Three decades of success stories in formal methods

Hubert Garavel
INRIA / LIG

with contributions of Susanne Graf
CNRS / Verimag
Context of this work

• A study on formal methods launched in 2011 by the German BSI (federal office on security in information technology)

• Currently, formal methods are not so widely used - mostly in niche applications

• Can we prove to (industrial or governmental) managers that formal methods are useful?
A frequent answer: failure stories

• List of well-known software failures:
  - Therac 25 radiation therapy engine
  - Denver airport
  - Patriot missile interceptor
  - Pentium 5 division algorithm
  - Ariane 5.01 maiden flight
  - Mars orbiter
  - etc.

• But:
  - these are disaster stories
  - they threaten the audience, but do not prove that formal methods would have enabled to avoid such failures
Be positive!

- Try to convince rather than threaten
- Find *success* stories, not disaster stories
Related studies (1)

There already exist formal methods surveys:

- **Craigen, Gerhart & Ralston 1993, 1994, 1995**
  - 12 case studies
  - old and unbalanced: SCR (formulas and tables), B, CAS logic, Gypsy (1st order predicate calculus), Z, Z, Cleanroom, Z, RAISE, Z+Occam, RAISE, VDM
  - nothing about protocol or hardware verification

- **Clarke & Wing 1996**
  - 22 pages
  - rich and dense
  - quite exhaustive and fair, but 16 years-old now
Related studies (2)

- **Woodcock, Larsen, Bicarregui & Fitzgerald 2009**
  - 40 pages
  - rich and deep
  - but a bit biased:
    - towards "US+UK" and "paper and pencil" methods
    - against model checking (admitted)
    - (not a word about the PRISM model checker!)
  - 8 featured projects (Section 4): Z+CSP, B, Z, RAISE+PVS+ACL2, SCADE, Z+SPIN, SPARK, VDM++

- **Haxthausen 2010**
  - 32 pages, a bit superficial, not balanced
  - 6 case studies: B, B, B, RAISE, SPIN, Z
Related studies (3)

- **Dagstuhl 2010 Manifesto**
  - comprehensive panorama
  - no case studies

- **www.fm4industry.org**
  - famous formal methods success stories
  - 8 case studies listed (1980-2009)

- **www.fmsurvey.org**
  - survey on industrial use of formal methods (2011)
  - 62 projects considered
  - 34% agree and 61% strongly agree that using formal methods was successful
A different methodology

• How to ensure:
  - a larger coverage over years?
  - a greater diversity of approaches and methods?
  - a better balance between countries?

• Idea:
  - Review 3 decades (from 1982 to 2011 included)
  - Select one "success story" each year
=> 30 featured case studies
Selection criteria

• Focus on concrete applications of formal methods, rather than theoretical discoveries or software tool releases

• Many applications took several years: the year of first publication is chosen, rather the dates of start or end of the project (often unknown or unclear)

• Avoid selecting the same approach twice
Difficulties (1)

• Exhaustivity is impossible:
  • Only 30 slots for a large formal methods community
  • Google Scholar: **220,000** answers for "formal methods"
  • Nearly **4000** scientists in verification [Woodcock et al]
    - USA : 1000 scientists in verification [Shankar 2009]
    - Europe : 1000
    - Nordic countries : 500
    - China : 250
    - Japan : 250
    - Australia / New Zealand / Brazil / Canada / Singapore / South Africa : 1000
Difficulties (2)

• The problem (finding one single "success story" per year) is perhaps over-constrained

• In recent years, there are several convincing applications of formal methods. No one is really the "best"

• Priority to case studies that were influential and/or reproduced later independently

• Some "tweaking" needed on publication dates (conference vs journal, etc.)
Formal specification, using temporal logic, of asynchronous circuits and sequential circuits
Verification of these circuits using state-space exploration and/or model checking
The EMC model checker revealed an error in a FIFO queue circuit element published in a popular textbook on VLSI design
[1982]


1983

[Billington-Bearman-Wilbur Ham]
[Courtiat-Ayache-Algayres]
[Jürgensen-Vuong]

Formal specifications with (extended Petri nets) of the OSI transport layer protocol

Three independent teams, one using the PROTEAN tool (Austria) and two using the OGIVE/OVIDE tool (France)

Various properties checked (general, specific, structural). No harmful error found
[1983]


[1983]


Automated proof checking using the NQTHM (Boyer-Moore) theorem prover of fundamental theorems of computer science:

