Static analysis and software verification

Vincenzo Arceri - University of Parma - vincenzo.arceri@unipr.it

Expectations?

Short Bio

- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript

Short Bio

- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript
- 2016: Master's Degree in Computer Science (University of Verona)
  - Static analysis of PHP

Short Bio

- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript
- 2016: Master's Degree in Computer Science (University of Verona)
  - Static analysis of PHP
- 2016-2019: PhD in Computer Science (University of Verona)
  - Static analysis of self-modifying code in JavaScript

Short Bio

- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript
- 2016: Master's Degree in Computer Science (University of Verona)
  - Static analysis of PHP
- 2016-2019: PhD in Computer Science (University of Verona)
  - Static analysis of self-modifying code in JavaScript
- 2019-2021: Post-doc researcher (Ca' Foscari University of Venezia)
  - String static analysis in dynamic languages & blockchain smart contracts

Short Bio

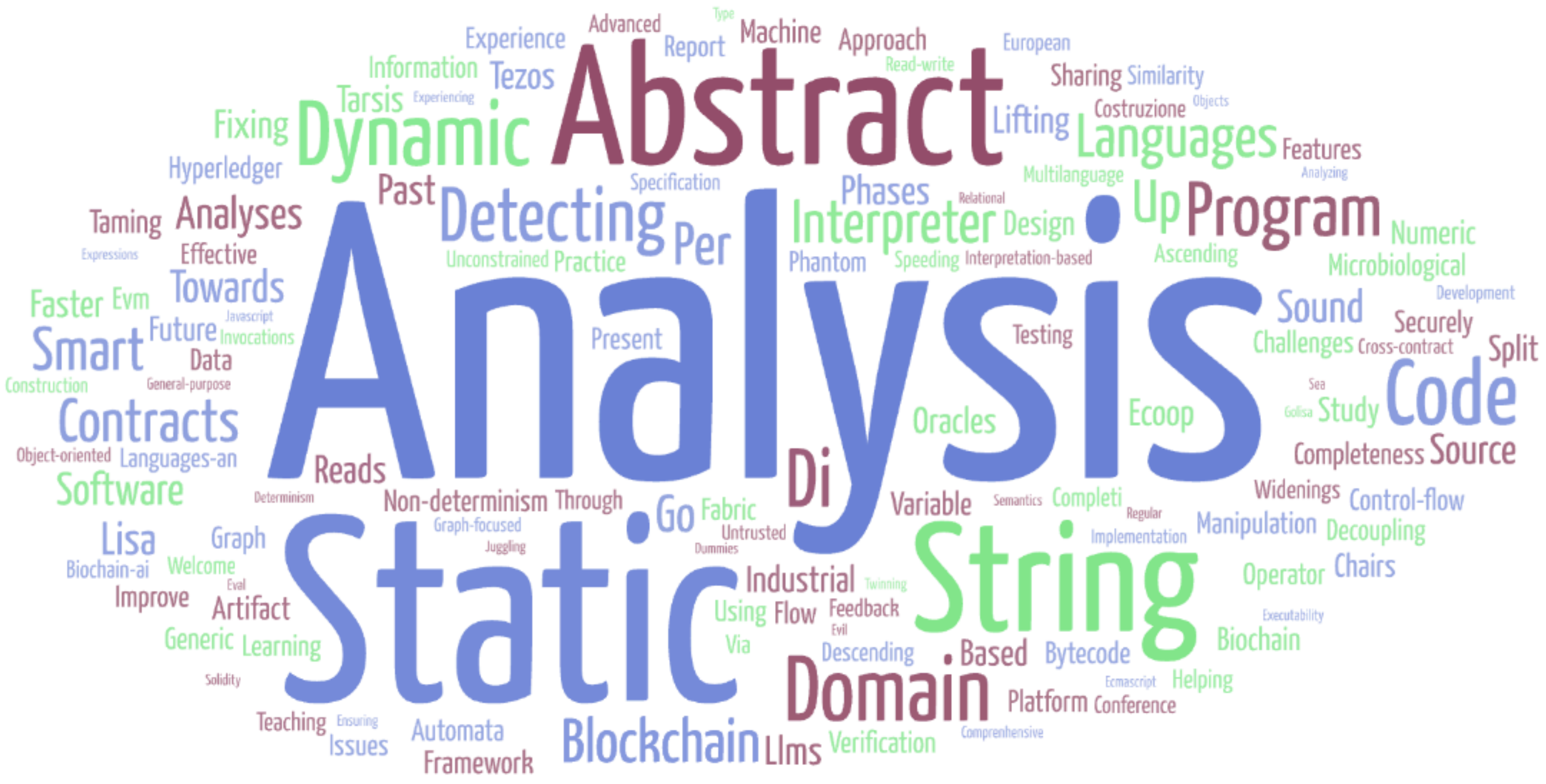
- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript
- 2016: Master's Degree in Computer Science (University of Verona)
  - Static analysis of PHP
- 2016-2019: PhD in Computer Science (University of Verona)
  - Static analysis of self-modifying code in JavaScript
- 2019-2021: Post-doc researcher (Ca' Foscari University of Venezia)
  - String static analysis in dynamic languages & blockchain smart contracts
- 2021-…: Assistant Professor - (University of Parma)
  - Static analysis of blockchain software
  - Static analysis of dynamic languages
  - Static analysis of data science software

Short Bio

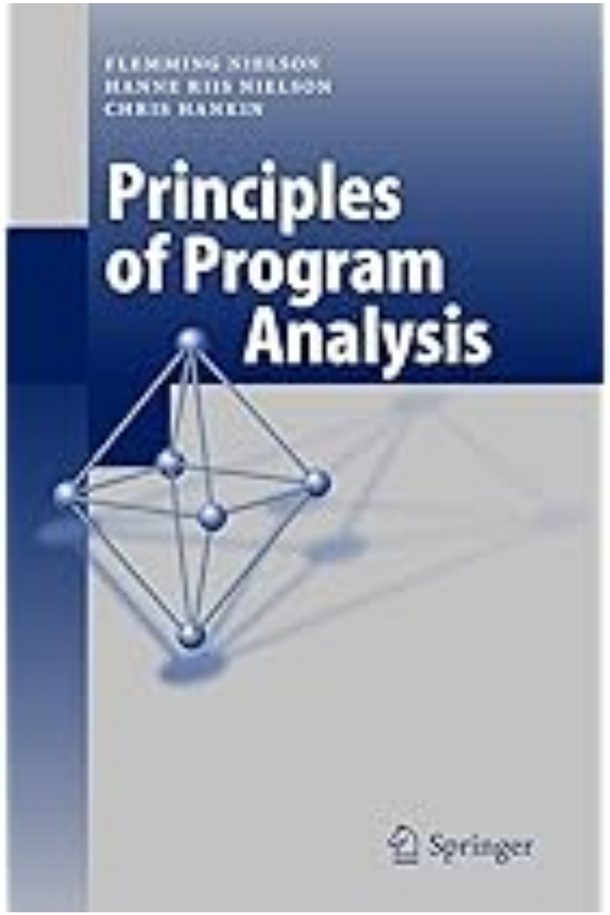
- 2014: Bachelor's Degree in Computer Science (University of Verona)
  - Static analysis of JavaScript
- 2016: Master's Degree in Computer Science (University of Verona)
  - Static analysis of PHP
- 2016-2019: PhD in Computer Science (University of Verona)
  - Static analysis of self-modifying code in JavaScript
- 2019-2021: Post-doc researcher (Ca' Foscari University of Venezia)
  - String static analysis in dynamic languages & blockchain smart contracts
- 2021-…: Assistant Professor - (University of Parma)
  - Static analysis of blockchain software
  - Static analysis of dynamic languages
  - Static analysis of data science software

