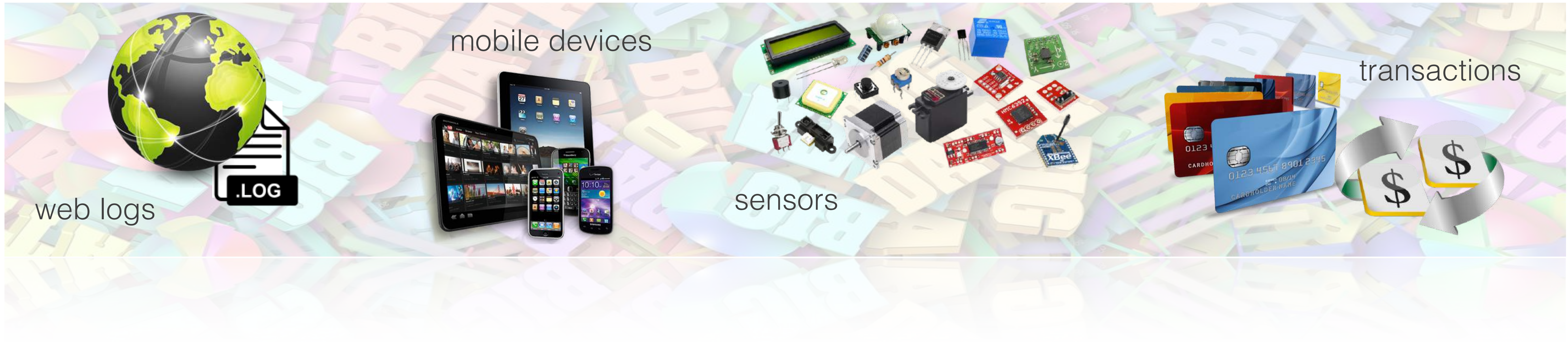
Static Analysis for Data Science

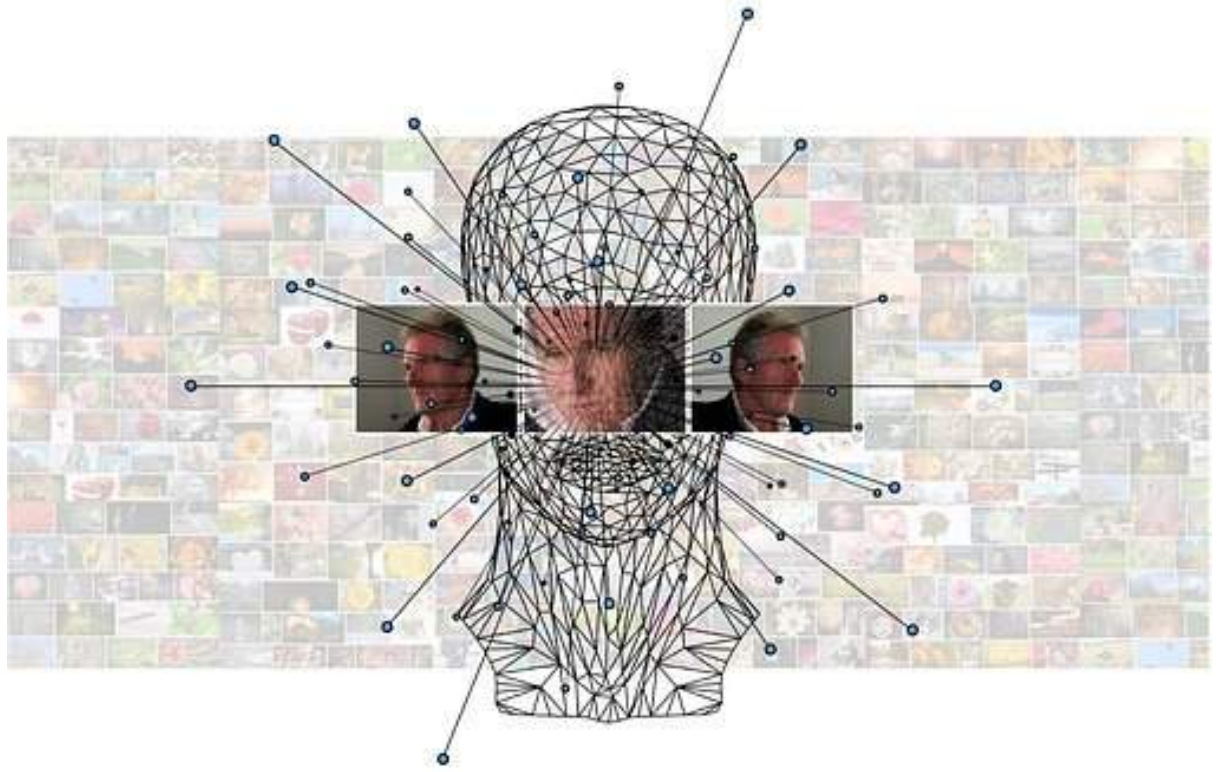
Caterina Urban
ANTIQUE Research Team, Inria & École Normale Supérieure | Université PSL

Data Science is Everywhere

vast amounts of cheap and ubiquitous data



impressive advances in machine learning



Data Science is Revolutionizing Industries

software plays an increasingly important role in assisting or even autonomously performing tasks

retail
- personalized recommendations
- targeted marketing

WIRED BUSINESS MORE SIGN IN SUBSCRIBE
ERIC NIILER BUSINESS 03.25.2019 07:00 AM

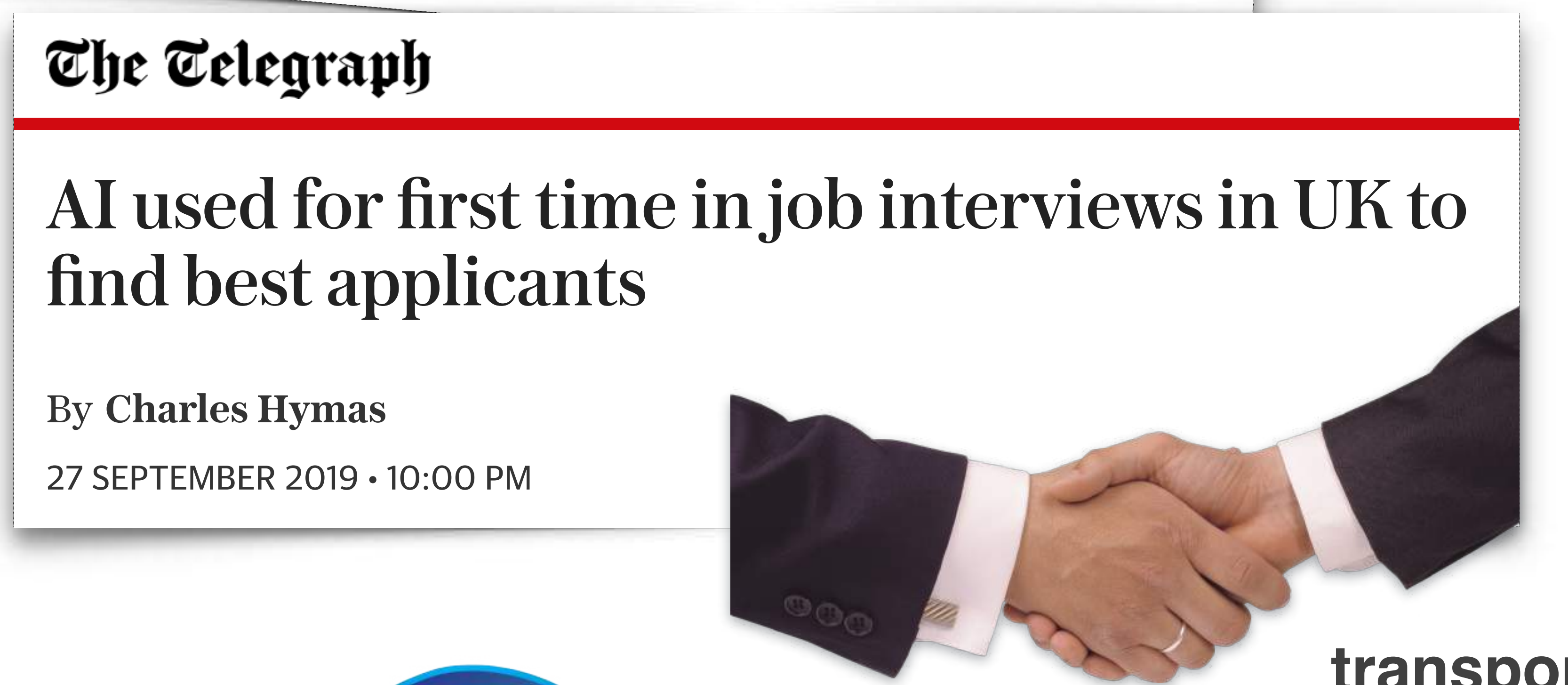
Can AI Be a Fair Judge in Court? Estonia Thinks So

Estonia plans to use an artificial intelligence program to decide some small-claims cases, part of a push to make government services smarter.

