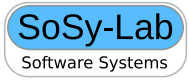
Formale Spezifikation und Verifikation

Wintersemester 2024

Prof. Dr. Gidon Ernst
gidon.ernst@lmu.de

Software and Computational Systems Lab
Ludwig-Maximilians-Universität München, Germany

September 30, 2024



Heute

- Formale Methoden in der Praxis:
  Erfolgreiche Projekte, Techniken, Tools aus Industrie und Forschung

- Fragestunde, Diskussion (sofern Zeit)

- Evaluation: Danke fürs Mitmachen
  - ✓ spannende Inhalte, Begeisterung für das Thema
  - ✓ Hybrid-Angebot
  - ✓ Interaktiv, praktische Beispiele
  - ? klarere Struktur, teilweise zu viel/schnell/anspruchsvoll
  - ? mehr Übung, z.B. Altklausuren (teilweise schon umgesetzt)
  - ? mehr Hintergrundmaterial (z.B. Skript), weniger Handschrift (insbes. Bsp)
  - ? Termin ungünstig für Mittagessen
  - ✗ Projektkorrektur dauert zu lange (wir stimmen zu)

Formale Methoden in der Praxis

Teaser [Woodcock et al. 2009]

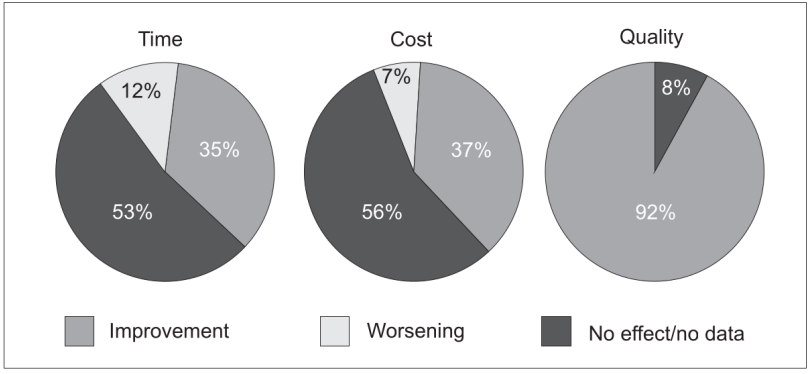


Fig. 6. Did the use of formal techniques have an effect on time, cost, and quality?

✓ Höhere Qualität ✓ Niedrigere Kosten

Teaser [Gleirscher, Marmsoler 2020]

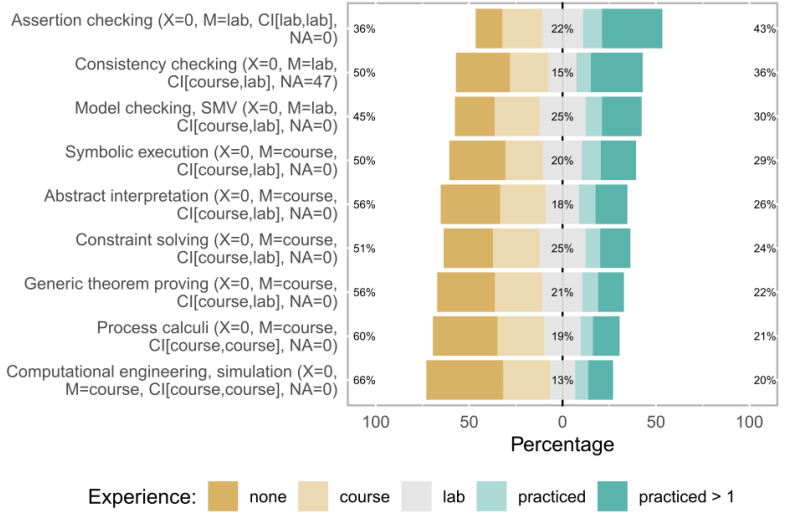


Fig. 7 (Q6) Describe your level of experience with each of the following classes of formal reasoning techniques