Recent publications

Keywords


Analysis
Static
Abstract
Dynamic
String
Domain
Detecting
Interpreter
Program
Languages
Code
Contracts
Smart
Blockchain
Experience
Advanced
Type
Machine
Approach
Report
European
Information
Tezos
Read-write
Sharing
Similarity
Tarsis
Experiencing
Costruzione
Objects
Fixing
Lifting
Features
Hyperledger
Specification
Multilanguage
Analyzing
Past
Phases
Relational
Taming
Analyses
Per
Design
Up
Numeric
Expressions
Effective
Unconstrained
Practice
Phantom
Speeding
Interpretation-based
Ascending
Microbiological
Faster
Evm
Towards
Development
Javascript
Sound
Future
Invocations
Present
Testing
Securely
Data
Challenges
Cross-contract
Split
Construction
General-purpose
Sea
Golisa
Study
Oracles
Ecoop
Object-oriented
Languages-an
Reads
Completeness
Source
Software
Di
Widenings
Determinism
Non-determinism
Through
Variable
Semantics
Completi
Control-flow
Regular
Go
Fabric
Manipulation
Decoupling
Lisa
Graph
Graph-focused
Untrusted
Implementation
Biochain-ai
Welcome
Juggling
Dummies
Operator
Chairs
Eval
Industrial
Twinning
Improve
Artifact
Executability
Using
Flow
Feedback
Generic
Learning
Evil
Biochain
Via
Descending
Based
Bytecode
Solidity
Ecmascript
Helping
Platform
Conference
Teaching
Ensuring
Automata
Issues
Comprehensive
Framework
Llms
Verification

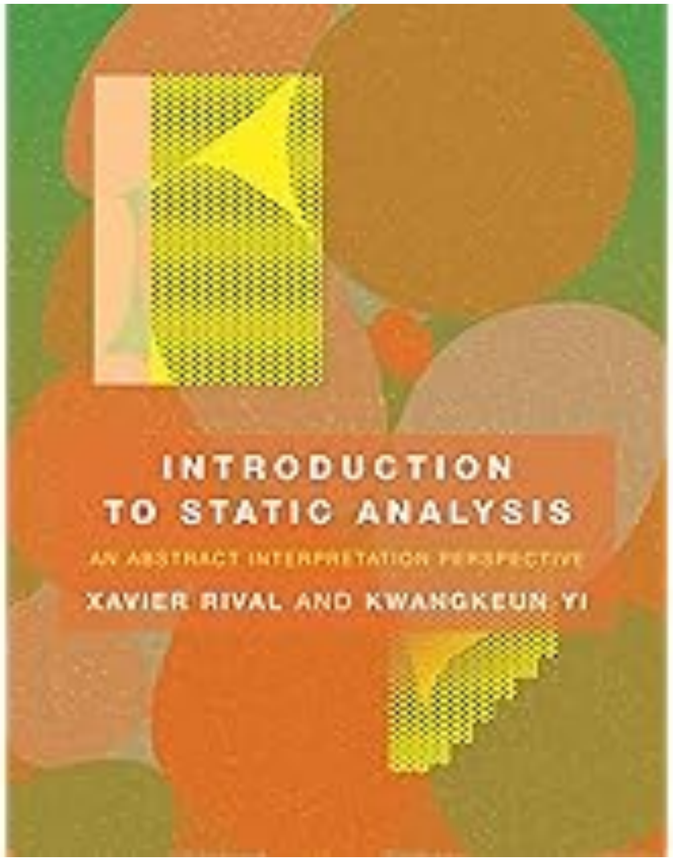
Books



- Principles of Program Analysis, by Flemming Nelson, Hanne Riis Nielson, Chris Hankin, Springer 2004

- This book is unique in providing an overview of the four major approaches to program analysis: data flow analysis, constraint-based analysis, abstract interpretation, and type and effect systems.

Books



- Introduction to Static Analysis: An Abstract Interpretation Perspective, by Xavier Rival and Kwangkeun Yi, MIT Press 2020

- A self-contained introduction to abstract interpretation-based static analysis, an essential resource for students, developers, and users

Books

- Principles of Abstract Interpretation, by Patrick Cousot, MIT Press 2021

- The bible. The book covers all necessary computer science and mathematical concepts related to abstract interpretation, serving as an introduction to abstract interpretation, with examples of applications to the semantics, specification, verification, and static analysis of computer programs.

What is Software Reliability?

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time”

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time"
- IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance."

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time"
- IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance."
- Using these definitions, software reliability consists of three activities:

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time"
- IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance."
- Using these definitions, software reliability consists of three activities:
  - Error prevention

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time"
- IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance."
- Using these definitions, software reliability consists of three activities:
  - Error prevention
  - Fault detection and removal

What is Software Reliability?

- IEEE 610.12-1990 defines *reliability* as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time"
- IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance."
- Using these definitions, software reliability consists of three activities:
  - Error prevention
  - Fault detection and removal
  - Measurements to maximize reliability, specifically measures that support the first two activities

Why is Software Verification important?

Why is Software Verification important?

- Software bugs cost the US economy around $60 billion each year (0.6% of the GDP)

Why is Software Verification important?

- Software bugs cost the US economy around $60 billion each year (0.6% of the GDP)
- Security is a necessity
  - The worldwide economic loss causes by all forms of overt attack is about $250 billion

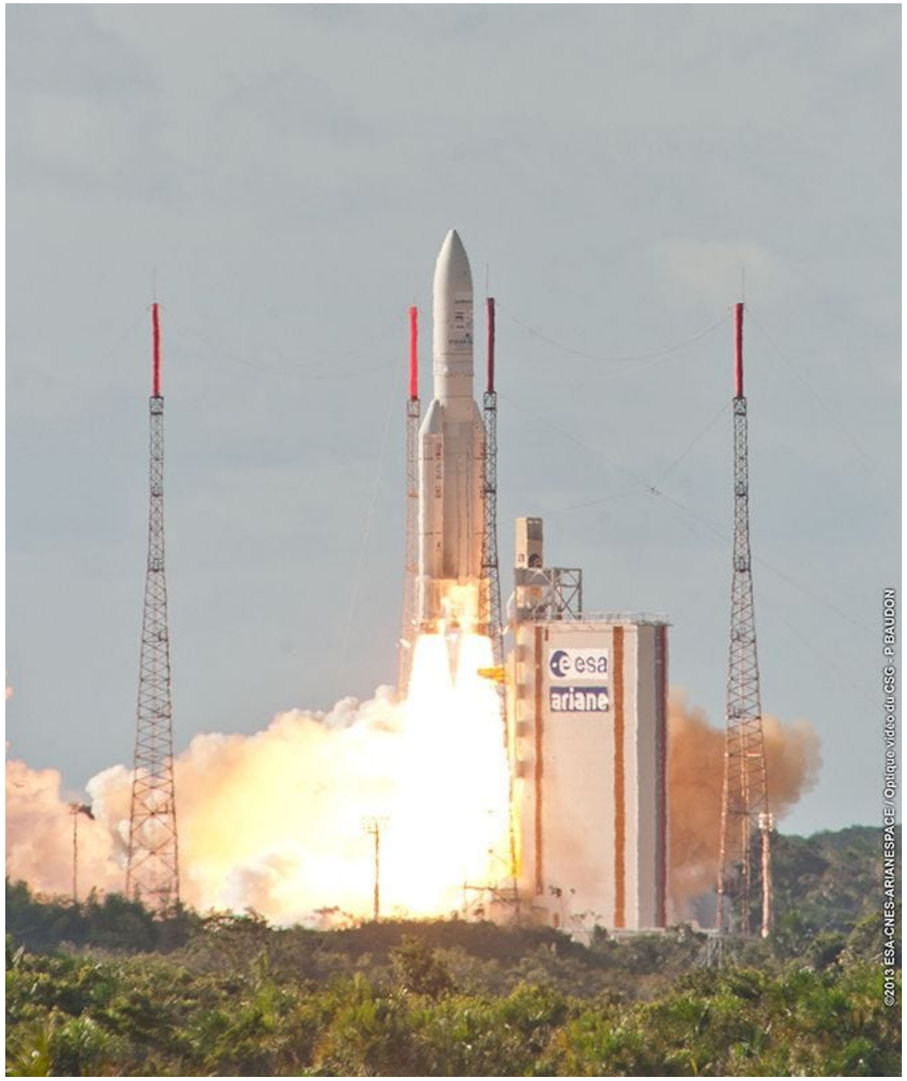
Why is Software Verification important?