pharmaceutical
- predictive models
- patient selection

health care
- personalized treatments
- preventive care

manufacturing
- equipment failure predictions
- internet of things



The Telegraph

AI used for first time in job interviews in UK to find best applicants

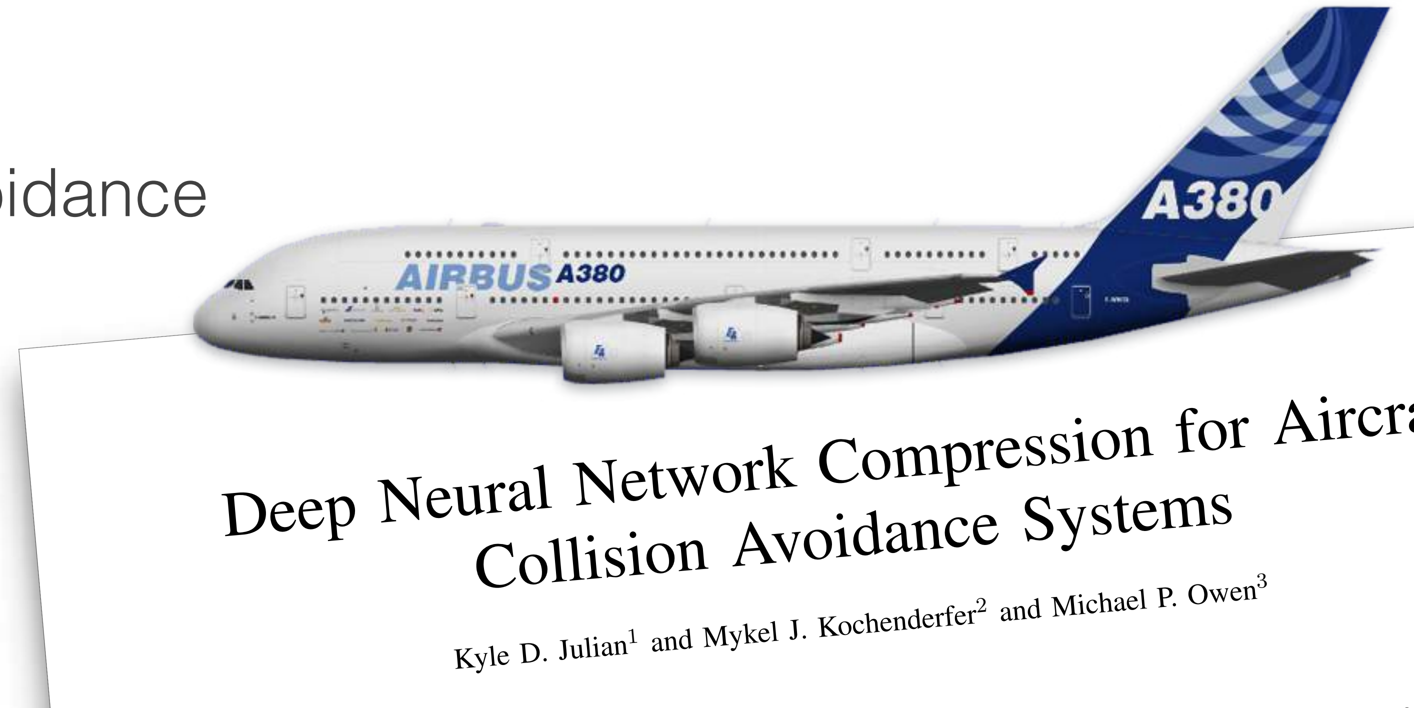
By Charles Hymas

27 SEPTEMBER 2019 • 10:00 PM

transportation
- self-driving cars
- aircraft collision avoidance

finance
- predictive models
- customized product offerings

energy
- exploration and discovery
- accident prevention

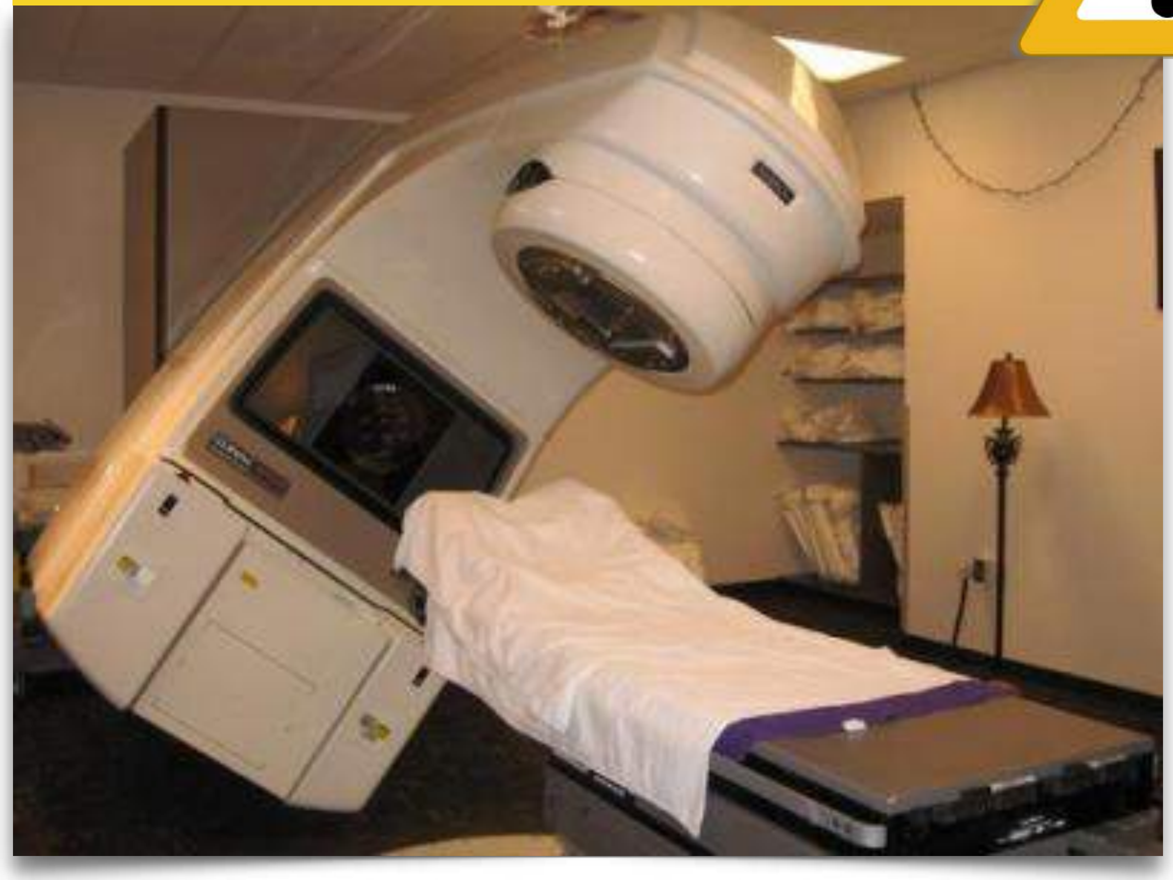


Deep Neural Network Compression for Aircra
Collision Avoidance Systems

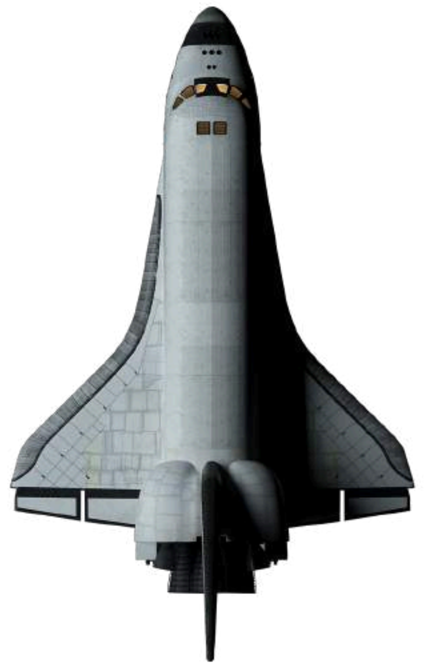
Kyle D. Julian[1] and Mykel J. Kochenderfer[2] and Michael P. Owen[3]

Software = Trouble

4 dead, 2 lifelong injuries



Therac-25, 1985-1987



loss of more than $370 000 000

Ariane 5, 4 June 1996

at least 89 deaths

Toyota, 2000-2010

Data Science Software = Silent Trouble

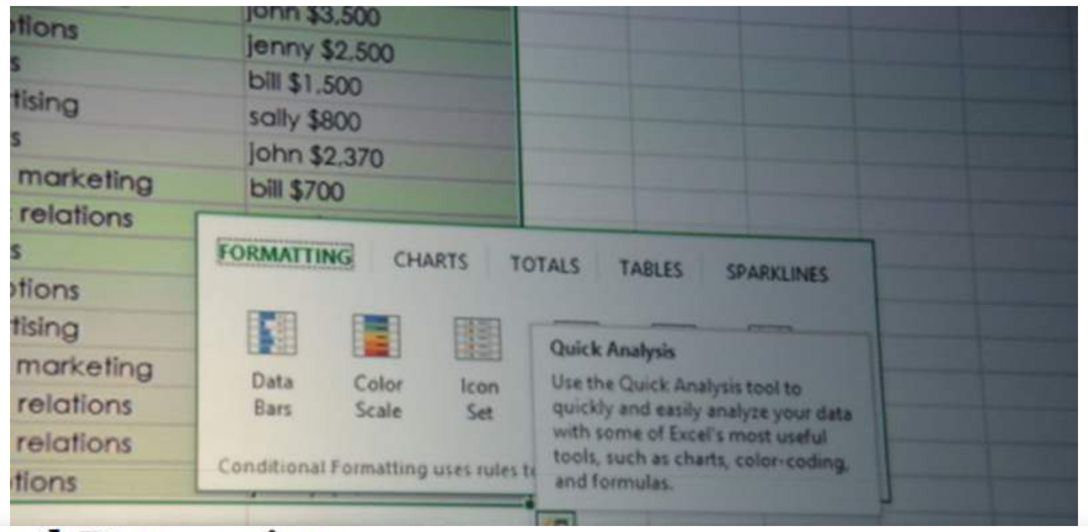
programming errors that do not cause failures can remain unnoticed

MICROSOFT

A simple Excel calculation error puts a famous economic study under scrutiny

By Nathan Ingraham | Apr 17, 2013, 11:08am EDT

Source Financial Times, University of Massachusetts, and Next New Deal | Via Business Insider and Bloomberg Businessweek



The Excel Depression

By PAUL KRUGMAN
Published: April 18, 2013 | 470 Comments

In this age of information, math errors can lead to disaster. NASA's Mars Orbiter crashed because engineers forgot to convert to metric measurements; JPMorgan Chase's "London Whale" venture went bad in part because modelers divided by a sum instead of an average. So, did an Excel coding error destroy the economies of the Western world?

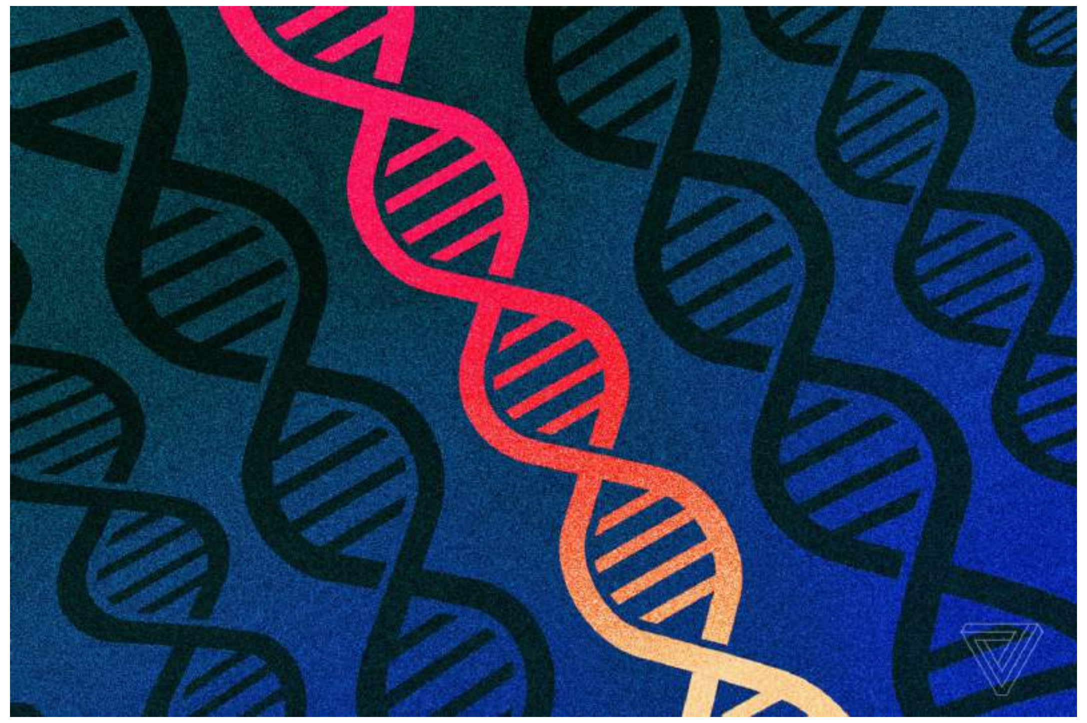
FACEBOOK
TWITTER
GOOGLE+
SAVE
EMAIL

MICROSOFT | REPORT | SCIENCE

Scientists rename human genes to stop Microsoft Excel from misreading them as dates

Sometimes it's easier to rewrite genetics than update Excel

By James Vincent | Aug 6, 2020, 8:44am EDT

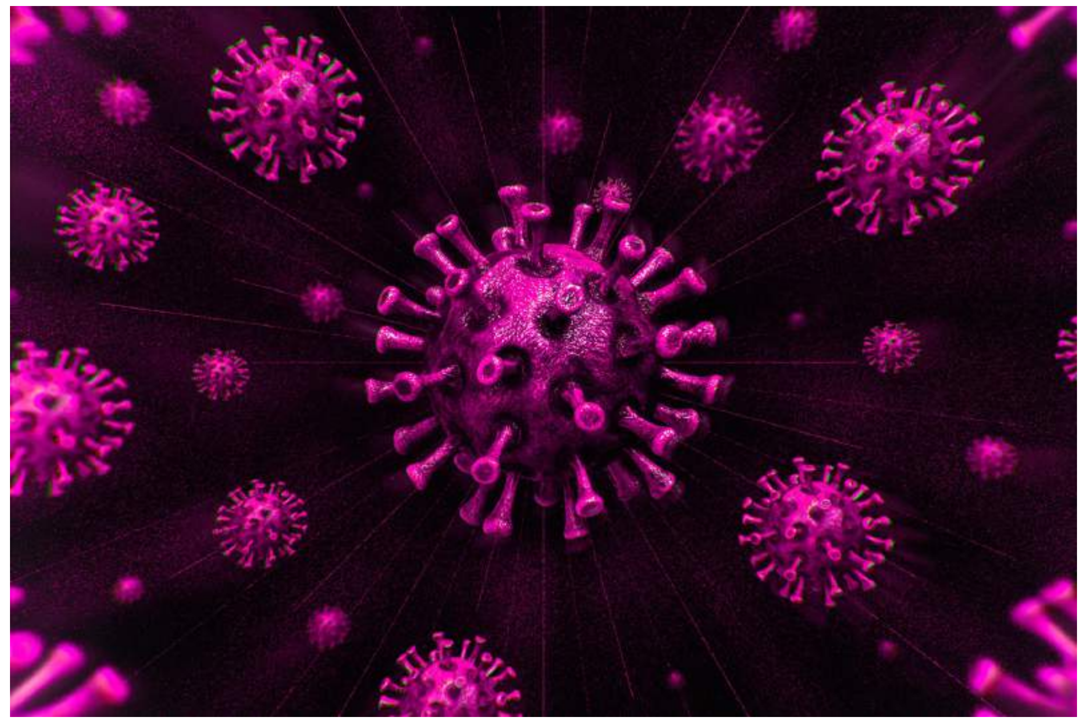


SCIENCE | US & WORLD | TECH

Excel spreadsheet error blamed for UK's 16,000 missing coronavirus cases

The case went missing after the spreadsheet hit its filesize limit

By James Vincent | Oct 5, 2020, 9:41am EDT



Data Science Software = Societal Impact

software can be **biased** and invade our **privacy**

3,658,826 views | Feb 16, 2012, 11:02am

How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did

Kashmir Hill Former Staff
Tech
Welcome to The Not-So Private Parts where technology & privacy collide

nature

NEWS · 24 OCTOBER 2019

UPDATE 26 OCTOBER 2019

Millions of black people affected by racial bias in health-care algorithms

Study reveals rampant racism in decision-making software used by US hospitals – and highlights ways to correct it.

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica

May 23, 2016

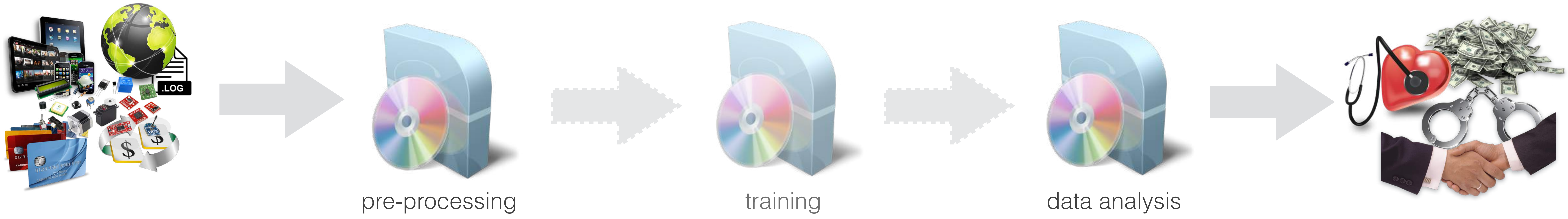
BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

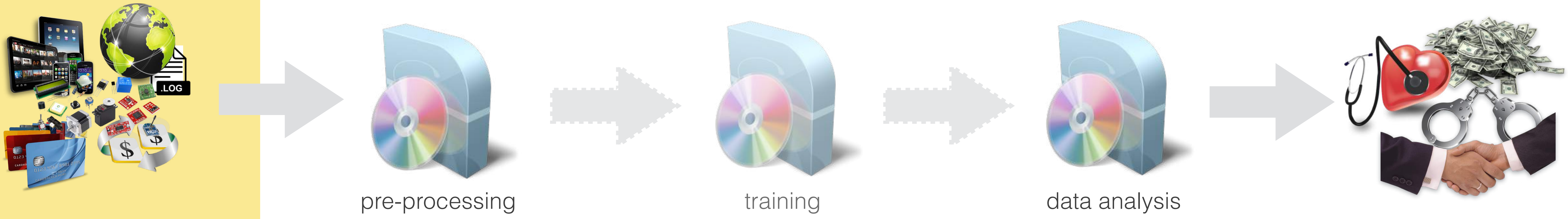
Jeffrey Dastin

Data Science Pipelines

software is often necessarily written by domain experts rather than software engineers