Führerlose Paris Métro 14

✓ Ada (150kLoC) + B-Method:
formale Verifikation von Code und der Streckendaten

seL4 - ein verifizierter OS Kernel [Klein et al. 2009]

Entwicklung

- Prototyp der Funktionalität in Haskell
- Realistische Implementierung in C
- Abstrakte Spezifikation
- Fokus: Verwaltung von Prozessen, Kommunikation, Speicher

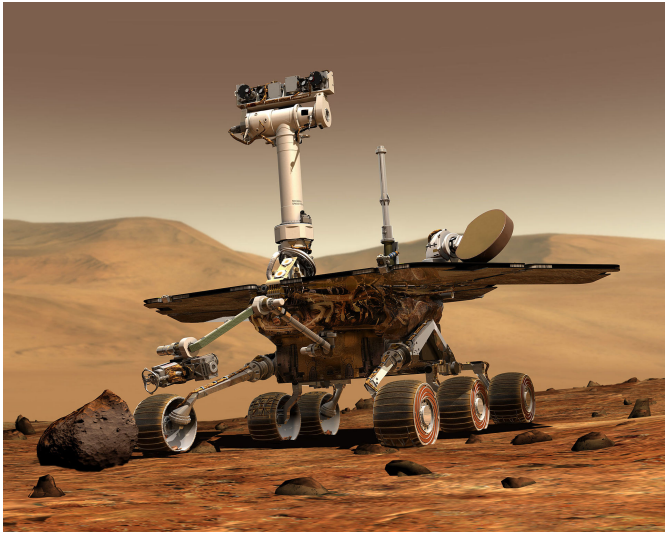
Formale Beweise

- > 20 Personenjahre Arbeit, Tool: Isabelle
- Zusätzlich: Informationssicherheit!
- seL4 Foundation mit zahlreichen Industriellen Partnern

seL4 - Darpa HACMS (High Assurance Cyber Military Systems)

Autonomer Heli: ✓ seL4-basierte Sicherheitsarchitektur widersteht Pentests

Flash File System Challenge [Joshi, 2007]



Flash File System Challenge [Joshi, 2007]

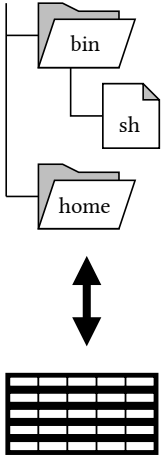
1) Flash Speicher

- kein Überschreiben
- beschränkte Lebensdauer
- unzuverlässig

2) Stromausfall

- Wie formal beschreiben?
- Beweisansatz?

3) Abstraktion

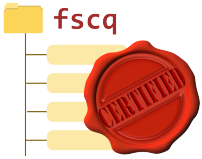


Verifikation von Dateisystemen

Flashix

- Flash-Speicher
- Tool: KIV, abstrakte Programme
- Übersetzung nach C (unverifiziert)

FSCQ



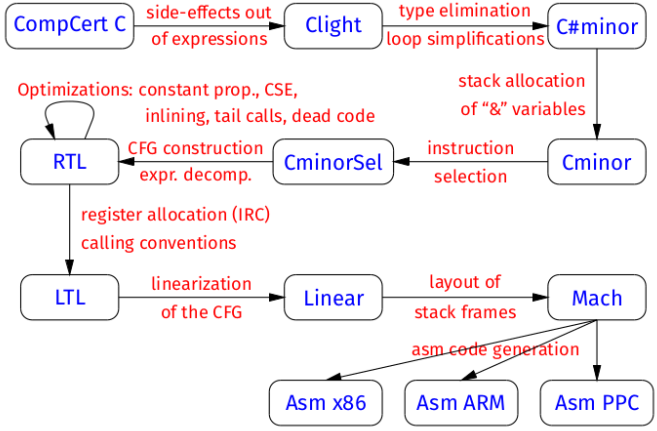
- allgemeine Speichermedien
- Tool: Coq
- Extraktion nach Ocaml/Haskell

Yggdrasil

- allgemeine Speichermedien
- Tool: Rosette/SMT (hohe Automatisierung)
- Kompromiss: obere Größe auf alle Daten

✓ alle drei Dateisysteme sind bewiesenermaßen *funktional korrekt* und *robust bei Stromausfall* → kein `fschk` notwendig!