- Software bugs cost the US economy around $60 billion each year (0.6% of the GDP)
- Security is a necessity
  - The worldwide economic loss causes by all forms of overt attack is about $250 billion
- Software defects make programming so painful

Why is Software Verification important?

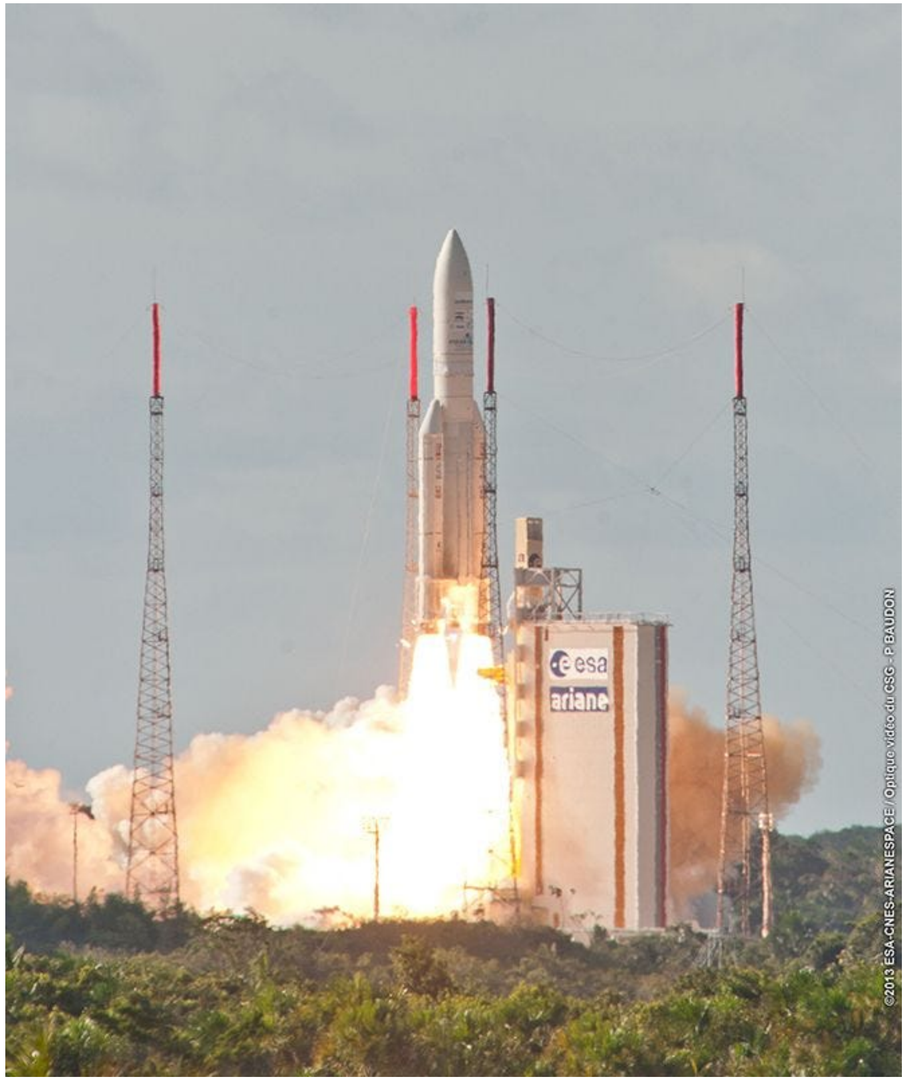
- Software bugs cost the US economy around $60 billion each year (0.6% of the GDP)
- Security is a necessity
  - The worldwide economic loss causes by all forms of overt attack is about $250 billion
- Software defects make programming so painful
- Some stories…

Ariane 5 Flight 501 (1996)



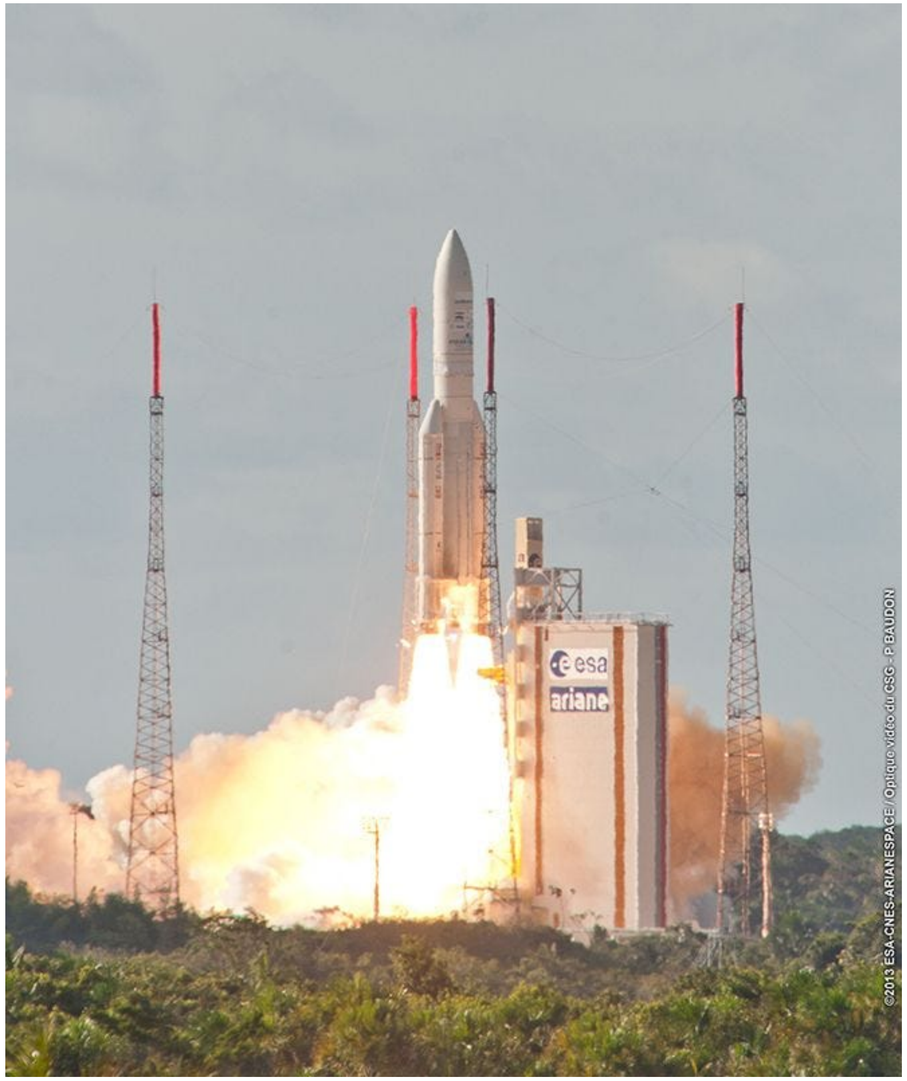
- Exploded 37 seconds after takeoff

Ariane 5 Flight 501 (1996)



- Exploded 37 seconds after takeoff
- Self-destructed due to a bug

Ariane 5 Flight 501 (1996)



- Exploded 37 seconds after takeoff
- Self-destructed due to a bug
- Cost: 370.000.000 $

Ariane 5 Flight 501 (1996)