- unsolvability of the halting problem,
- Gödel's first incompleteness theorem
- Church-Rosser theorem of λ-calculus and other theorems of practical value:
- invertibility of the RSA encryption algorithm

51. Theorem. CRYPT.INVERTS:
(IMPLIES
  (AND (PRIME P)
    (PRIME Q)
    (NOT (EQUAL P Q))
    (EQUAL N (TIMES P Q))
    (NUMBERP M)
    (LESSP M N)
    (EQUAL (REMAINDER (TIMES E D)
      (TIMES (SUB1 P) (SUB1 Q)))
      1))
  (EQUAL (CRYPT (CRYPT M E N) D N) M))


Formal verification of the 16-bit FM8501 microprocessor using the NQTHM theorem prover

This was the first verified microprocessor, followed by many others


1986

[West] + [Rudin, Zafiropulo]

Formal analysis of the (slightly simplified) OSI session layer protocol, using finite state machines communicating by bounded FIFO queues

Verification using automated protocol validation techniques based on state space exploration.

Various errors found and reported to ISO

Fig. 2. The OSI session layer validation system.
[1986]


1987

[Graf-Richier-Rodriguez-Sifakis-Voiron]

Specification in Estelle/R (a rendezvous-based variant of the protocol description language Estelle) and verification using the Xesar model checker of two protocols:

- a generic sliding window protocol, later intensively studied by the computer-aided verification community under the name "bounded retransmission protocol"

- an atomic multicast protocol for the Delta-4 distributed dependable architecture
[1987]


Formal methods were used to specify OSI (Open System Interconnection) standards in a concise, unambiguous, implementation-neutral way. LOTOS has been used intensively to specify:

- the service and protocol of the session layer
- the service and protocol of the transport layer
- the service and protocol of the network layer
- at the application layer
  - ROSE (Remote Operations Service Element) service
  - CCR (Commitment, Concurrency and Recovery) service and protocol
  - DTP (Distributed Transaction Processing) protocol

[Ajubi-Aujla-Caneschi-Ferreira Pires-Freestone-Scollo-Turner-van de Heijden-van de Lagemaat-van Sinderen-Vissers-Widya]


[1988]


[1988]


1989

[Stålmarck-Säflund-Borälv-Sheeran] [Groote-Koorn-Van Vlijmen] [Fokkink] [Eisner]

Formal verification, using a novel algorithm for efficiently proving large theorems of propositional logic, of safety-critical applications such as

- reverse flushing control in a nuclear plant's emergency cooling system
- landing gear control for a military aircraft
- railway signalling systems (interlocking verification)
[1989]


1990

Formal specification using the B language and correctness proofs using Hoare-like logic (in addition to traditional code inspection and testing approaches) of SACEM, a fault-tolerant railway signalling system that controls train speed, signals drivers, and activates emergency brakes.

SACEM was the first safety-critical software system certified by the French railway authority; it is used in Paris (800,000 passengers carried per day) and other cities in the world.

[Guiho-Hennebert]


Use of Z in two large IBM projects:

- a major new release IBM's CICS (Customer Information Control System) on-line transaction processing system
- the API (Application Programming Interface) of CICS.

Very few tools were used (only syntax and type checkers)

The authors report that the use of Z reduced the number of errors by a factor of 2.5 and saved 9% of the total development cost

But the significance of these conclusions was questioned later [Finney-Fenton]


1992

Formalization, using LOTOS and ACTL logic of the concept of "interactor", a software architectural model used to build complex user interface software

Several applications, e.g., MATIS, a multimodal interactive system enabling users to get information about flight schedules using speech, mouse and keyboard, or a combination of them

[Paterno-Faconti] + [Duke-Harrison-Fornari-Mezzanotte-Sciacchitano-Löwgren] [Markopoulos]


[1992]


Formal specification and verification of the cache coherence protocol of IEEE standard 896.1-1991 "Futurebus+" using the SMV symbolic model checker

Several design errors previously undetected were found

First time that a formal verification tool was used to find errors in an IEEE standard


Applications of the abstract interpretation to build static analyzers for C programs:

- LCLint annotation-assisted static checker (later extended to check dynamic memory allocation and buffer overflow vulnerabilities, and renamed into Splint)

- IABC static analysis tool for pointer manipulation and aliasing, which later went to marked under the name Polyspace Verifier


Discovery, using CSP and FDR of an unknown, subtle "man-in-the-middle" attack in the classical Needham-Schroeder public-key protocol, which forms the basis of Kerberos authentication. This fueled a lot of research on formal methods and tools for the analysis of security protocols.


