Modelling and Validation of Concurrent System

António Ravara

May 6, 2024

What drives my research?

How can we (coders) make programs go right?

What drives my research?

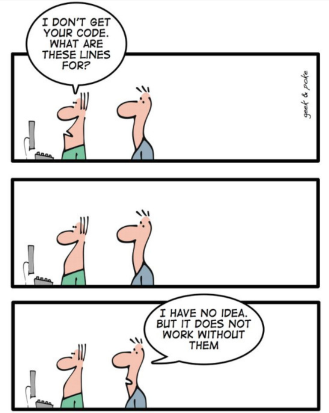
How can we (coders) make programs go right?

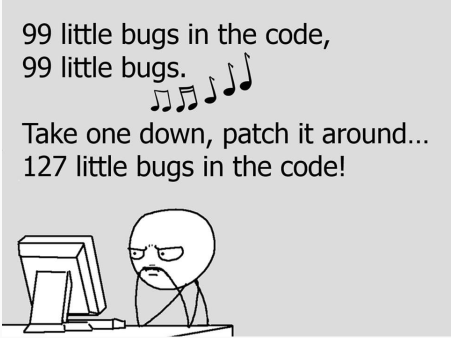
Let's take steps to avoid horror stories

What drives my research?

How can we (coders) make programs go right?

Let's take steps to avoid horror stories





Scenario under consideration in this course

Modern Software Systems

- Pervasive – our society fundamentally depends on them.

Scenario under consideration in this course

Modern Software Systems

- Pervasive – our society fundamentally depends on them.
- Crucial ones are huge:
  airspace, banking, taxes, telecoms, ...

Scenario under consideration in this course

Modern Software Systems

- Pervasive – our society fundamentally depends on them.
- Crucial ones are huge:
  airspace, banking, taxes, telecoms, ...
- Main characteristics:
  large-scale, distributed, communication-intensive,
  hold critical data.

Scenario under consideration in this course

Modern Software Systems

- Pervasive – our society fundamentally depends on them.
- Crucial ones are huge:
  airspace, banking, taxes, telecoms, ...
- Main characteristics:
  large-scale, distributed, communication-intensive,
  hold critical data.

  In short:
  no room to failures!

Problems/Challenges with modern software systems

- Difficult to build and to maintain.
  Blueprints? Development methodology? Updates?

Problems/Challenges with modern software systems

- Difficult to build and to maintain.
  Blueprints? Development methodology? Updates?
- How to express their goals and intended functionality?
  Requirements? Specifications?

Problems/Challenges with modern software systems

- Difficult to build and to maintain.
  Blueprints? Development methodology? Updates?
- How to express their goals and intended functionality?
  Requirements? Specifications?
- How to ensure them correct?
  Do they do what they are supposed to? Will they crash?

Problems/Challenges with modern software systems

- Difficult to build and to maintain.
  Blueprints? Development methodology? Updates?
- How to express their goals and intended functionality?
  Requirements? Specifications?
- How to ensure them correct?
  Do they do what they are supposed to? Will they crash?
- How to keep them safe?
  Are they hackable?

Modelling and validating (large) software systems

Modelling challenges
Informal lists of requirements in natural language are not enough.

Modelling and validating (large) software systems

Modelling challenges
Informal lists of requirements in natural language are not enough.

Validation challenges
Testing/Debugging is not enough.

Modelling and validating (large) software systems

Modelling challenges
Informal lists of requirements in natural language are not enough.

Validation challenges
Testing/Debugging is not enough.

Viable approach
Mathematical tools:

- represent rigorously the intended behaviour
- allow to formally verify correctness.

Dijkstra –'72 Turing award: "The humble programmer"

"Program testing can be used to show the presence of bugs, but never to show their absence!"

Are formal tools really needed? What can go wrong...

One striking example

Joshua Bloch, Google Research Blog (2006):
"Nearly All Binary Searches and Mergesorts are Broken"

Are formal tools really needed? What can go wrong...

One striking example

Joshua Bloch, Google Research Blog (2006):
"Nearly All Binary Searches and Mergesorts are Broken"

- A 9 years old bug on binary search in the standard Java Library
- A clear presentation on how to implement:
  Jon Bentley - Programming Pearls. 1986 (2nd ed. 2000)
  The challenge of binary search

Are formal tools really needed? What can go wrong...

One striking example

Joshua Bloch, Google Research Blog (2006):
"Nearly All Binary Searches and Mergesorts are Broken"

- A 9 years old bug on binary search in the standard Java Library
- A clear presentation on how to implement:
  Jon Bentley - Programming Pearls. 1986 (2nd ed. 2000)
  The challenge of binary search