- Exploded 37 seconds after takeoff
- Self-destructed due to a bug
- Cost: 370.000.000 $

```
horizontal_veloc_bias: integer;
horizontal_veloc_sensor: float;
-- ...
pragma suppress(numeric_error,
   horizontal_veloc_bias);
sensor_get(horizontal_veloc_sensor);
horizontal_veloc_bias :=
   integer(horizontal_veloc_sensor);
-- ...
```

Ariane 5 Flight 501 (1996)

- Exploded 37 seconds after takeoff
- Self-destructed due to a bug
- Cost: 370.000.000 $

```
1 horizontal_veloc_bias: integer;
2 horizontal_veloc_sensor: float;
3 -- ...
4 pragma suppress(numeric_error,
     horizontal_veloc_bias);
5 sensor_get(horizontal_veloc_sensor);
6 horizontal_veloc_bias :=
     integer(horizontal_veloc_sensor);
7 -- ...
```

Overflow from 64-bit float to 16-bit signed integer

Pentium FDIV bug (1994)

- Wrong floating-point division

Pentium FDIV bug (1994)

- Wrong floating-point division
- Cost: 475.000.000 $

Pentium FDIV bug (1994)

- Wrong floating-point division
- Cost: 475.000.000 $

$$\frac{4195835}{3145727} = 1.333739$$

- Error: 0.006%

CrowdStrike Faulty Update (2024)

- >5000 flights cancelled
- Impact on banks, governments, healthcare…

CrowdStrike Faulty Update (2024)

- >5000 flights cancelled
- Impact on banks, governments, healthcare…
- (est.) cost: 5.400.000.000 $

CrowdStrike Faulty Update (2024)

- >5000 flights cancelled
- Impact on banks, governments, healthcare…
- (est.) cost: 5.400.000.000 $

```
EXCEPTION_RECORD:  fffffb0d18d3ec28 -- ( .exr 0xfffffb0d18d3ec28 )
ExceptionAddress: fffff8021df335a1 (csagent+0x00000000000e35a1)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: 000000000000009c
Attempt to read from address 000000000000009c

CONTEXT:  fffffb0d18d3e460 -- ( .cxr 0xfffffb0d18d3e460 )
rax=fffffb0d18d3f2b0 rbx=0000000000000000 rcx=0000000000000003
rdx=fffffb0d18d3f280 rsi=ffff9a81b596f9a4 rdi=ffff9a81b596605c
rip=fffff8021df335a1 rsp=fffffb0d18d3ee60 rbp=fffffb0d18d3ef60
 r8=000000000000009c  r9=0000000000000000 r10=0000000000000000
r11=0000000000000014 r12=fffffb0d18d3ef28 r13=fffffb0d18d3f0d0
r14=000000000000001a r15=0000000000000004
iopl=0         nv up ei pl nz na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00050206
csagent+0xe35a1:
fffff802`1df335a1 458b08          mov     r9d,dword ptr [r8] ds:002b:00000000`0000009c=????????
Resetting default scope

BLACKBOXBSD: 1 (!blackboxbsd)


BLACKBOXNTFS: 1 (!blackboxntfs)


BLACKBOXPNP: 1 (!blackboxpnp)


BLACKBOXWINLOGON: 1

PROCESS_NAME:  System

READ_ADDRESS:  000000000000009c

ERROR_CODE: (NTSTATUS) 0xc0000005 - The instruction at 0x%p referenced memory at 0x%p. The memory could not

EXCEPTION_CODE_STR:  c0000005

EXCEPTION_PARAMETER1:  0000000000000000

EXCEPTION_PARAMETER2:  000000000000009c

EXCEPTION_STR:  0xc0000005

STACK_TEXT:
fffffb0d`18d3ee60 fffff802`1df09152     : 00000000`00000000 00000000`e01f008d fffffb0d`18d3f202 fffff802`1e
fffffb0d`18d3f000 fffff802`1df0a3e9     : 00000000`00000000 00000000`00000010 00000000`00000000 ffff9a81`b5
fffffb0d`18d3f130 fffff802`1e14954f     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00
fffffb0d`18d3f260 fffff802`1e145d9b     : ffff9a81`93735280 fffffb0d`18d3f5d0 00000000`00000000 00000000`00
fffffb0d`18d3f4d0 fffff802`1deb8fd0     : 00000000`000030f1 fffffb0d`18d3f790 ffff9a81`992cbb30 ffffe409`b7
```

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed personal information for 145.5 million people. The data included birth dates, addresses, some driver's license numbers, about 209,000 credit card numbers, and Social Security numbers—meaning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed 5 million people. The da ses, some driver's licens it card numbers, and S ning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed 5 million people. The da ses, some driver's license it card numbers, and S ning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed 5 million people. The da sses, some driver's license it card numbers, and S ning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

Cross-site scripting vulnerability

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed [illegible] million people. The da[illegible]ses, some driver's license[illegible]it card numbers, and S[illegible]ning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

Cross-site scripting vulnerability

BUSINESS INSIDER UK

The TalkTalk hack cost it £42 million

May 12, 2016, 8.33 AM

TalkTalk unveiled the full impact of the cyber attack it suffered last year — 95,000 customers left in the immediate aftermath and the cleanup cost hit £42 million ($60.53 million).

theguardian

TalkTalk hit with record £400k fine over cyber-attack

Wednesday 5 October 2016 14.00 BST

"The technique used by the attacker, called **SQL injection**, has been well known in security circles for almost 20 years. "SQL injection is well understood, defences exist and TalkTalk ought to have known it posed a risk to its data," the Information Commissioner's Office said."

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm Equifax disclosed a massive breach at the beginning of September, which exposed [illegible] million people. The da[illegible]ses, some driver's license[illegible]it card numbers, and S[illegible]ning that almost half the US population potentially had their crucial secret identifier exposed. Because the information Equifax coughed up was so sensitive, it's widely considered the worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

Cross-site scripting vulnerability

BUSINESS INSIDER UK

The TalkTalk hack cost it £42 million

May 12, 2016, 8:33 AM

TalkTalk unveiled the full impact of the cyber attack it suffered last year — 95,000 customers left in the immediate aftermath and the cleanup cost hit £42 million ($60.53 million).

SQL Injection

theguardian

TalkTalk hit with record £400k fine over cyber-attack

Wednesday 5 October 2016 14.00 BST

[illegible] called [illegible]own in [illegible] "SQL injection is well understood, defences exist and TalkTalk ought to have known it posed a risk to its data," the Information Commissioner's Office said."

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm disclosed a massive breach at the beginning of which exposed 5 people. The data s driver's license i numbers, and S n almost half the US population potentially had secret identifier exposed. Because the informa coughed up was so sensitive, it's widely consid worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

eBay Inc.
@ebayinc
Following

eBay asks all users to change passwo
due to cyberattack that compromised
financial info in a database:
ebayinc.com/in_the_news/st...

Reply Retweet Favorite More

Cross-site scripting vulnerability

vip-ohota.com.ua/co

www.ebay.co.uk/

The New York Times

DealBook Summit | Highlights | Takeaways | Jeff Bezos on Trump | Jerome Powell on Inflation | Sam Altman on A.I. and Musk

DealBook / Business & Policy

A Hacking of More Than $50 Million Dashes Hopes in the World of Virtual Currency

TalkTalk unveiled the full impact of the cyber attack it suffered last year — 95,000 customers left in the immediate aftermath and the cleanup cost hit £42 million ($60.53 million).

SQL Injection

theguardian

ecord £400k fine over cyber-attack

called own in "SQL injection is well understood, defences exist and TalkTalk ought to have known it posed a risk to its data," the Information Commissioner's Office said."

A lot more!