CompCert [Leroy, 2009]



Vollständig verifizierter C Compiler

✓ Generiert z.T. *effizienteren* Code als Konkurrenz

AbsInt - Zertifizierung

	WCET analysis		Stack usage analysis	
	generic	compiler-specific	generic	compiler-specific
ARM	✓	TI 4.9.1 TI 20.2.1.LTS GCC 4.9.3 Keil 5.02.0.28	✓	TI 4.9.1 GCC 4.7.4 GCC 4.9.3 GCC 4.9.4 GHS 2019.1.4 Keil 5.02.0.28 Keil 3.1.0.939
C16x	✓	Tasking 8.8r1 Tasking VX 2.4r1	✓	Tasking VX 2.4r1 Tasking VX 3.1r2
C28x	✓	✗	✗	✗
dsPIC	✗	✗	✓	✗
i386	✓	✗	✗	✗
LEON3	✓	GCC 3.4.4	✗	✗
M68020	✓	GCC 4.2.1	✓	GCC 4.2.1

https://www.absint.com/qualification/

Studien zu Rust: PBT, symbolische Suche

Rust testing or verifying: Why not both?

Dijkstra famously dissed testing by saying “Program testing can be used to show the presence of bugs, but never to show their absence!” As if you should choose one over the other. I don’t see them as opposites but as complementary techniques that should both be used to improve the quality of your code.

READ MORE

✓ Klee symbolic execution + ✓ Rust Property-Based Testing framework

https://alastairreid.github.io/why-not-both/

Google Project Zero

Wednesday, December 1, 2021

This shouldn't have happened: A vulnerability postmortem

Posted by Tavis Ormandy, Project Zero

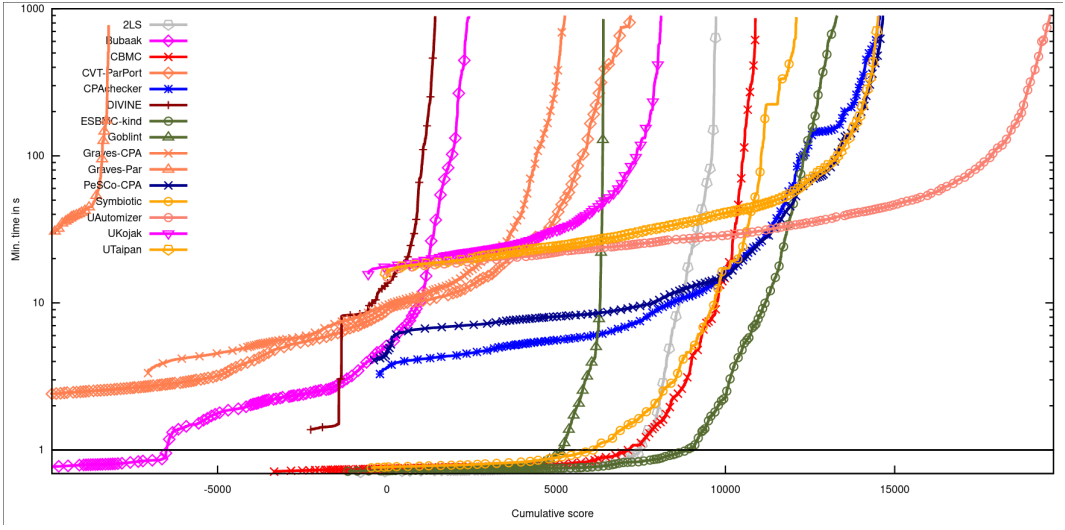
Introduction

This is an unusual blog post. I normally write posts to highlight some hidden attack surface or interesting complex vulnerability class. This time, I want to talk about a vulnerability that is neither of those things. The striking thing about this vulnerability is just how simple it is. This should have been caught earlier, and I want to explore why that didn't happen.

https://googleprojectzero.blogspot.com/2021/12/this-shouldnt-have-happened.html

✓ Random/Fuzz Testing at scale → viele Sicherheitslücken gefunden!