Data is Dirty



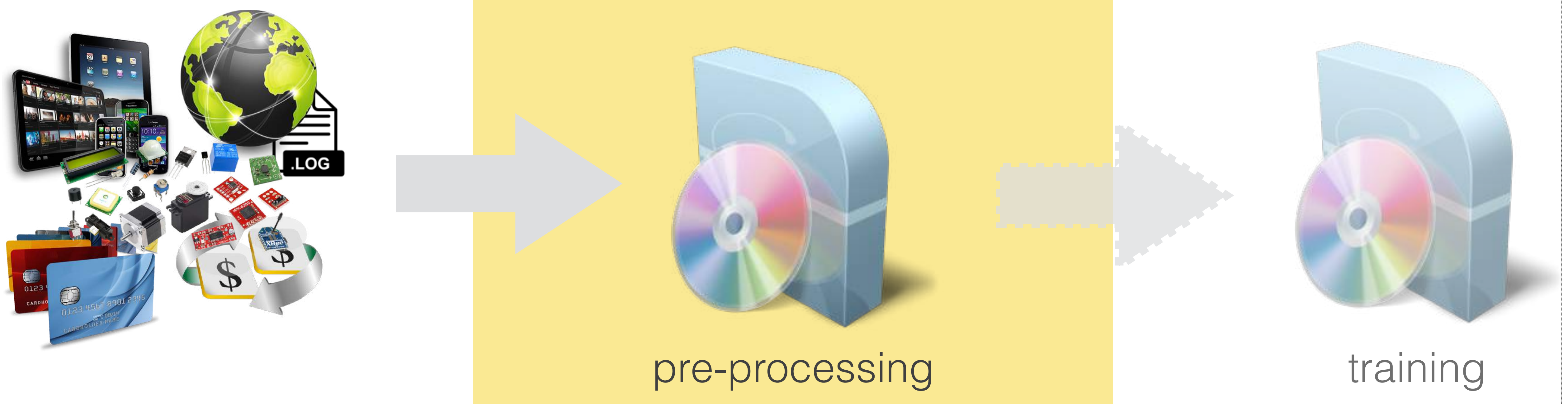
inconsistent data

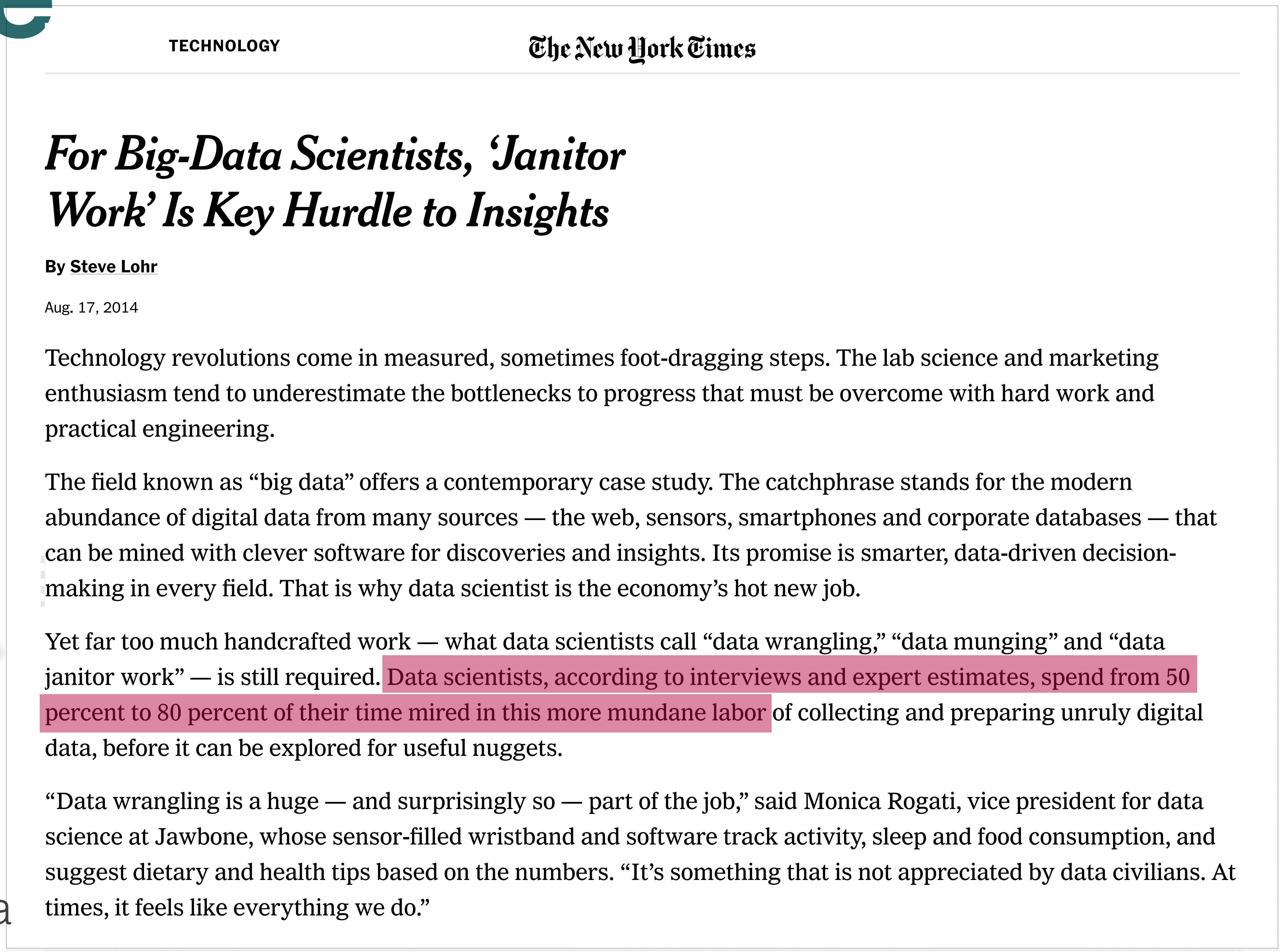
incorrect data

incomplete data

inaccurate data

Pre-Processing is Fragile





TECHNOLOGY

The New York Times

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

By Steve Lohr

Aug. 17, 2014

Technology revolutions come in measured, sometimes foot-dragging steps. The lab science and marketing enthusiasm tend to underestimate the bottlenecks to progress that must be overcome with hard work and practical engineering.

The field known as "big data" offers a contemporary case study. The catchphrase stands for the modern abundance of digital data from many sources — the web, sensors, smartphones and corporate databases — that can be mined with clever software for discoveries and insights. Its promise is smarter, data-driven decision-making in every field. That is why data scientist is the economy's hot new job.

Yet far too much handcrafted work — what data scientists call "data wrangling," "data munging" and "data janitor work" — is still required. Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.

"Data wrangling is a huge — and surprisingly so — part of the job," said Monica Rogati, vice president for data science at Jawbone, whose sensor-filled wristband and software track activity, sleep and food consumption, and suggest dietary and health tips based on the numbers. "It's something that is not appreciated by data civilians. At times, it feels like everything we do."