WIRED

LILY HAY NEWMAN SECURITY 12.31.17 07:00 AM

THE WORST HACKS OF 2017

Equifax

This was really bad. The credit monitoring firm disclosed a massive breach at the beginning of which exposed 5 people. The data s driver's license i numbers, and S n almost half the US population potentially had secret identifier exposed. Because the informa coughed up was so sensitive, it's widely consid worst corporate data breach ever. For now.

Out-of-date vulnerable dependency

eBay Inc.
@ebayinc
Following

eBay asks all users to change passwo due to cyberattack that compromised financial info in a database:
ebayinc.com/in_the_news/st...

Reply Retweet Favorite More

Cross-site scripting vulnerability

The New York Times

DealBook Summit | Highlights | Takeaways | Jeff Bezos on Trump | Jerome Powell on Inflation | Sam Altman on A.I. and Musk

DealBook / Business & Policy

A H Dashes Hopes in the World of Virtual Currency

Reentrancy vulnerability

TalkTalk unveiled the full impact of the cyber attack it suffered last year — 95,000 customers left in the immediate aftermath and the cleanup cost hit £42 million ($60.53 million).

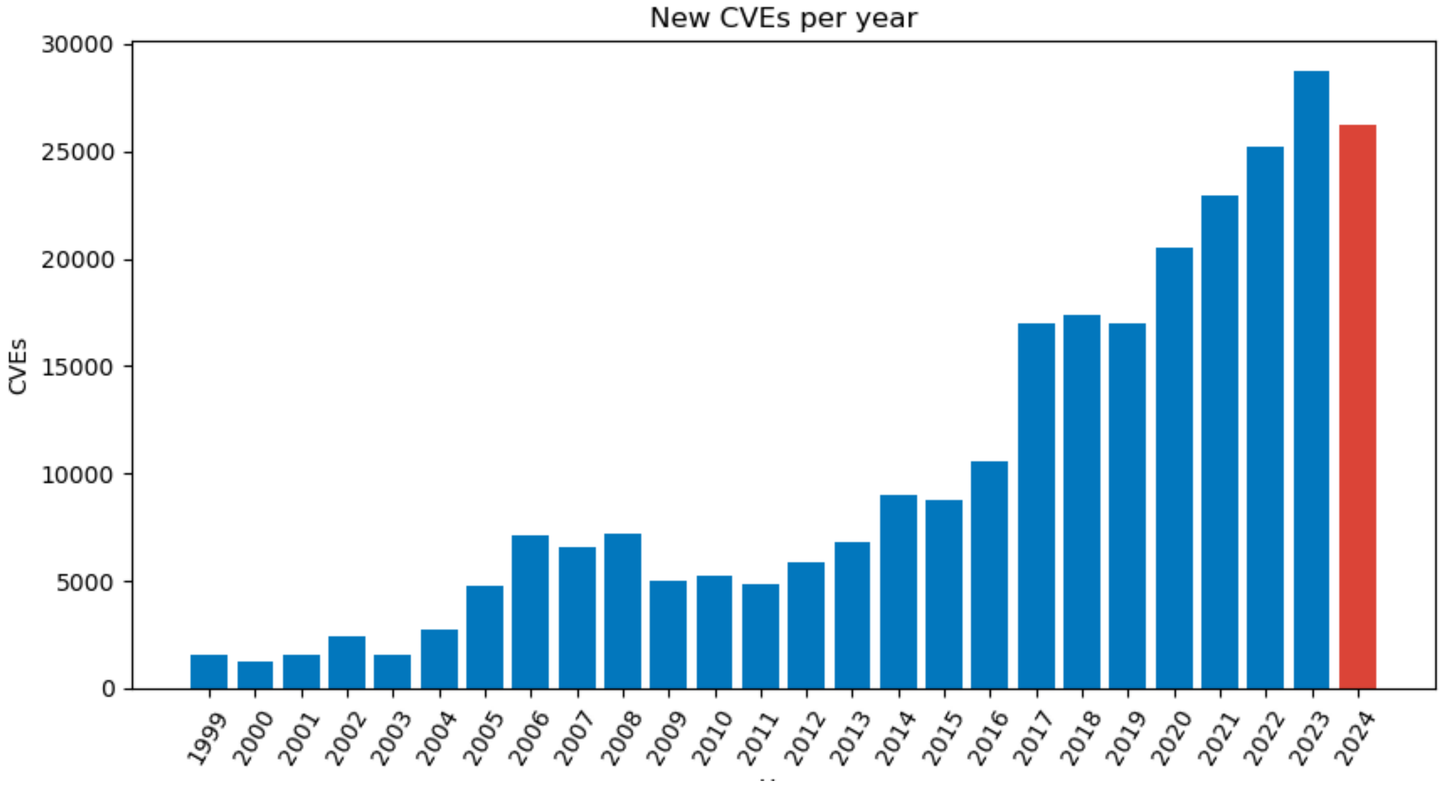
SQL Injection

theguardian

ecord £400k fine over cyber-attack

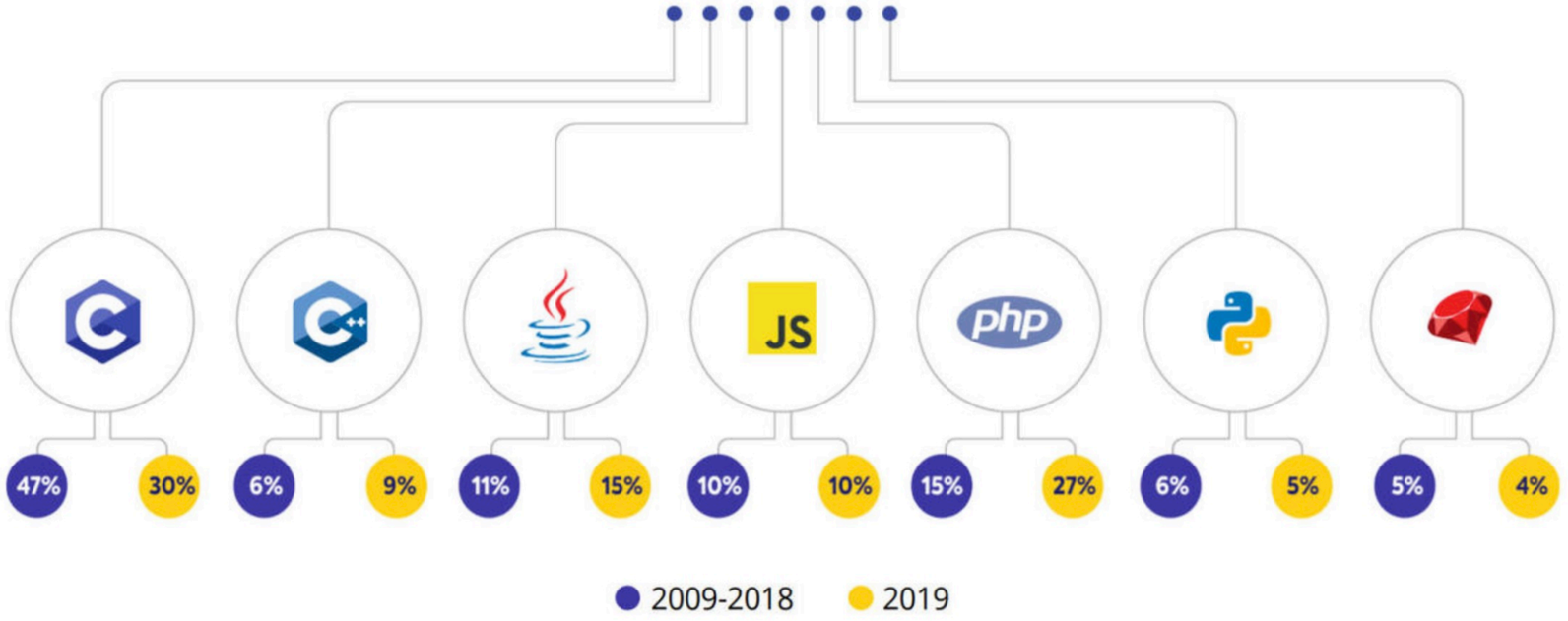
called own in "SQL injection is well understood, defences exist and TalkTalk ought to have known it posed a risk to its data," the Information Commissioner's Office said."