Specification using Z and Promela, and model checking using SPIN of the software controlling the storm surge barrier that protects Rotterdam from flooding, a life-critical application certified at the highest safety integrity level (SIL4)
[1996]


1997

[Luttik]
[Kühne-Hooman-de Roever]
[Sighireanu-Mateescu]

Specification and analysis, using various formal methods, of the asynchronous mode of the Link Layer protocol of the IEEE Standard 1394 "Firewire" high-speed serial bus. Two problems found:

- a missing handling of pending requests discovered independently using PVS and μCRL
- a deadlock discovered using LOTOS and CADP in only one person.month without prior knowledge of the protocol

During the next decade, other IEEE 1394 protocols (root contention, tree identity, leader election, etc.) have been intensely scrutinized.
[1997]


1998

[Bowman-Facconti-Katoen-Latella-Massink]
[Lindahl-Pettersson-Yi]
[Tripakis-Yovine]

Automated verification of several real-time protocols using:
- Kronos [Daws-Olivero-Tripakis-Yovine-Bozga-Maler]
- Uppaal [Bengtsson-Larsen-Larsson-Pettersson-Yi]
[1998]


[LPY98] Magnus Lindahl, Paul Pettersson, and Wang Yi. Formal Design and Analysis of a Gear Controller. In Bernhard Steffen, editor,


TTA (Time-Triggered Architecture) is a communication bus infrastructure guaranteeing dependability, predictability, and real-time requirements [Kopetz-Bauer-Braun-Gründsteidl-etc]

TTA and similar architectures are used for distributed-control safety-critical applications in automotive, aerospace, railways, industrial automation and process control, medical systems, etc.

Formal verification using PVS of several key protocols of TTA
[1999]


Automated validation of several randomized distributed algorithms (taken from the literature) using the PRISM probabilistic model checker

PRISM has been used to analyze case studies in many different application domains
[2000]


Development of a verification platform (based on static analysis and symbolic model checking) for analyzing the source code of Microsoft Windows drivers (and more generally any source C code)

Check whether the invocations of API (Application Programming Interfaces) primitives obey rules for proper use


Automated analysis of Lucent's CDMA base station call-processing software library (100,000s lines of C/C++ code) using the VeriSoft tool for systematic state space exploration

Detection of several critical bugs


Formal proof using the ACL2 theorem prover that the microcode of the Rockwell Collins AAMP7 microprocessor respects a security policy corresponding to a static separation kernel.

The microprocessor received a MILS Certificate from NSA to concurrently process Unclassified through Top Secret codeword information.


Proof, using the Astrée static analyzer based on abstract interpretation, of the absence of any run-time error in several safety-critical C programs of Airbus:

- primary flight-control software for the A340 fly-by-wire system
- electric flight-control codes for the A380 series


2005

Computer-checked proof using Coq of the "four color theorem", the second most famous unsolved problem in mathematics
[2005]


Formal verification using Coq of a C compiler (front-end and back-end) with a realistic subset of the C language usable for critical embedded software
[2006]


Design, validation, and implementation of avionics, automotive, railway, and other safety-critical applications using the SCADE tools for the synchronous language Lustre


Formal verification using JML and ESC/Java in the vote-tallying part of the KOA open source software used for remote voting in Dutch public elections: discovery of specification errors and programming bugs undetected so far
2009

[Platzer-Clarke]

Formal verification of curved flight collision avoidance maneuvers using the KeYmaera verification tool for hybrid systems

Figure 8: Flyable aircraft roundabout (multiple aircraft)

Formal verification of the seL4 general purpose operating-system micro-kernel using the Isabelle/HOL theorer prover


Formal modelling of the EMV (Europay-MasterCard-Visa) protocol suite in the F# language

Automated analysis of these protocols by joint use of:
- FS2PV translator [Barghavan-Fournet-Gordon-Tse]
- ProVerif [Blanchet]
[2011]


Conclusion
Conclusion

• **About formal methods:**
  - Much has been done in 30 years
  - Great diversity of applications
  - Key issue: how to incorporate formal methods in standard engineering practice?

• **About this study:**
  - A careful selection of success stories
  - Not limitative: there are more success stories
  - Perhaps biased towards over-published works
  - Other lists could be made:
    - 30 most useful fundamental results
    - 30 best software tools