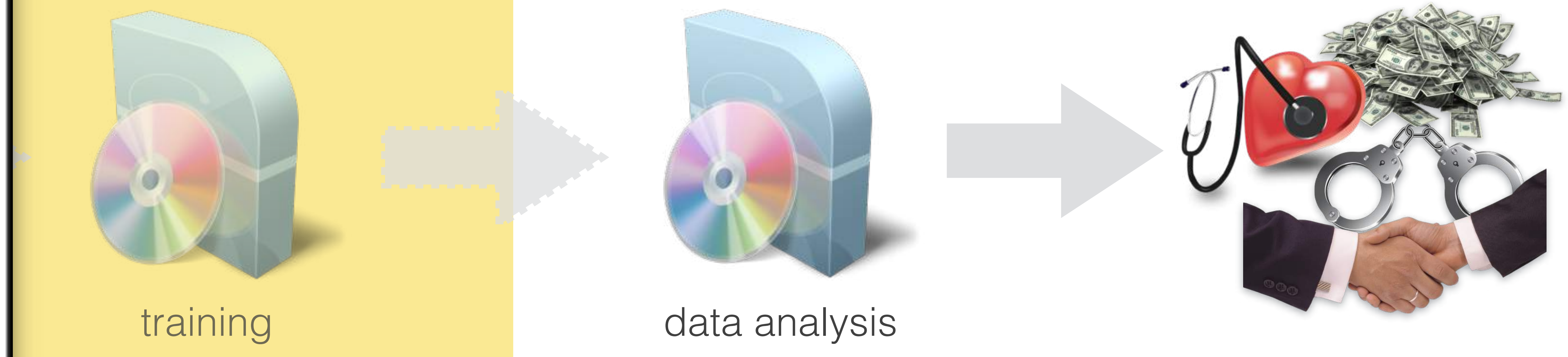
mislabeled data

accidentally duplicated data

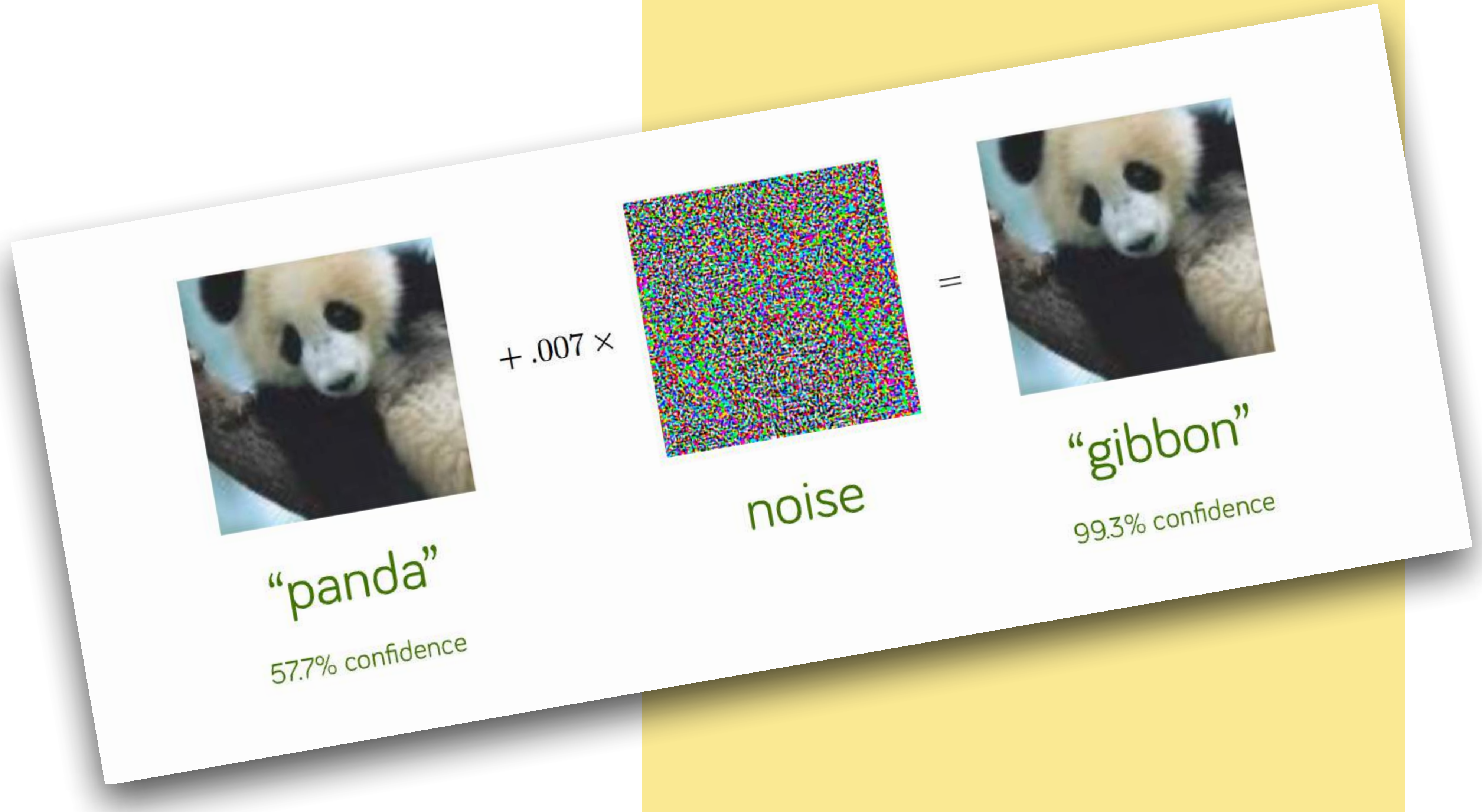
wrongly converted data

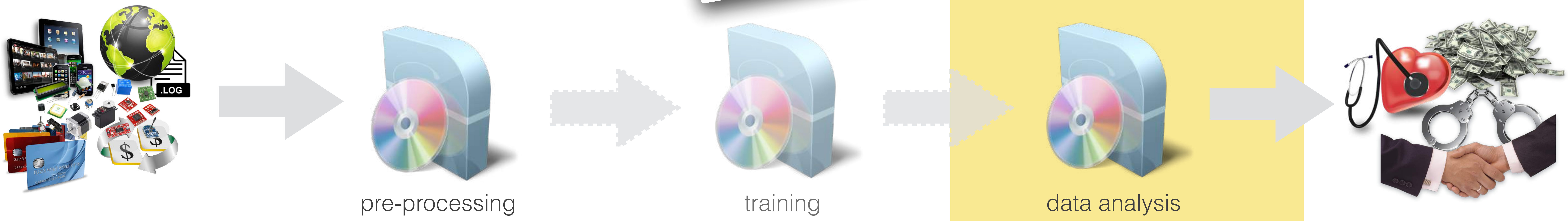
accidentally (un)used data

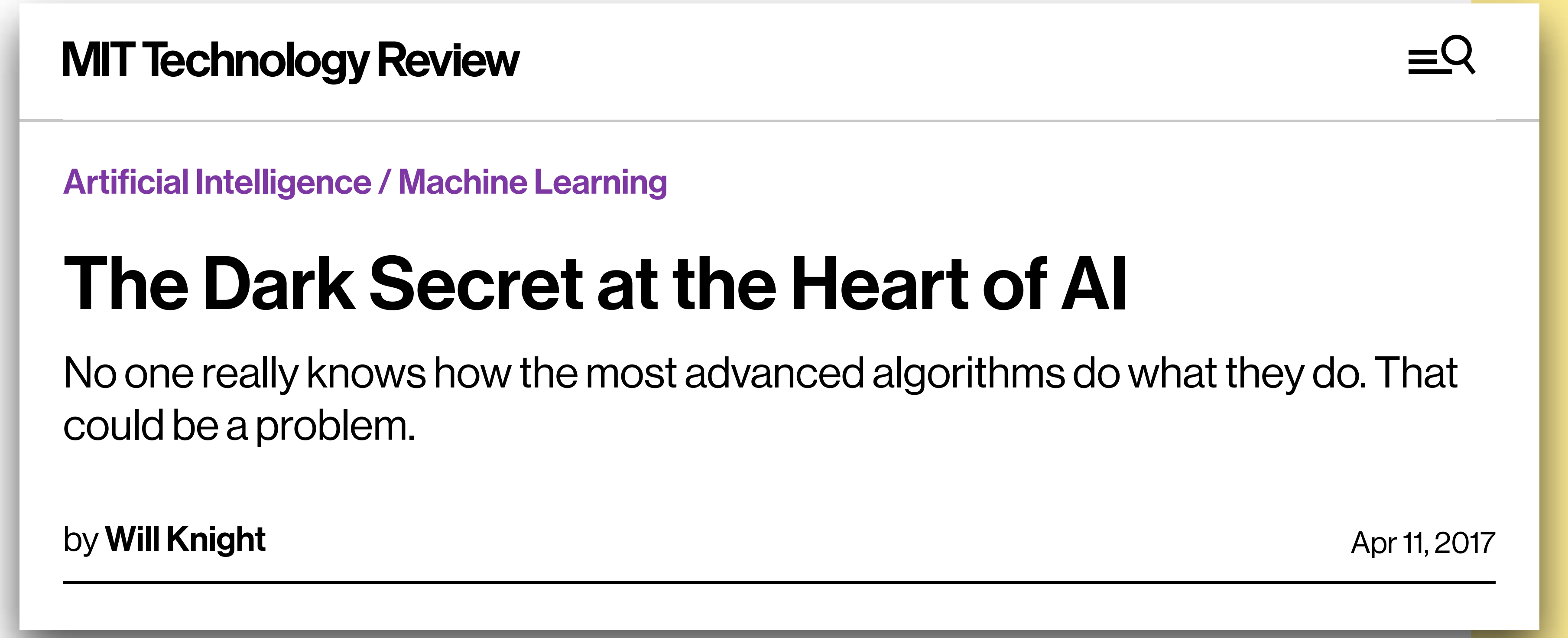
Accuracy is Meaningless



Inscrutability






MIT Technology Review

Artificial Intelligence / Machine Learning

The Dark Secret at the Heart of AI

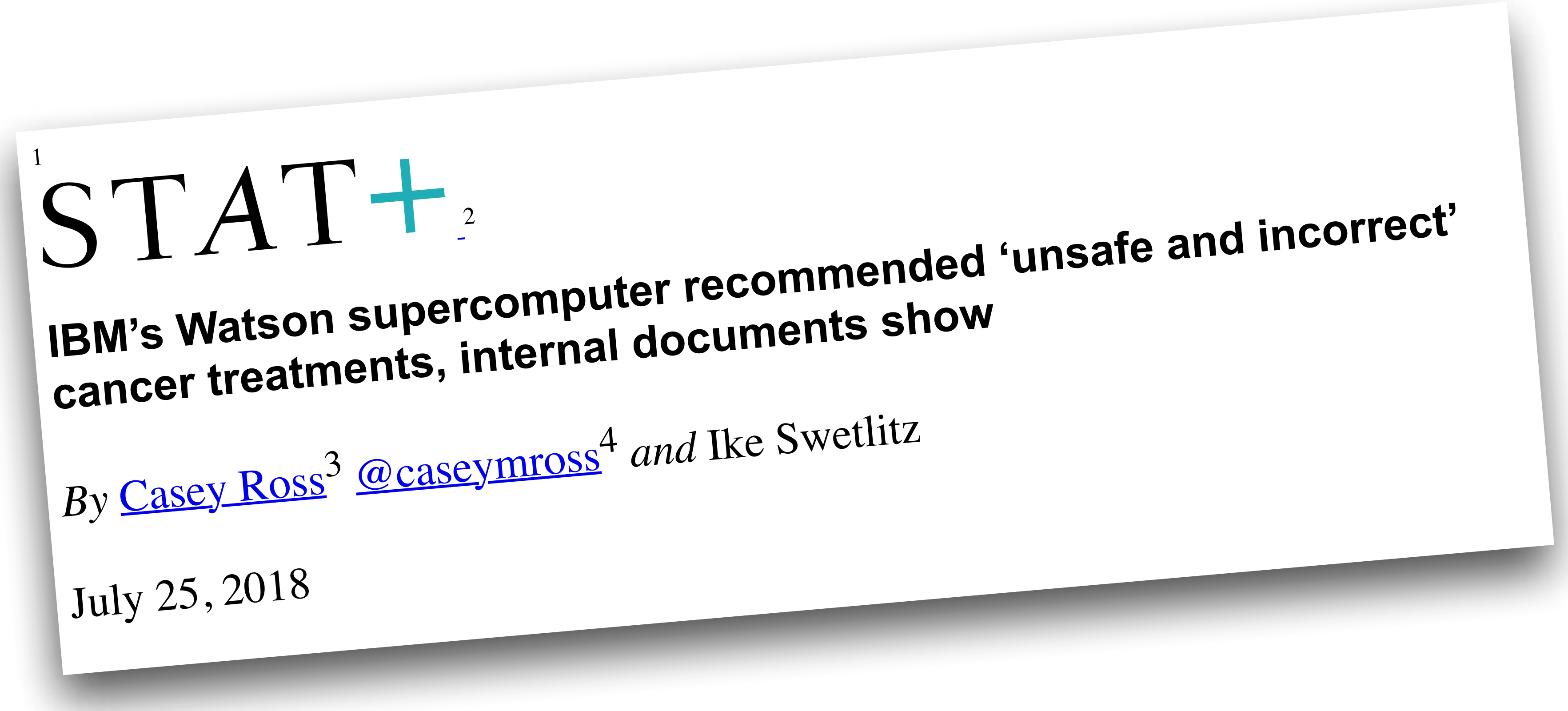
No one really knows how the most advanced algorithms do what they do. That could be a problem.

by **Will Knight**

Apr 11, 2017

Data Science in Safety-Critical Scenarios

health care
- personalized treatments
- preventive care



STAT+

IBM's Watson supercomputer recommended 'unsafe and incorrect' cancer treatments, internal documents show

By Casey Ross[3] @caseymross[4] *and* Ike Swetlitz

July 25, 2018

A self-driving Uber ran a red light last December, contrary to company claims

Internal documents reveal that the car was at fault

By Andrew Liptak | @AndrewLiptak | Feb 25, 2017, 11:08am EST

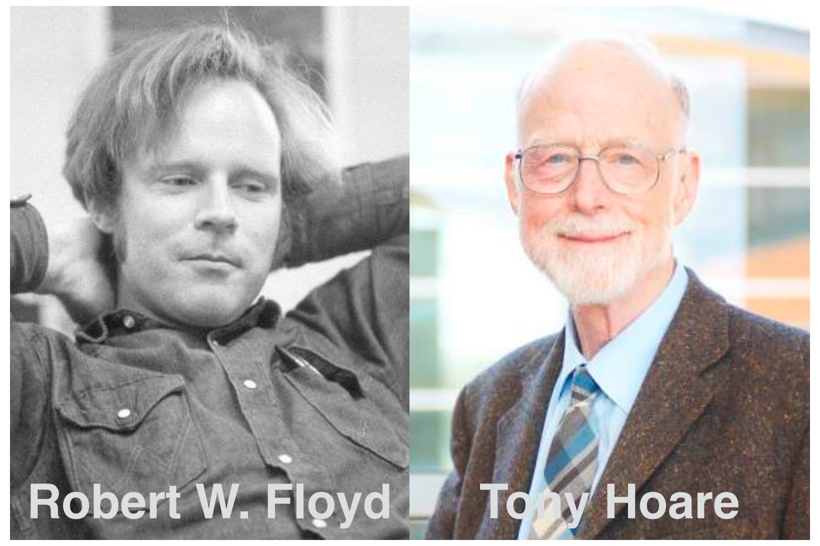
transportation
- self-driving cars
- aircraft collision avoidance

Deep Neural Network Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian[1] and Mykel J. Kochenderfer[2] and Michael P. Owen[3]

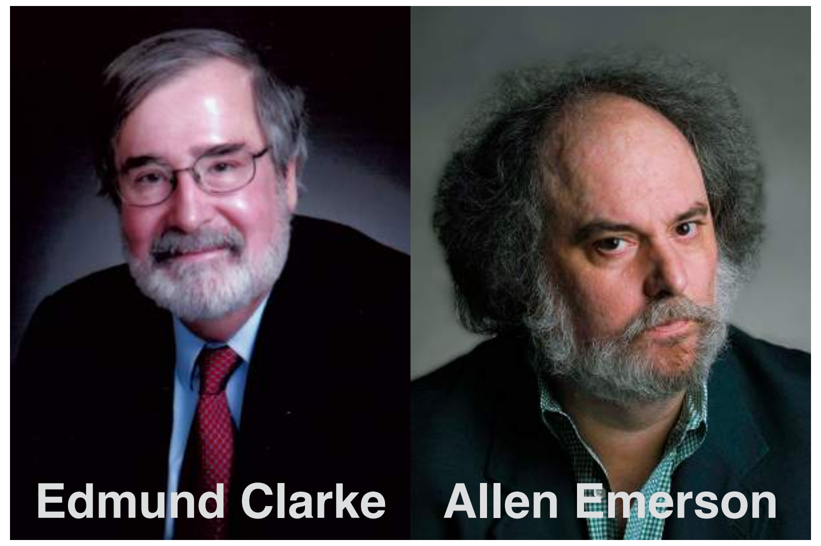
Formal Methods to the Rescue

provide <u>mathematical guarantees</u> of software safety, reliability, and security



Deductive Verification

- extremely **expressive**
- **relies on the user** to guide the proof



Model Checking

- analysis of a **model** of the software
- **sound and complete with respect to the model**

Static Analysis

- analysis of the **source or object code**
- fully **automatic** and **sound** by construction
- generally **not complete**

Static Analysis Today

integral part of the development of safety-critical software

AIRBUS

aviation software

esa

space software

HELBAKO

automotive software

AREVA

nuclear software

successfully employed by software companies

Google

Facebook

Infer

Microsoft

Static Analysis Tomorrow

integral part of the development of data science software

more and more **legal regulations**

Europeal General Data Protection Regulation, 2016

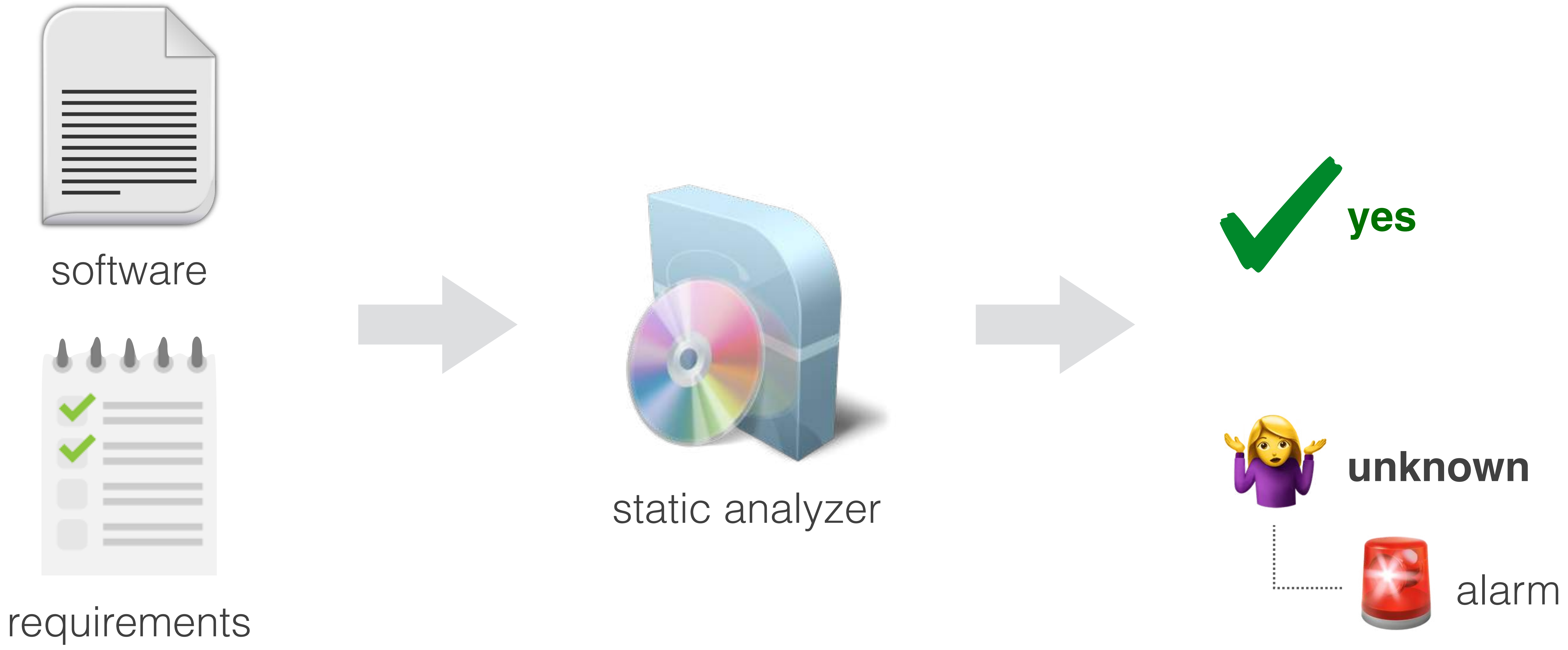
more and more **administrative audits**

Static Analysis

Quick Tutorial

A Mathematically-Proven Hard Problem

Relaxed Problem



Abstraction and Over-Approximation

"the purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise"