The trend per year



Source: CVE program (*cybersec* vulnerabilities)

Open-source vulnerabilities per language



Open-source vulnerabilities per language, 2019 vs 2009-2018: C still had most, due to the high volume of code written in it.

Validation *vs* Verification

Validation *vs* Verification

- Validation: does the software system meets the user's real needs?

Validation *vs* Verification

- Validation: does the software system meets the user's real needs?

"Are we building the right software?"

Validation *vs* Verification

- Validation: does the software system meets the user's real needs?

"Are we building the right software?"

- Verification: does the software system meets the requirements specifications?

Validation *vs* Verification

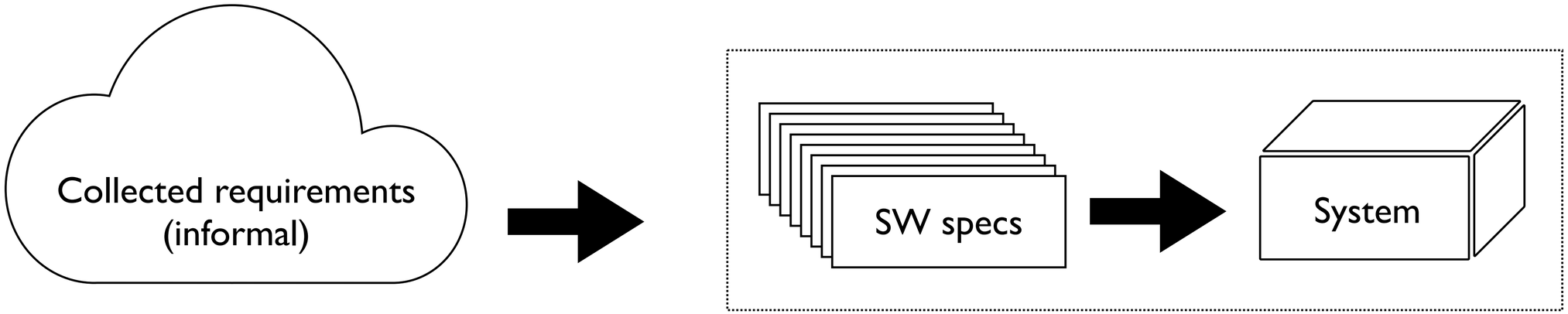
- Validation: does the software system meets the user's real needs?

"Are we building the right software?"

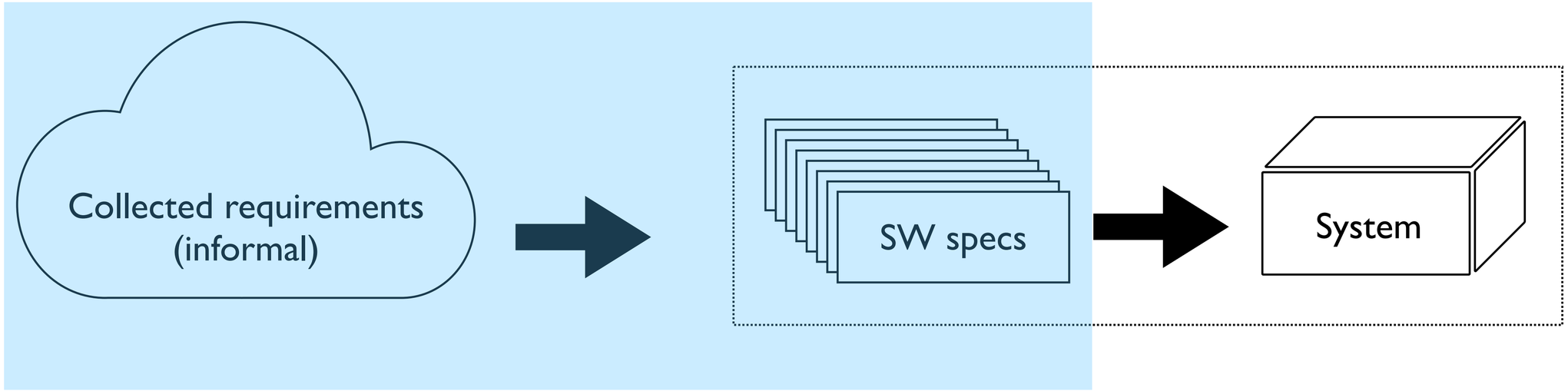
- Verification: does the software system meets the requirements specifications?

"Are we building the software right?"

Validation *vs* Verification

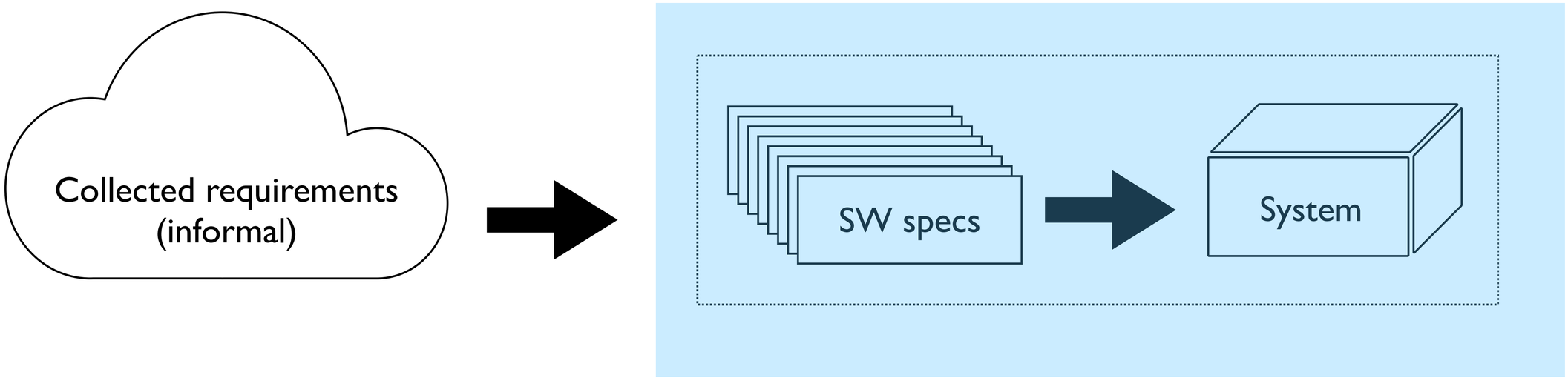


Validation *vs* Verification



Validation
(e.g., usability testing, user feedback, requirement engineering)

Validation *vs* Verification



Verification
(e.g., testing, inspections, static analysis)

Example: elevator response

Example: elevator response

- Validable but unverifiable specification: *if a user presses a request button at floor i, an available elevator must arrive at floor i in a reasonable time*

Example: elevator response

- Validable but unverifiable specification: *if a user presses a request button at floor i, an available elevator must arrive at floor i in a reasonable time*
- Verifiable specification: *if a user presses a request button at floor i, an available elevator must arrive at floor i within 30 seconds*

Software Verification

Software Verification

- The task is to verify some correctness statement about a program:

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document
  - The **absence of non-functional errors** (not about *what* the program does, but *how*)

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document
  - The **absence of non-functional errors** (not about *what* the program does, but *how*)
    - satisfaction of space/time constraints

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document
  - The **absence of non-functional errors** (not about *what* the program does, but *how*)
    - satisfaction of space/time constraints
    - absense of run-time errors

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document
  - The **absence of non-functional errors** (not about *what* the program does, but *how*)
    - satisfaction of space/time constraints
    - absense of run-time errors
    - satisfaction of security constraints

