The value of a variable should not .. The output y depends on the input x..

Non-Functional properties

Response time should not ... Less data failures

Abstraction/Refinement principles

Abstraction

Abstraction is a simple way for representing systems details in more compact form.

Refinement

Refinement is a sample way for building systems with higher detail starting from simple ones by adding features gradually.

Abstraction to Refinement

We start from abstract model of a system with basic functionalities and try to reach a concret model, after several steps of refinement.

Formal Verification Techniques

- Equivalence checking
- Model checking
- Theorem proving

Theorem proving

- Specification and design are logical formulae
- Checking involves proving a theorem

Types: Semi-automatic
High degree of human expertise required
Mainly confined to academic
Automatic: success in industry
Number of public domain tools :
Nqthm, STeP, PVS, HOL

Theorem proving

ACL2, successor to Nqtm (Boyer-Moore Theorem Prover), used at AMD to formally verify floating point units

Model Checking

Another promising automatic technique; Checking design models against specification with the following properties: Specifications temporal properties and environment constraints; use of temporal logic and automata; Checking is automatic with bug traces

Many Commercial and academic tools: Spin (Bell Labs.), Formal-Check (Cadence), VIS (UCB), SMV (CMU, Cadence)

In-house tools: Rule Base (IBM), Intel, SUN, Bingo (Fujitsu), SLAM (Microsoft), etc

Programming languages: Java-Pathfinder

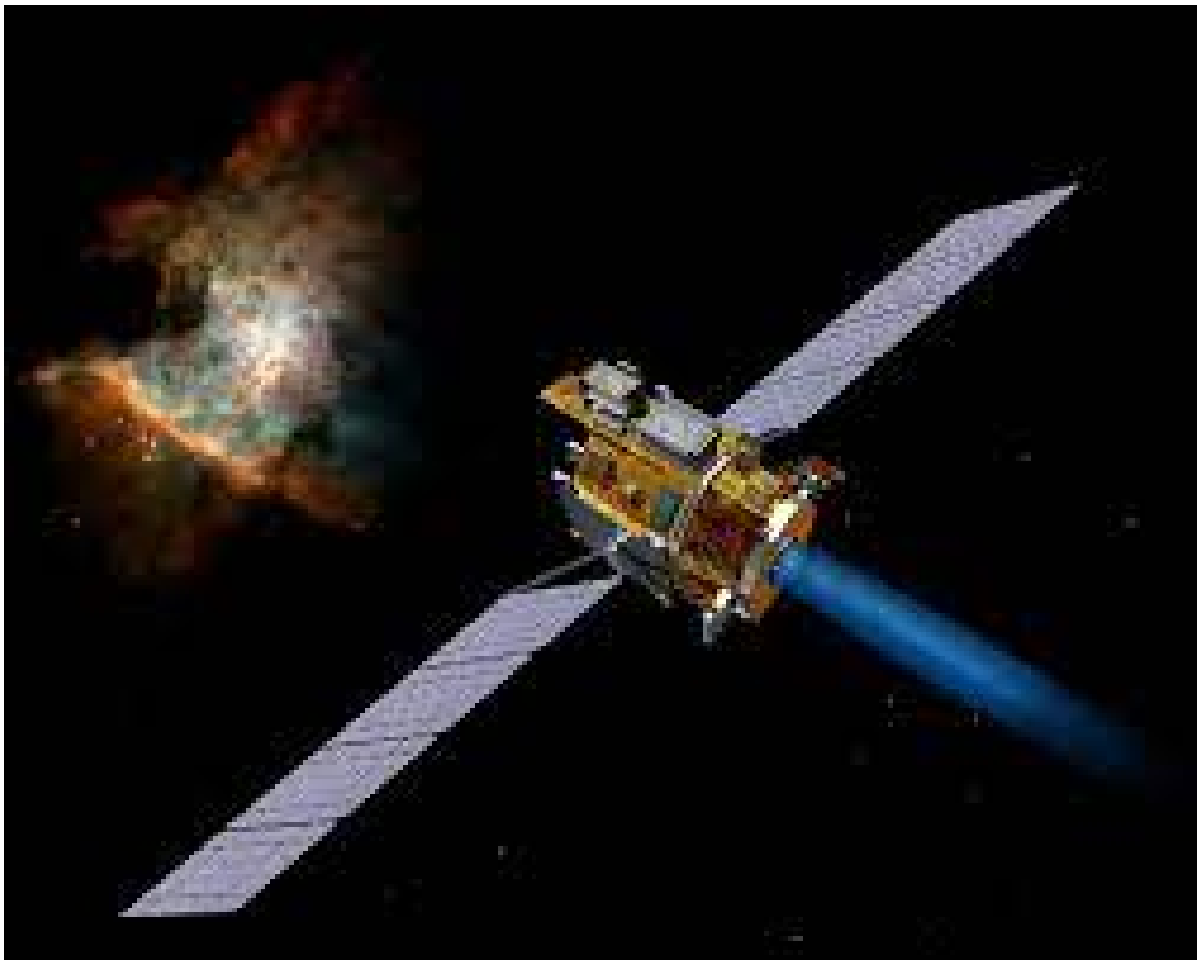
Model Checking with Rulebase in IBM:

- Applied model checking to some extent on approximately 40 design components: the instruction unit, control logic, memory subsystem and I/O chip,...
- More than 200 design flaws at various stages have been found
- At least one bug was found by almost every application of formal verification
- 15% of bugs would have evaded simulation.
- Some of the bugs literally escaped 1-2 years of simulation

Systems Verified

- Various microprocessors (instruction level verification): DLX pipelined architectures, AAMP5 (avionics applications), FM9001 (32 bit processor), PowerPC
- SRT division (Pentium), recent Intel ex-fpu, ADK IEEE multiplier, AMD division Multiprocessor coherence protocols
- SGI, sun S3.Mp architectures, Gigamax, futurebus+ Memory subsystems of PowerPC
- Fairisle ATM switch core

Remote Agent on NASA's Deep Space 1 mission



Remote Agent on NASA's Deep Space 1 mission

Formal methods have been used to verify a component of the Remote Agent software. Remote Agent is the first artificial intelligence control system to control a spacecraft without human supervision.
Deep Space 1 is a spacecraft dedicated to testing high risk technologies to lower the cost and risk to future science-driven missions
The formal specification language of SPIN model checker was used to detect successfully five concurrency errors in the LISP code that have not been discovered by developers during testing.

Figure: Y2K problem, Millennium bug, Y2K bug

Finance- Mondex smart cards

Formal methods have been applied in the development of Mondex, an electronic purse hosted on a smart card. Each card stores financial value (equivalent to cash) as electronic information on a micro chip and provides operations for making financial transactions with other cards via a communication device.
Using formal methods helped to reveal a bug in the implementation of the secondary protocol, which allowed Mondex to get the ITSEC certificate level.
The Z notation was used to prove security properties, which allows Modex to achieve ITSEC level E6, ITSEC's highest granted security-level classification.

More Applications of Formal Verification

Blockchain: *CertiKOS:* The program is the first example of formal verification in the blockchain world, and an example of formal verification being used explicitly as a security program.
CertiK leads blockchain security by pioneering the use of cutting-edge Formal Verification technology on smart contracts and blockchains. Unlike traditional security audits, Formal Verification mathematically proves program correctness and hacker-resistance.
Networking Network software vendors that offer formal verification solutions include Cisco Forward Networks, and Veriflow Systems.
C language The CompCert C compiler is a formally verified
Machine Learning Safety verification and trustworthy of Deep neural networks

Tools Support

Tool Support for Specification Languages

Analysis:

model checking proof checking

Synthesis

refinement code generation test case generation

Success stories - Landing on March

• SPIN previously used in Cassini, Deep Space One and Mars Exploration missions
• MSL mission: 120 parallel tasks under control of real-time operating system, high potential for race conditions
• SPIN + Modex used to verify critical software components: • dual-CPU boot-control algorithm
• the non-volatile flash file system
• the data-management subsystem (the largest one, 45 k lines of code, converted manually to a Spin model of 1.600 lines)
• model-checking performed routinely after every change in the code of the file system, in most cases identified subtle concurrency flaws

SLAM-Microsoft Windows

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification were building tools that can do actual proof about the software and how it works in order to guarantee the reliability." Bill Gates, April 18, 2002. Keynote address at WinHec 2002

Success stories - SOFTWARE SUCCESS STORY

- 85% of system crashes of Windows XP caused by bugs in third-party kernel-level device drivers (2003)
- one of reasons is the complexity of the Windows drivers API
- SLAM: automatically checks device drivers for certain correctness properties with respect to the Windows device drivers API
- now core of Static Driver Verifier, which in turn is a part of Windows Driver Development Kit, a toolset for drivers developers, and integrated into Visual Studio

- An Introduction to Formal Methods for the Development of Safety-critical Applications Haxthausen, Anne Elisabeth, 2010.
- Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal Methods: Practice and Experience. ACM Computing Surveys, 41(4):1–36, October 2009.
- Hardware Verification - Application of formal techniques to chip designs Jacob Abraham (July 7, 2001).