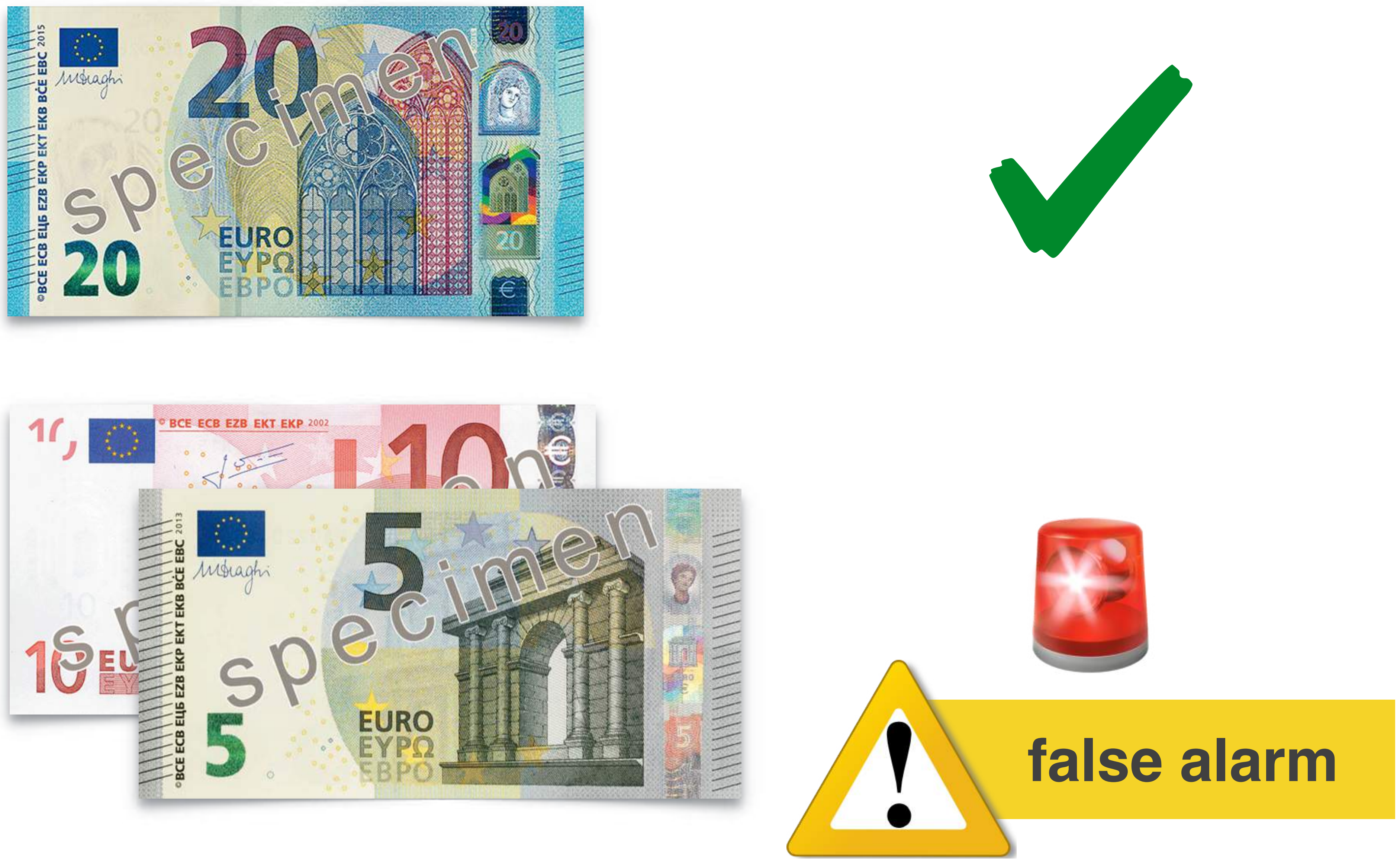
?

Abstraction and Over-Approximation

"the purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise"

Abstraction and Over-Approximation

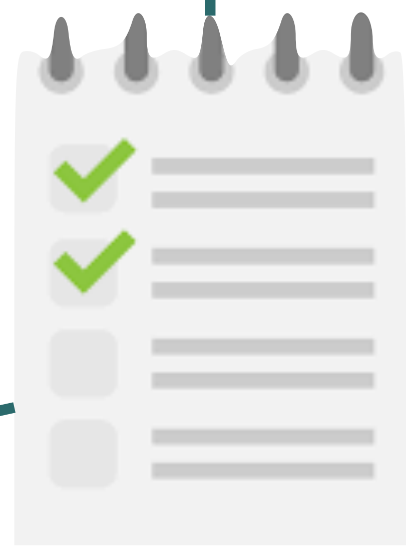
“the purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise”



Example

```
requires A ≥ 0 and B ≥ 0
def mod(A, B):
    Q = 0
    R = A
    while R >= B:
        R = R - B
        Q = Q + 1
    return R
ensures R ≥ 0
```

software



requirements

static analyzer

?

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 7$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 7$

6: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

$$
\begin{array}{lllll}
1: & A \mapsto 10 & B \mapsto 3 & & \\
2: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 0 & \\
3: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 0 & R \mapsto 10 \\
4: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 0 & R \mapsto 10 \\
5: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 0 & R \mapsto 7 \\
6: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 1 & R \mapsto 7 \\
4: & A \mapsto 10 & B \mapsto 3 & Q \mapsto 1 & R \mapsto 7
\end{array}
$$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

$^{1}: A \mapsto 10 \quad B \mapsto 3$

$^{2}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

$^{3}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

$^{4}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

$^{5}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 7$

$^{6}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

$^{4}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

$^{5}: A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 4$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1:	$A \mapsto 10$	$B \mapsto 3$		
2:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	
3:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 7$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 4$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 4$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 7$

6: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 4$

6: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 2 \quad R \mapsto 4$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 2 \quad R \mapsto 4$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1: $A \mapsto 10 \quad B \mapsto 3$

2: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0$

3: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 10$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 0 \quad R \mapsto 7$

6: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 7$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 1 \quad R \mapsto 4$

6: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 2 \quad R \mapsto 4$

4: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 2 \quad R \mapsto 4$

5: $A \mapsto 10 \quad B \mapsto 3 \quad Q \mapsto 2 \quad R \mapsto 1$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1:	$A \mapsto 10$	$B \mapsto 3$		
2:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	
3:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 7$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 4$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 4$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 4$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 1$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 3$	$R \mapsto 1$

Execution Traces

most straightforward way to model the software behavior

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1:	$A \mapsto 10$	$B \mapsto 3$		
2:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	
3:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 10$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 0$	$R \mapsto 7$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 7$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 1$	$R \mapsto 4$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 4$
4:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 4$
5:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 2$	$R \mapsto 1$
6:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 3$	$R \mapsto 1$
7:	$A \mapsto 10$	$B \mapsto 3$	$Q \mapsto 3$	$R \mapsto 1$

Execution Traces = Not Feasible

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

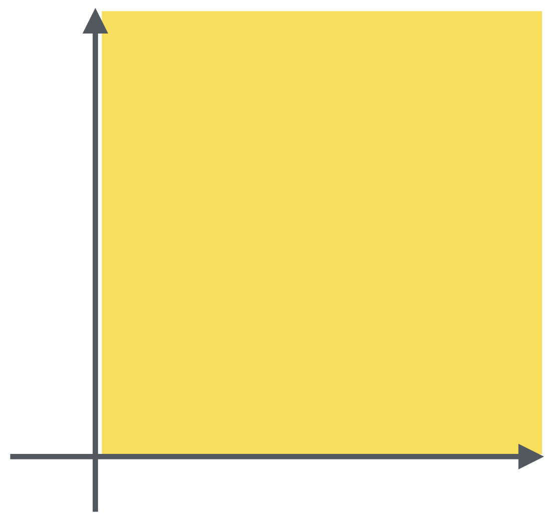
1: $A \mapsto 10 \quad B \mapsto 3$

⚠ one execution trace **for each value of A and B**

static analyzer

Sign Analysis

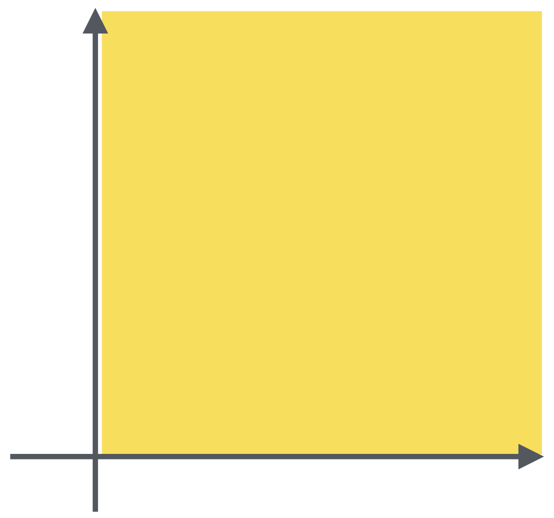
replaces actual *concrete values* with *abstract sign values*



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

Sign Analysis

replaces actual *concrete values* with *abstract* sign values

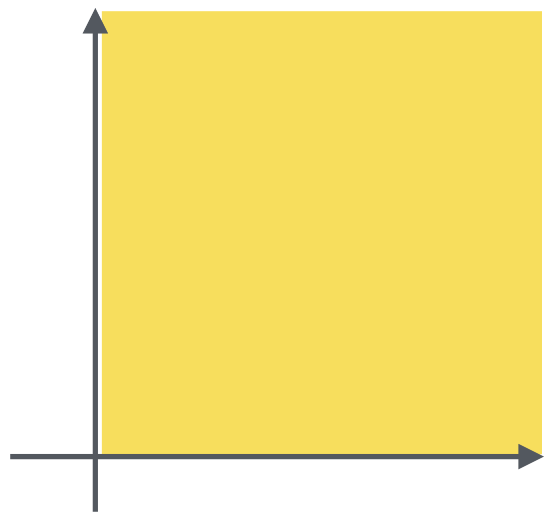


```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

$^{1}: A \mapsto \geq 0 \quad B \mapsto \geq 0$

Sign Analysis

replaces actual *concrete values* with *abstract sign values*



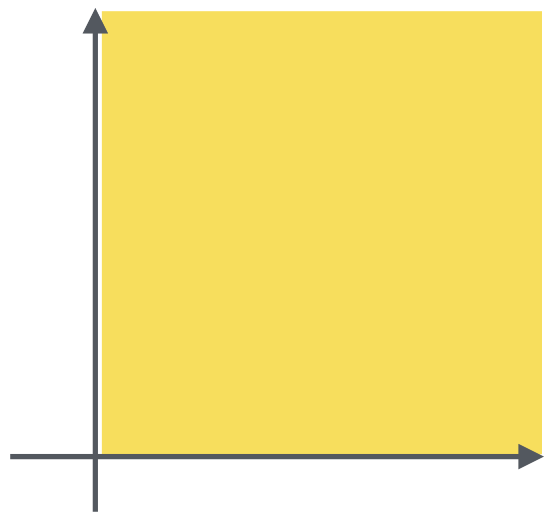
```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

represents multiple concrete executions

Sign Analysis

replaces actual *concrete* values with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

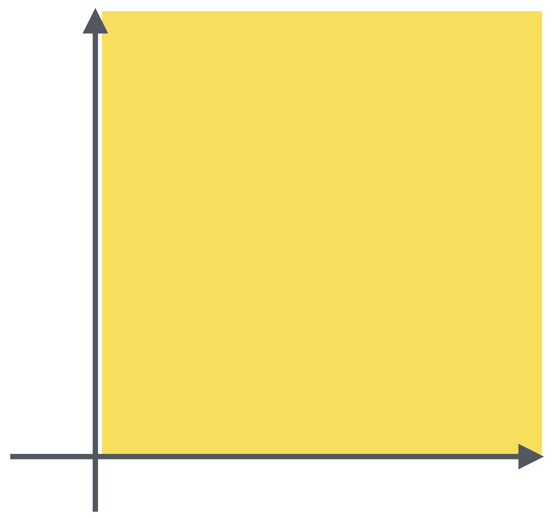
2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

represents multiple concrete executions

Antoine Miné - Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation (FnTPL 2017)

Sign Analysis

replaces actual *concrete* values with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

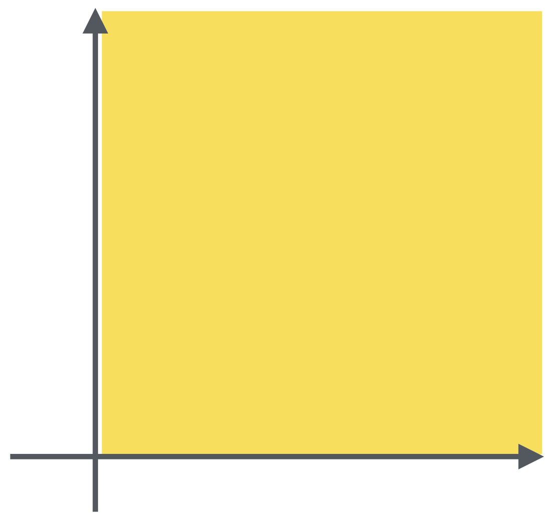
2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

represents multiple concrete executions

Sign Analysis

replaces actual *concrete values* with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