Software Verification

- The task is to verify some correctness statement about a program:
  - The **functional correctness**, i.e. that the program offers the services exactly described in its specification document
  - The **absence of non-functional errors** (not about *what* the program does, but *how*)
    - satisfaction of space/time constraints
    - absense of run-time errors
    - satisfaction of security constraints
  - In this course, we mean verification in a strong sense: we look for **guarantees**

Bad news: there's no silver bullet

Bad news: there's no silver bullet

- Don't expect a verification method that is
  - **Automatic**: does not require human interaction
  - **Powerful**: able to prove non-trivial properties
  - **Sound**: never prove the property if it doesn't hold
  - **Complete**: always prove the property if it holds, never fail to claim correctness if the program is correct

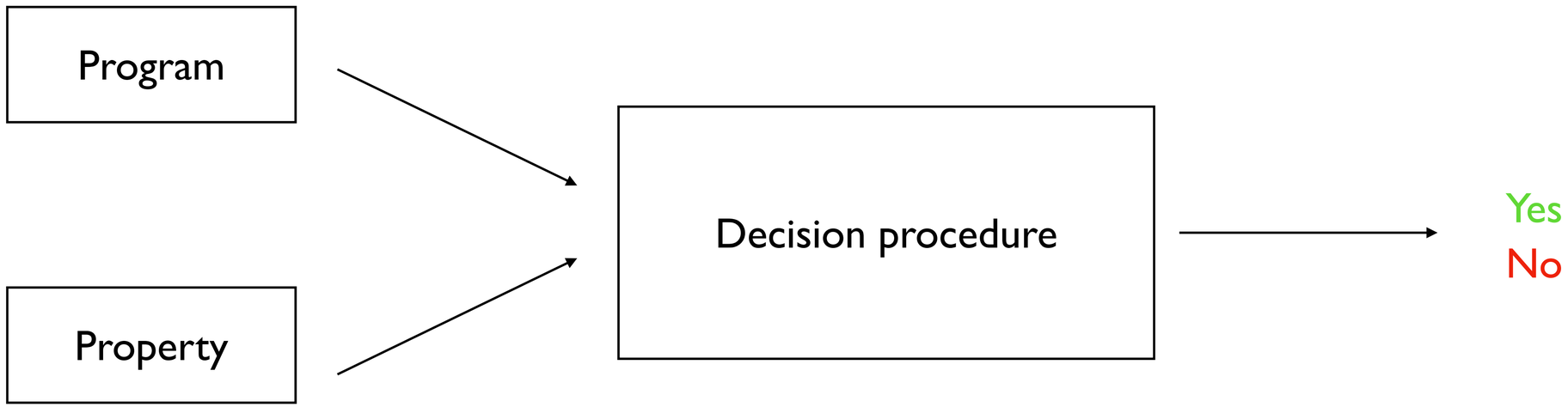
Bad news: there's no silver bullet

- Don't expect a verification method that is
  - **Automatic**: does not require human interaction
  - **Powerful**: able to prove non-trivial properties
  - **Sound**: never prove the property if it doesn't hold
  - **Complete**: always prove the property if it holds, never fail to claim correctness if the program is correct
- **Rice's Theorem** states that all non trivial properties of program behaviour in a Turing-complete programming language are undecidable

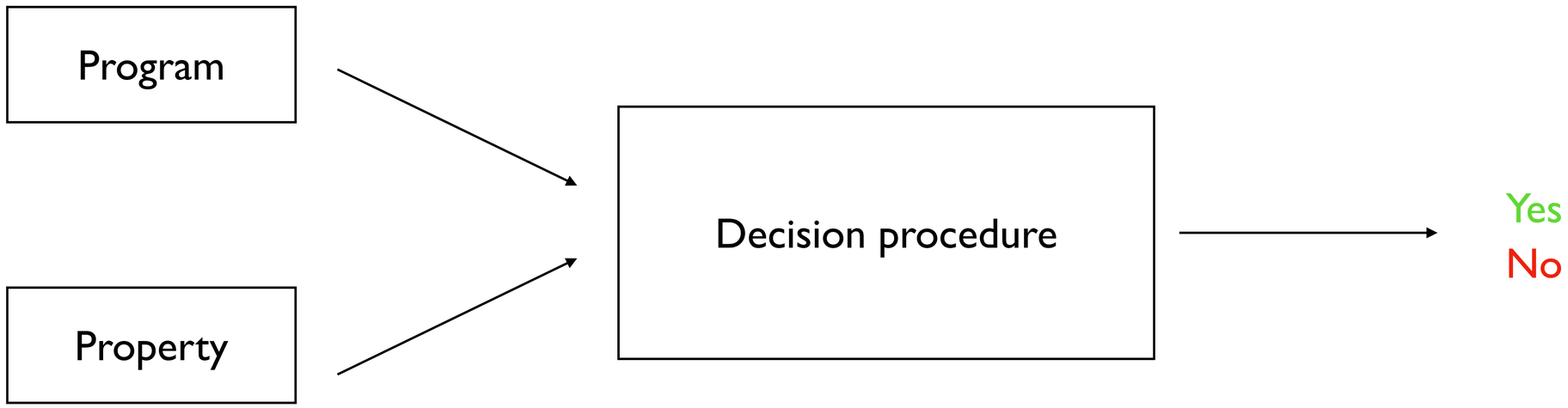
Bad news: there's no silver bullet

- Don't expect a verification method that is
  - **Automatic**: does not require human interaction
  - **Powerful**: able to prove non-trivial properties
  - **Sound**: never prove the property if it doesn't hold
  - **Complete**: always prove the property if it holds, never fail to claim correctness if the program is correct
- **Rice's Theorem** states that all non trivial properties of program behaviour in a Turing-complete programming language are undecidable
- **Undecidability** of a property means that there is no automatic verification method that is sound and complete

You can’t always get what you want



You can’t *ever* get what you want



Behavioural properties are **undecidable**:
the halting problem can be embedded in almost every property of interest

What to give up?

- **Soundness**
- **Powerful**
- **Automatic**
- **Completeness**

What to give up?

- **Soundness** no way!
- **Powerful**
- **Automatic**
- **Completeness**

What to give up?

- **Soundness** no way!
- **Powerful** how much as possible
- **Automatic**
- **Completeness**

What to give up?

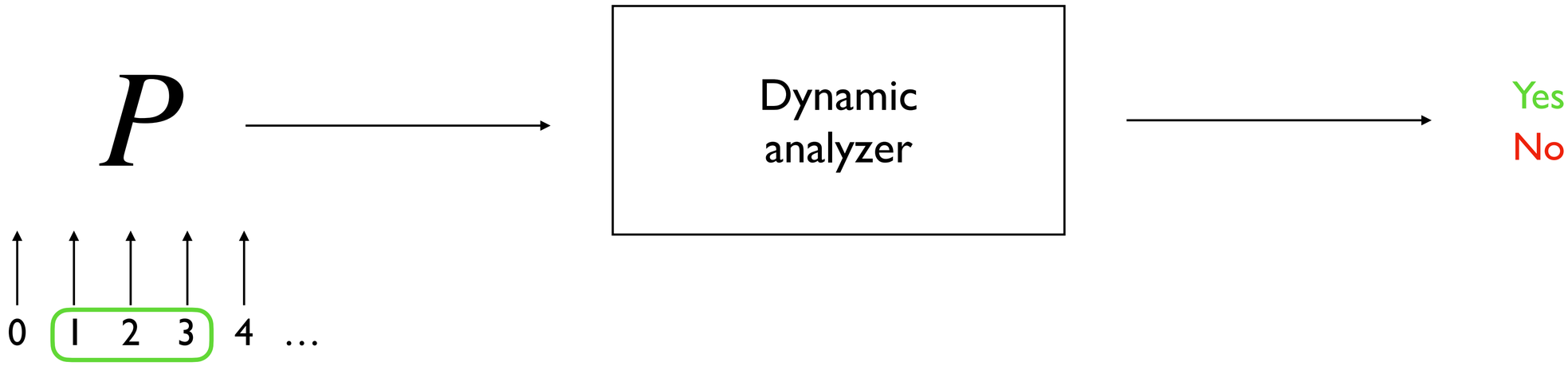
- **Soundness** no way!
- **Powerful** as much as possible
- **Automatic** deductive verification/interactive theorem proving (out of scope)
- **Completeness**

What to give up?