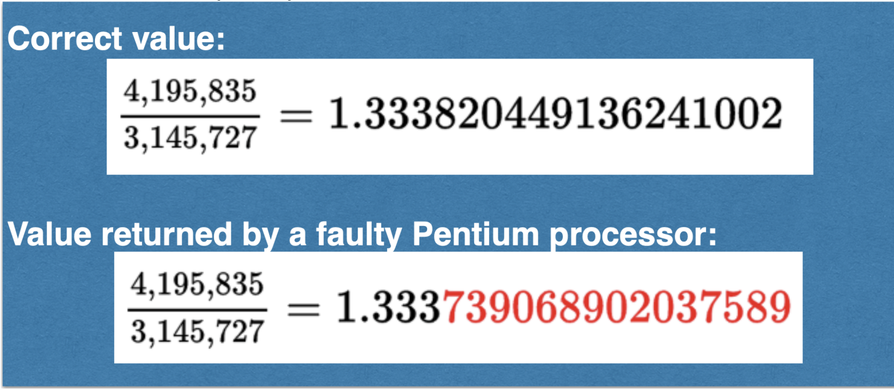
How to avoid these kind of problems:

Hacker-Proof Coding:
`https://www.wired.com/2016/09/computer-scientists-close-perfect-hack-proof-code/`

`https://cacm.acm.org/magazines/2017/8/219596-hacker-proof-coding/fulltext`

Are formal tools really needed? What can go wrong...

Pentium 5 bug (1994): rounding error



https://en.wikipedia.org/wiki/Pentium_FDIV_bug

Formal tools at work

Success stories

- Hardware verification at Intel
  https://www.quora.com/
  What-does-a-SOC-verification-engineer-at-Intel-or-AMD-a

Formal tools at work

Success stories

- Hardware verification at Intel
  `https://www.quora.com/`
  `What-does-a-SOC-verification-engineer-at-Intel-or-AMD-a`
- Driver verification at Microsoft
  `https://msdn.microsoft.com/en-us/library/windows/`
  `hardware/ff552806(v=vs.85).aspx`

Formal tools at work

Success stories

- Hardware verification at Intel
  `https://www.quora.com/What-does-a-SOC-verification-engineer-at-Intel-or-AMD-a`
- Driver verification at Microsoft
  `https://msdn.microsoft.com/en-us/library/windows/hardware/ff552806(v=vs.85).aspx`
- Specification logic (TLA) at Amazon
  `http://cacm.acm.org/magazines/2015/4/184701-how-amazon-web-services-uses-formal-methods/fulltext`

Formal tools at work

Success stories

- Hardware verification at Intel
  https://www.quora.com/
  What-does-a-SOC-verification-engineer-at-Intel-or-AMD-a
- Driver verification at Microsoft
  https://msdn.microsoft.com/en-us/library/windows/
  hardware/ff552806(v=vs.85).aspx
- Specification logic (TLA) at Amazon
  http://cacm.acm.org/magazines/2015/4/
  184701-how-amazon-web-services-uses-formal-methods/
  fulltext
- Code verification at Facebook
  http://fbinfer.com

Plan of the course

Today: concurrent reactive systems

- requirements to model concurrent reactive systems
- the calculus of communicating systems (CCS)

Plan of the course

Today: concurrent reactive systems

- requirements to model concurrent reactive systems
- the calculus of communicating systems (CCS)

Tomorrow: equivalences for CCS

- requirements for an equivalence notion
- observational behavioural equivalence

Plan of the course

Today: concurrent reactive systems

- requirements to model concurrent reactive systems
- the calculus of communicating systems (CCS)

Tomorrow: equivalences for CCS

- requirements for an equivalence notion
- observational behavioural equivalence

Wednesday: dynamic communication topologies

- the pi-calculus (syntax and operational semantics)
- observational behavioural equivalences

Plan of the course

Today: concurrent reactive systems

- requirements to model concurrent reactive systems
- the calculus of communicating systems (CCS)

Tomorrow: equivalences for CCS

- requirements for an equivalence notion
- observational behavioural equivalence

Wednesday: dynamic communication topologies

- the pi-calculus (syntax and operational semantics)
- observational behavioural equivalences

Thursday: logics–theory and tools

Plan of the course

Today: concurrent reactive systems

- requirements to model concurrent reactive systems
- the calculus of communicating systems (CCS)

Tomorrow: equivalences for CCS

- requirements for an equivalence notion
- observational behavioural equivalence

Wednesday: dynamic communication topologies

- the pi-calculus (syntax and operational semantics)
- observational behavioural equivalences

Thursday: logics–theory and tools

Friday (optional): Research problems

Bibliography and resources

- R. Milner: Communication and concurrency. Prentice Hall 1989
- R. Milner: Communicating and mobile systems - the Pi-calculus. Cambridge University Press 1999
- L. Aceto, A. Ingólfsdóttir, K. Larsen, J. Srba: Reactive systems: modelling, specification and verification. Cambridge University Press 2007
- C. Stirling: Modal and Temporal Properties of Processes. Texts in Computer Science, Springer 2001
- J. C. Bradfield, C. Stirling: Modal mu-calculi. Handbook of Modal Logic 2007: 721-756

Bibliography and resources

- R. Milner: Communication and concurrency. Prentice Hall 1989
- R. Milner: Communicating and mobile systems - the Pi-calculus. Cambridge University Press 1999
- L. Aceto, A. Ingólfsdóttir, K. Larsen, J. Srba: Reactive systems: modelling, specification and verification. Cambridge University Press 2007
- C. Stirling: Modal and Temporal Properties of Processes. Texts in Computer Science, Springer 2001
- J. C. Bradfield, C. Stirling: Modal mu-calculi. Handbook of Modal Logic 2007: 721-756

Slides and exercises

`block`