$^{1}: A \mapsto \geq 0 \quad B \mapsto \geq 0$

represents multiple concrete executions

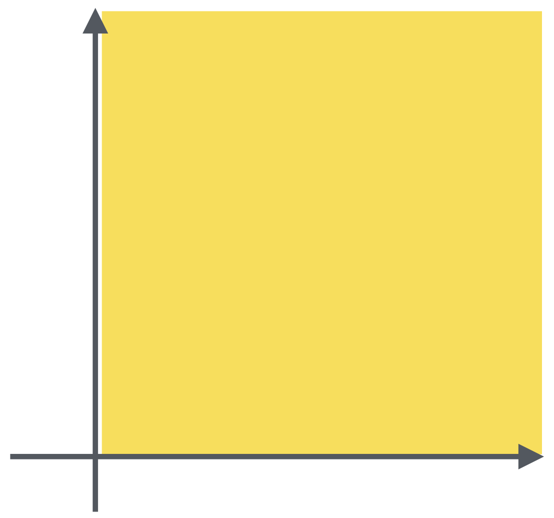
$^{2}: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

$^{3}: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

$^{4}: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

Sign Analysis

replaces actual *concrete* values with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

represents multiple concrete executions

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

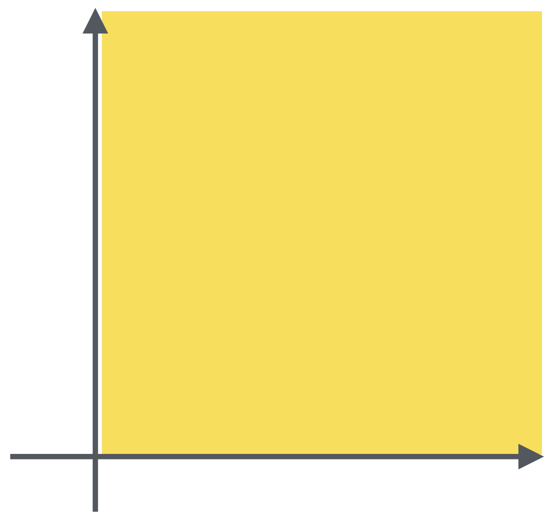
3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

Sign Analysis

replaces actual *concrete* values with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

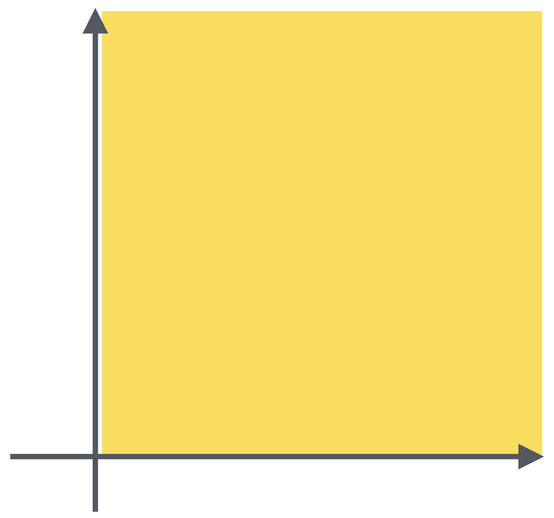
represents multiple concrete executions

abstract interpretation using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis

replaces actual *concrete values* with *abstract sign values*



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

6: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

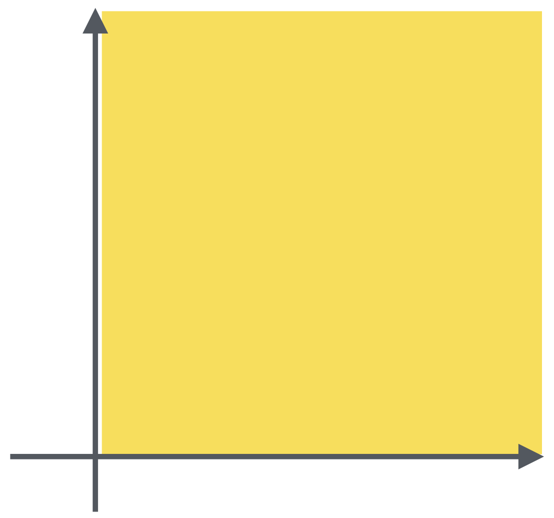
represents multiple concrete executions

abstract interpretation
using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis

replaces actual *concrete values* with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

6: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

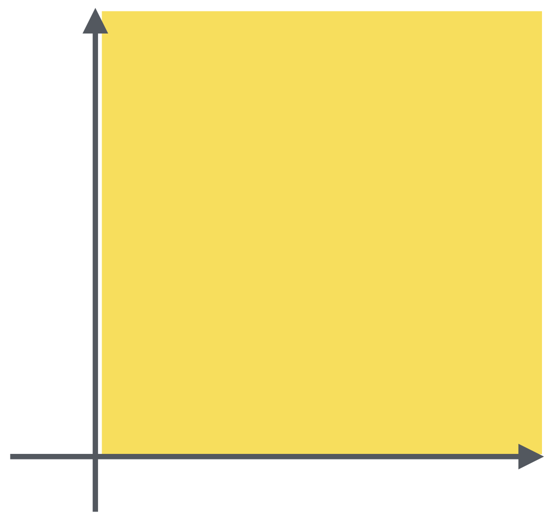
represents multiple concrete executions

abstract interpretation
using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis

replaces actual *concrete values* with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$ **represents multiple concrete executions**

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

6: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

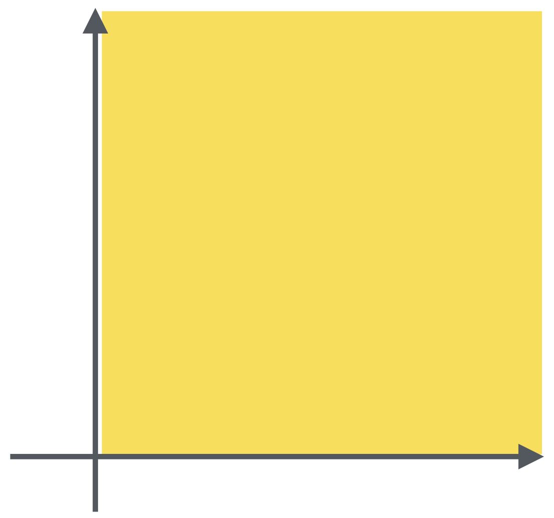
5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

abstract interpretation
using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis

replaces actual *concrete* values with *abstract* sign values



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

$1: A \mapsto \geq 0 \quad B \mapsto \geq 0$

represents multiple concrete executions

$2: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

$3: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

$4: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

$5: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

$6: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

$4: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

$5: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

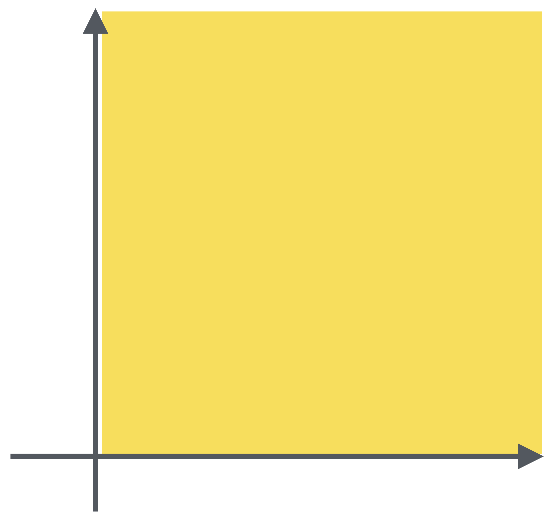
$6: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

abstract interpretation
using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis

replaces actual *concrete values* with *abstract sign values*



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

represents multiple concrete executions

1: $A \mapsto \geq 0 \quad B \mapsto \geq 0$

2: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0$

3: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \geq 0$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto 0 \quad R \mapsto \top$

6: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

4: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

5: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

6: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

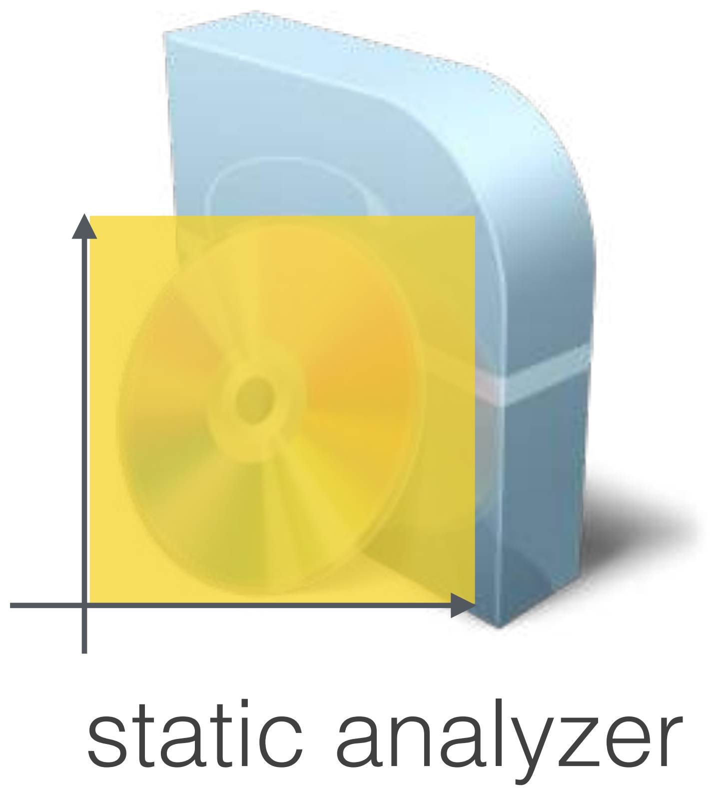
7: $A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

abstract interpretation
using the *rules of signs*:

- $(\geq 0) - (\geq 0) = \top$
- $(\geq 0) + (\geq 0) = \geq 0$

Sign Analysis = **Not Precise Enough**

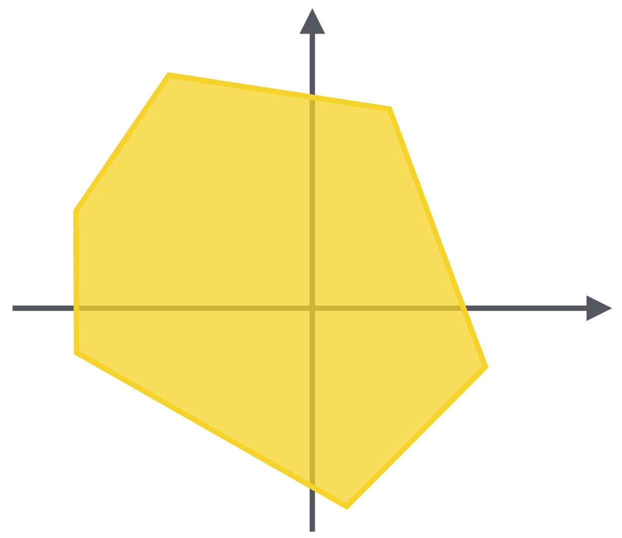
```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```



$7: A \mapsto \geq 0 \quad B \mapsto \geq 0 \quad Q \mapsto \geq 0 \quad R \mapsto \top$

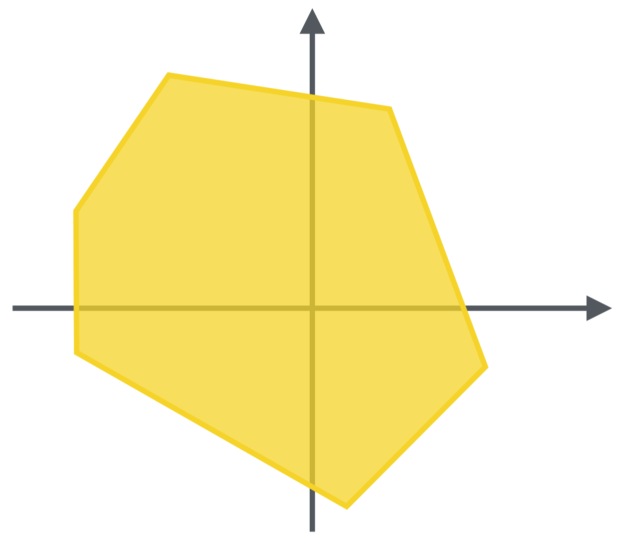
Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

Linear Inequalities Analysis



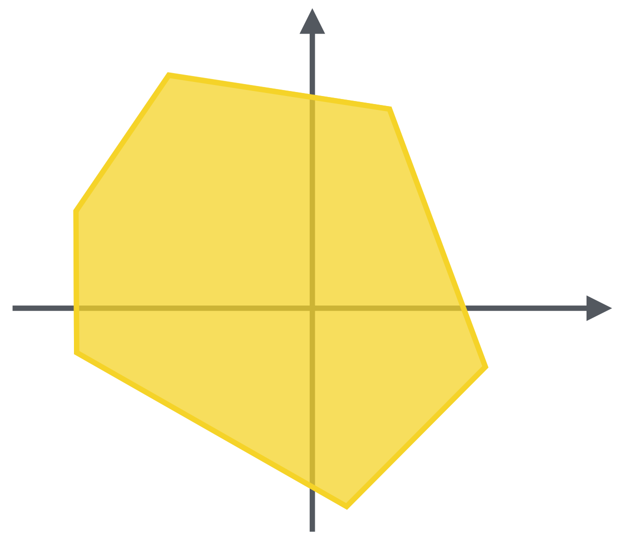
replaces actual *concrete values* with *abstract* linear inequalities between variables

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \geq 0 \quad B \geq 0$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables



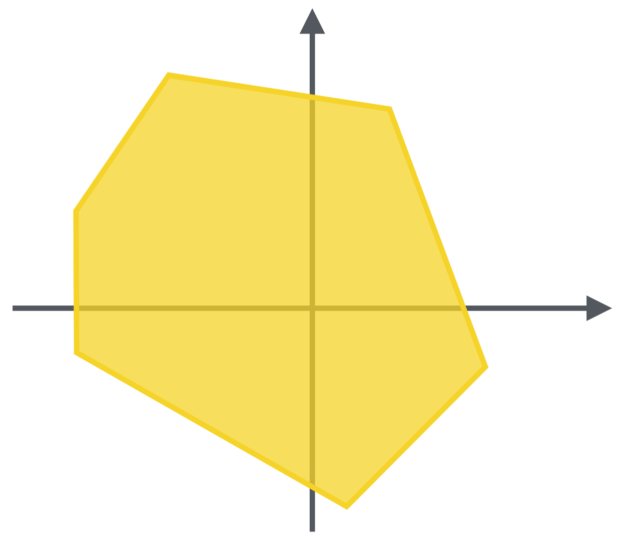
```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \geq 0 \quad B \geq 0$