- **Soundness** no way!
- **Powerful** as much as possible
- **Automatic** deductive verification/interactive theorem proving (out of scope)
- **Completeness** hard or even impossible to obtain (more later in this course)

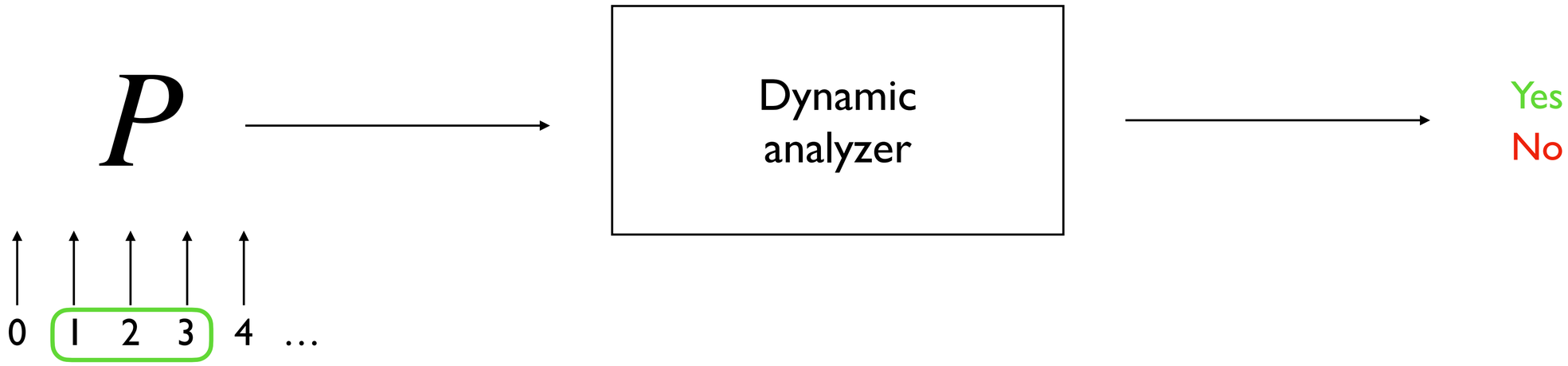
Dynamic analysis *vs* Static analysis

Dynamic analysis



Dynamic analysis *vs* Static analysis

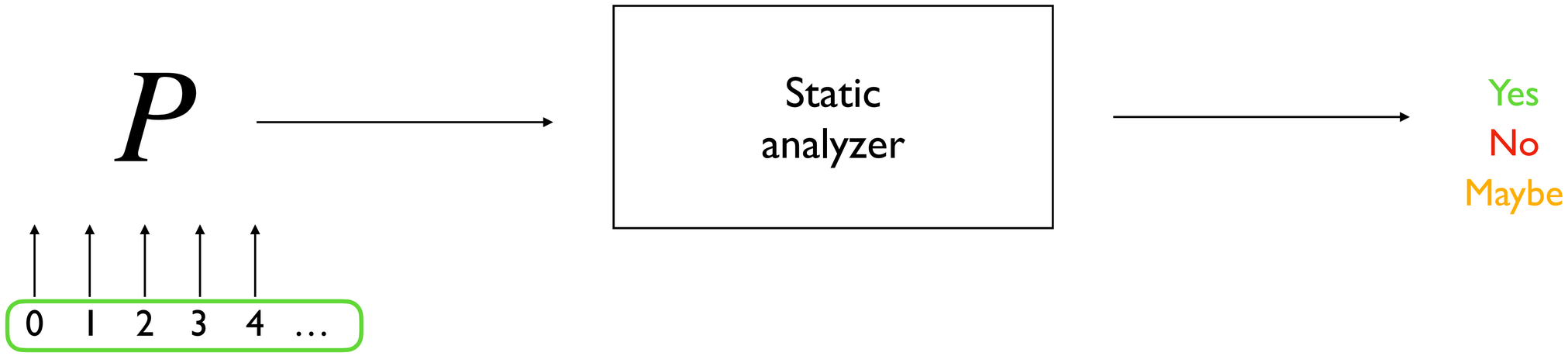
Dynamic analysis



- Correct w.r.t. a concrete finite set of inputs and a program (e.g., testing, fuzzing)

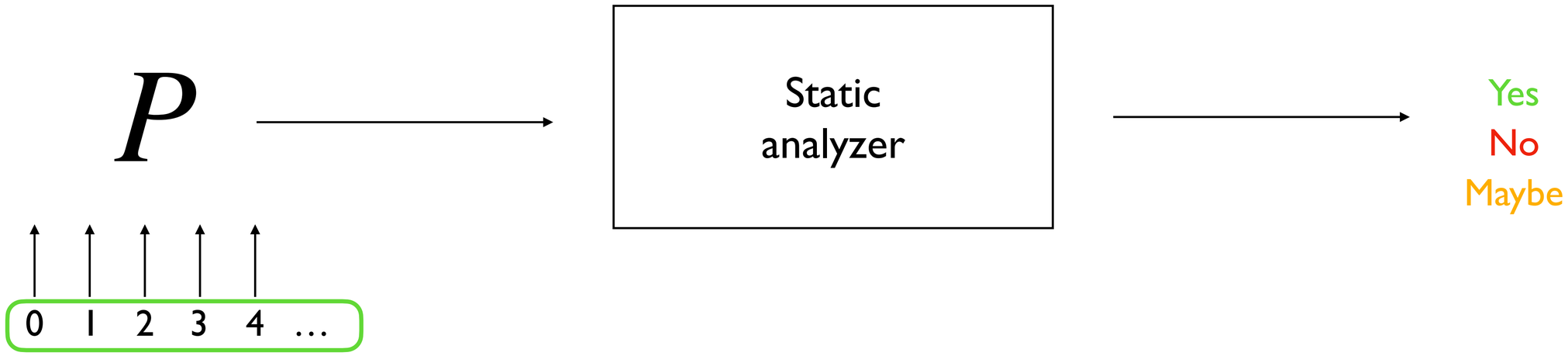
Dynamic analysis *vs* Static analysis

Static analysis



Dynamic analysis *vs* Static analysis

Static analysis



- Correct w.r.t. *all inputs* and a program, but may lose precision (static analysis)

Course goals

Static analysis

Course goals

Static analysis

- Understand basic techniques for developing program static analyses

Seminars

- Greta Dolcetti (UniVE)
- Isabella Mastroeni (UniVR)
- Caterina Urban (INRIA)
- Laura Titolo (ex NASA, CodeMetal Inc.)
- Roberto Bagnara (UniPR)
- Pietro Ferrara (ex IBM, UniVE)
- Luca Negrini (UniVE)
- Luca Olivieri (UniVE)

- E-mail: vincenzo.arceri@unipr.it

- E-mail: vincenzo.arceri@unipr.it
- Ufficio: (Plesso di Matematica, primo piano a sinistra)

- E-mail: vincenzo.arceri@unipr.it
- Ufficio: (Plesso di Matematica, primo piano a sinistra)
- Ricevimento: fissare appuntamento via email

- E-mail: vincenzo.arceri@unipr.it
- Ufficio: (Plesso di Matematica, primo piano a sinistra)
- Ricevimento: fissare appuntamento via email
- Materiale didattico e avvisi: pagina Elly del corso

Exam

- Seminar (1+ paper discussion) and oral exam