Wettbewerbe: SV-COMP 2023

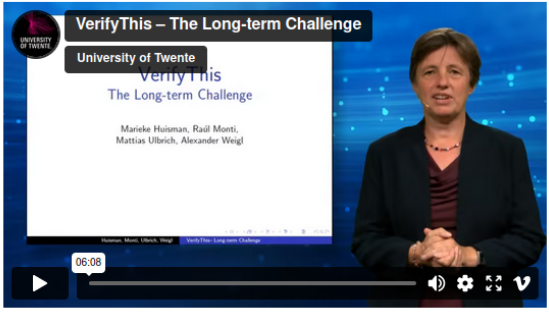


Vollautomatische Tools: CPAchecker, CBMC, Ultimate Automtizer, ...

Wettbewerbe: VerifyThis

What is VerifyThis?

The VerifyThis verification competition is a regular event run as an onsite meeting with workshop character in which three challenges are proposed that participants have to verify in 90 minutes each using their favourite program verification tool.



Interaktive Tools: Dafny, Why3, VeriFast, Viper, KIV, ...

Formale Methoden in der Industrie

- Microsoft: Z3, Dafny, Lean, F*, viele Projekte
- Amazon s2n (TLS 1.3 Protokoll), Cloud and Network Security
- Facebook: Infer; Google: Project Zero
- Intel: Model-Checking, HOL Light
- Viele kleinere Player: Galois, AbsInt, DiffBlue, Ada Core, Proofcraft
- Event-B, B-Method: Railway
- VDM: Pasmo card

Facebook: Infer [O'Hearn et al 2018]

Tool: Infer https://fbinfer.com/
Erfolgreicher Transfer aktueller Forschung in die Praxis

- Separation Logic [Reynolds 2005]: geschickter Umgang mit verketteten Datenstrukturen
- Incorrectness Logic [deVries, Koutavas 2013; O'Hearn 2018]: Statische Analyse ohne False Positives

Integration in die Entwicklungspraxis

- automatisches Code-Review beim commit
- schnelles Feedback! $\rightarrow$ incrementelle Anaslyse
- sehr gut angenommen durch Entwickler

Demo: Verifizerite Programmierung

https://wiki.portal.chalmers.se/agda/pmwiki.php

Formalisierung von Mathematik mit Lean

Way to the future

What are the limiting factors for a wider adoption of formal methods by industry?	
Engineers lack proper training in formal methods	71.5%
Academic tools have limitations and are not professionally maintained	66.9%
Formal methods are not properly integrated in the industrial design life cycle	66.9%
Formal methods have a steep learning curve	63.8%
Developers are reluctant to change their way of working	62.3%
Managers are not aware of formal methods	57.7%
Many companies do not pay enough attention to software quality and security	56.2%
Formal methods are not properly integrated in standard certification processes	46.9%
Formal methods focus on relevant problems, but only on a small part of all problems	36.9%
Benefits of formal methods are not marketed enough	36.9%
There are too many formal methods, with a lack of independent comparison	28.5%
Formal methods are too costly, with no perceived immediate added value	26.9%
Formal methods are too slow to meet current time-to-market constraints	17.7%
Professional tools are too expensive because of the small market for them	14.6%
Other approaches to software quality outperform formal methods	13.1%
Industrial software development practices change too often and too quickly	8.5%
Formal methods focus on the wrong problems	7.7%
Other	13.8%

Fazit

Formale Methoden $\neq$ “nur Korrektheitsbeweise”

- ✓ Viele interessante, neue Anwendungen
- ✓ Mächtige Tools, insbesondere SMT
- ✓ Industrie: Skalierbarkeit wird endlich realisiert

License