2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract linear inequalities between variables*



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

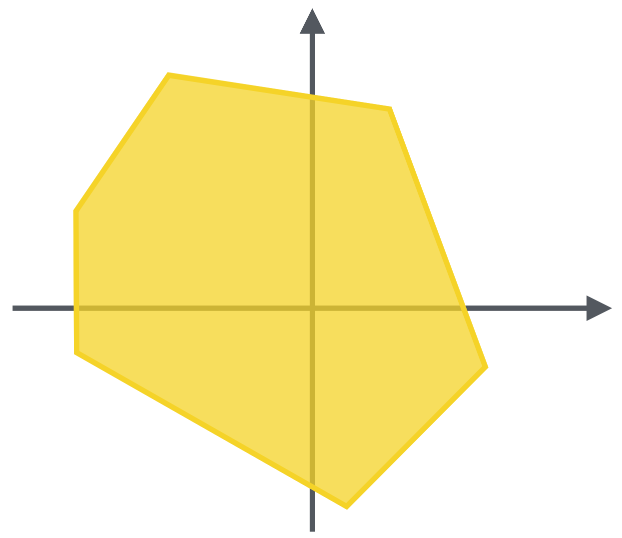
1: $A \geq 0 \quad B \geq 0$

2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

3: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R = A$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1: $A \geq 0 \quad B \geq 0$

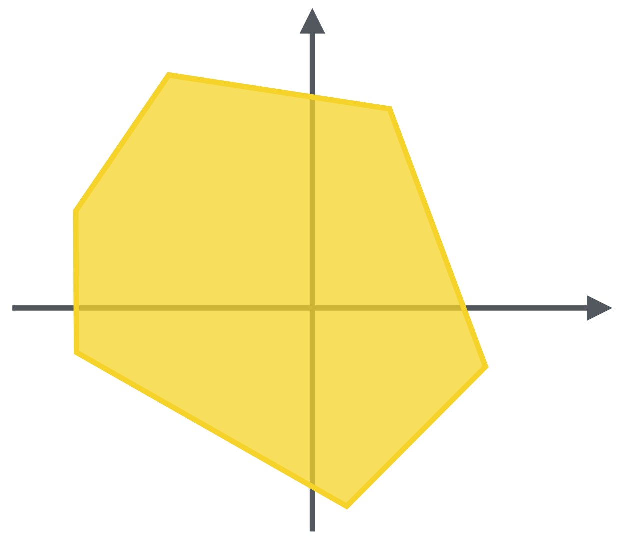
2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

3: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R = A$

4: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq B$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables



requires A ≥ 0 and B ≥ 0

```
def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R
```

ensures R ≥ 0

1: $A \geq 0 \quad B \geq 0$

2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

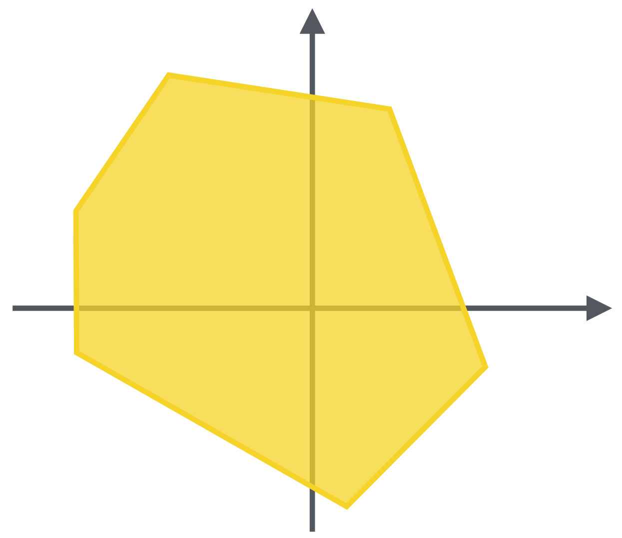
3: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R = A$

4: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq B$

5: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq 0$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract linear inequalities between variables*



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
   1
   Q = 0
   2
   R = A
   3
   while R >= B:
      4
      R = R - B
      5
      Q = Q + 1
      6
   7 return R

ensures R ≥ 0
```

1: $A \geq 0 \quad B \geq 0$

2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

3: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R = A$

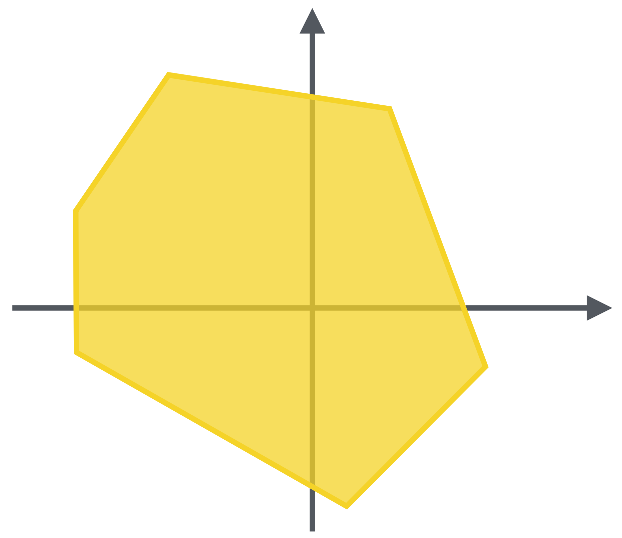
4: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq B$

5: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq 0$

6: $A \geq 0 \quad B \geq 0 \quad Q = 1 \quad R \geq 0$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables



```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1: $A \geq 0 \quad B \geq 0$

2: $A \geq 0 \quad B \geq 0 \quad Q = 0$

3: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R = A$

4: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq B$

5: $A \geq 0 \quad B \geq 0 \quad Q = 0 \quad R \geq 0$

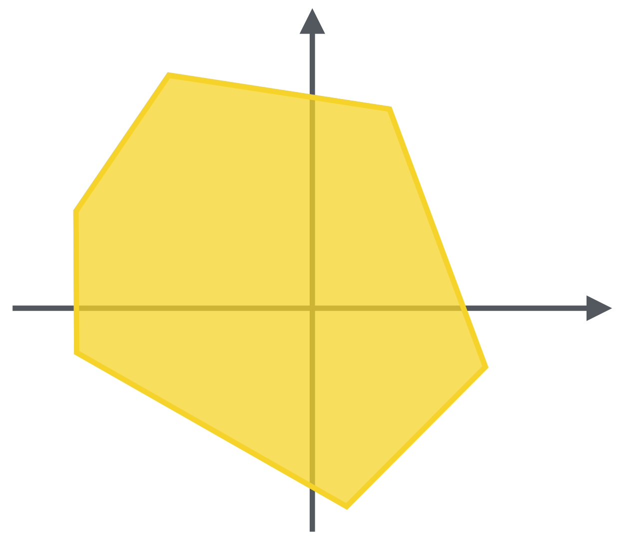
6: $A \geq 0 \quad B \geq 0 \quad Q = 1 \quad R \geq 0$

4: $A \geq 0 \quad B \geq 0 \quad Q \geq 0 \quad R \geq B$

Antoine Miné - Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation (FnTPL 2017)

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract linear inequalities between variables*

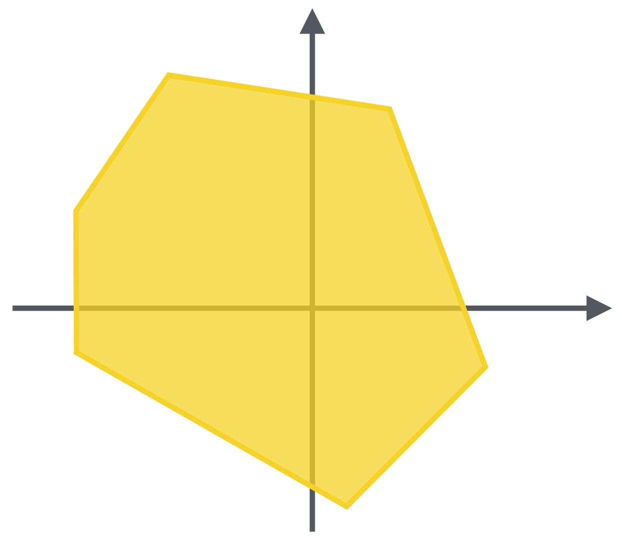


```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1:	$A \geq 0$	$B \geq 0$		
2:	$A \geq 0$	$B \geq 0$	$Q = 0$	
3:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R = A$
4:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq 0$
6:	$A \geq 0$	$B \geq 0$	$Q = 1$	$R \geq 0$
4:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq 0$

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract linear inequalities between variables*



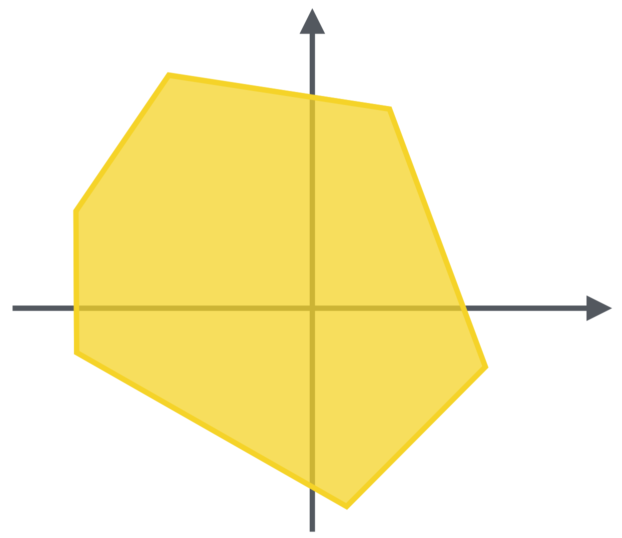
```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

1:	$A \geq 0$	$B \geq 0$		
2:	$A \geq 0$	$B \geq 0$	$Q = 0$	
3:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R = A$
4:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq 0$
6:	$A \geq 0$	$B \geq 0$	$Q = 1$	$R \geq 0$
4:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq 0$
6:	$A \geq 0$	$B \geq 0$	$Q \geq 1$	$R \geq 0$

Antoine Miné - Tutorial on Static Inference of Numeric Invariants by Abstract Interpretation (FnTPL 2017)

Linear Inequalities Analysis

replaces actual *concrete values* with *abstract* linear inequalities between variables

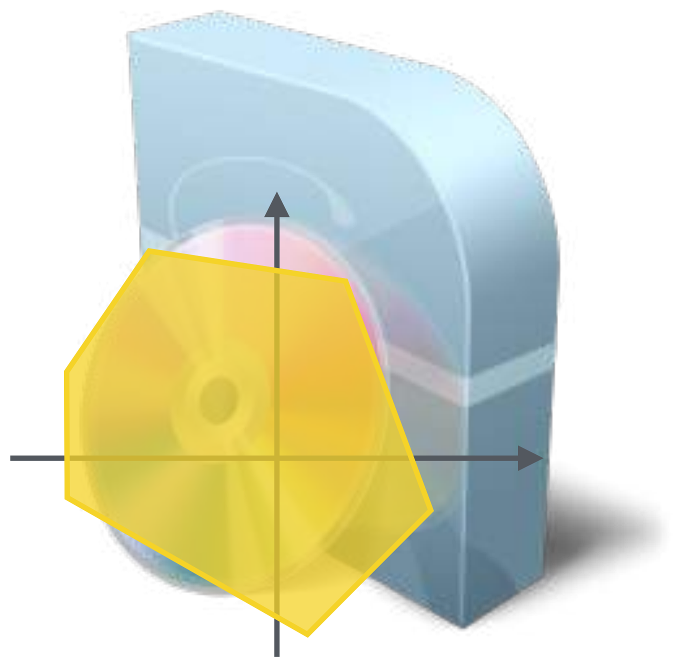


```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
    1
    Q = 0
    2
    R = A
    3
    while R >= B:
        4
        R = R - B
        5
        Q = Q + 1
        6
    7 return R

ensures R ≥ 0
```

1:	$A \geq 0$	$B \geq 0$		
2:	$A \geq 0$	$B \geq 0$	$Q = 0$	
3:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R = A$
4:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q = 0$	$R \geq 0$
6:	$A \geq 0$	$B \geq 0$	$Q = 1$	$R \geq 0$
4:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq B$
5:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$R \geq 0$
6:	$A \geq 0$	$B \geq 0$	$Q \geq 1$	$R \geq 0$
7:	$A \geq 0$	$B \geq 0$	$Q \geq 0$	$0 \leq R < B$

Linear Inequalities Analysis = ✓

```
requires A ≥ 0 and B ≥ 0

def mod(A, B):
  1
  Q = 0
  2
  R = A
  3
  while R >= B:
    4
    R = R - B
    5
    Q = Q + 1
    6
  7 return R

ensures R ≥ 0
```

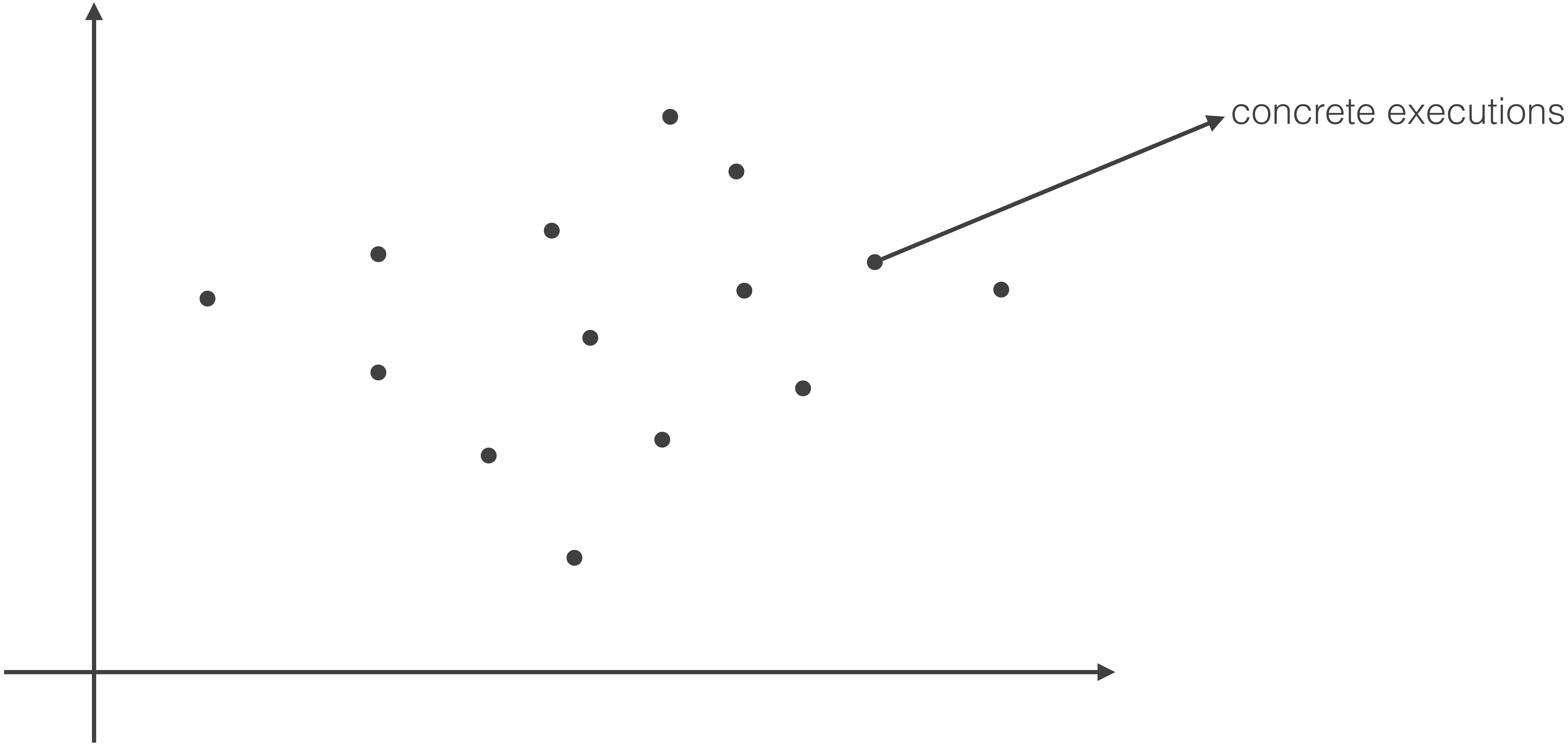


static analyzer

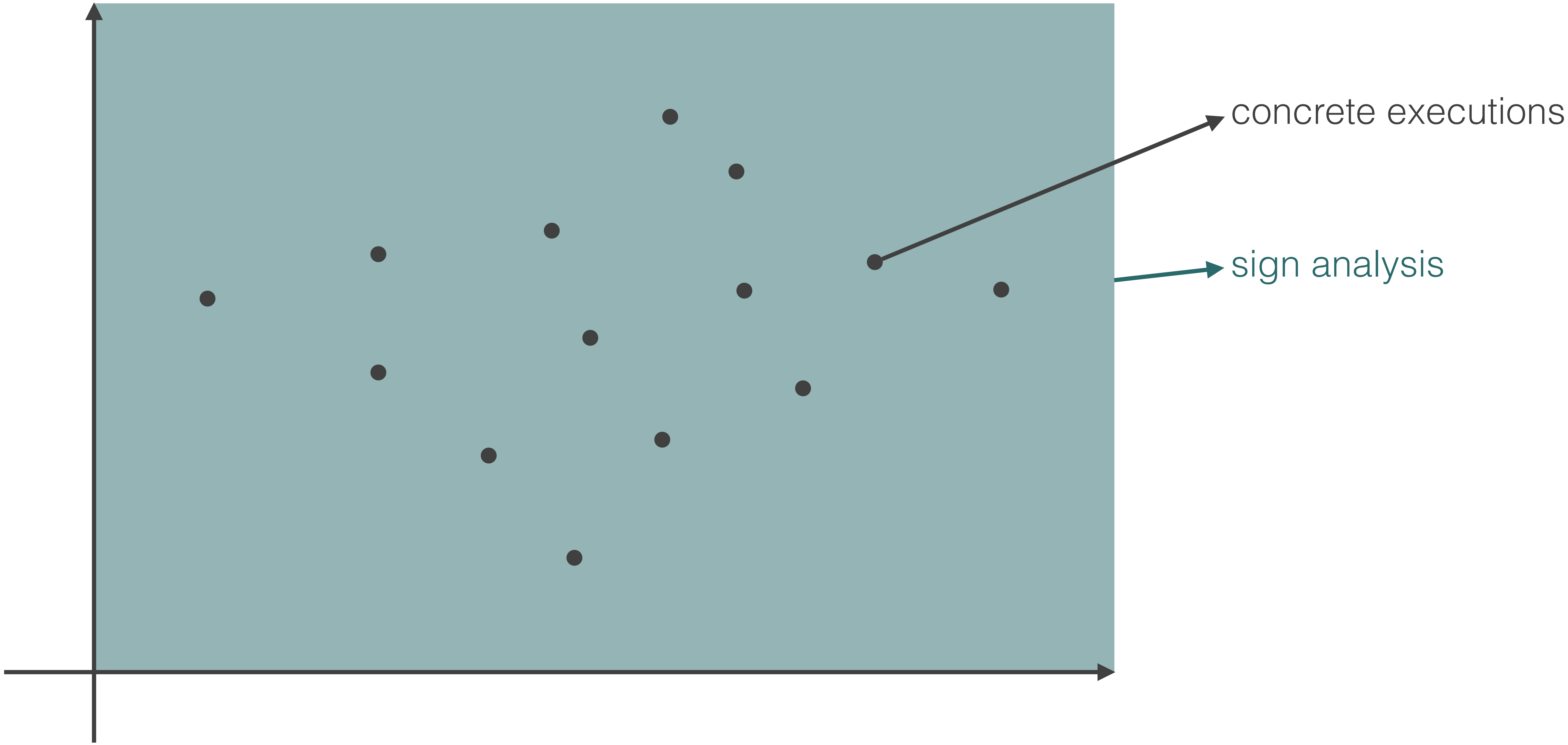
✓

$7: A \geq 0 \quad B \geq 0 \quad Q \geq 0 \quad \mapsto \mathbf{0 \leq R} < B$

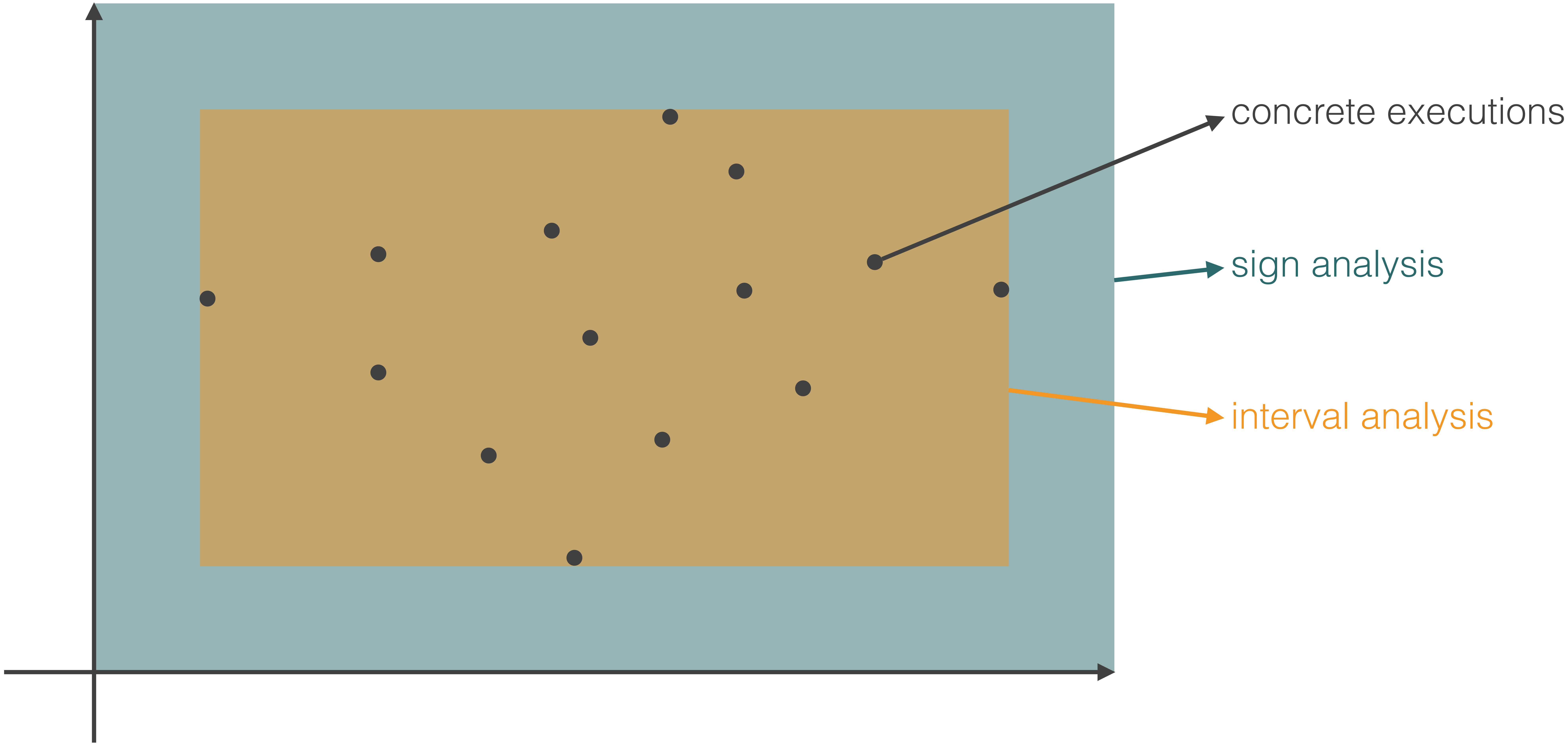
Precision-vs-Cost Tradeoff



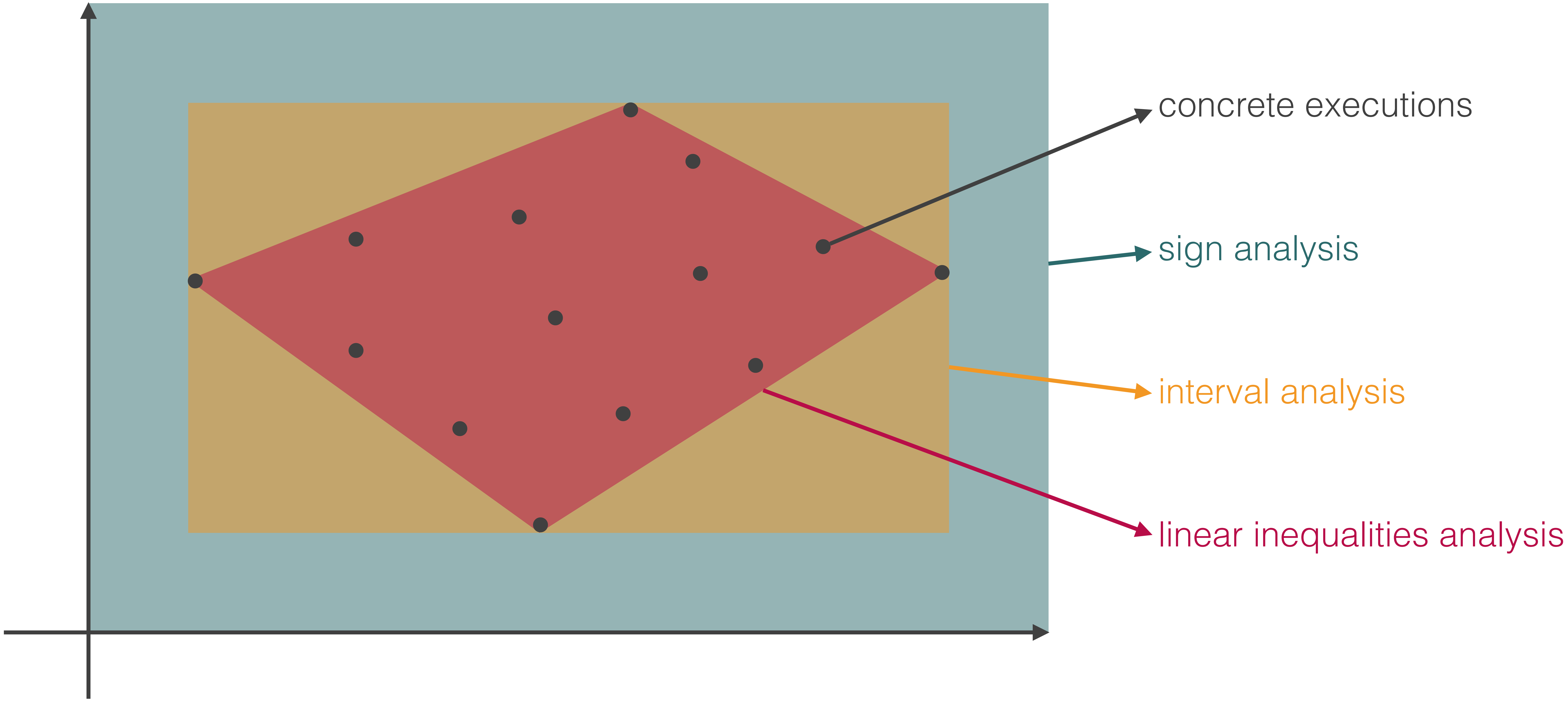
Precision-vs-Cost Tradeoff



Precision-vs-Cost Tradeoff



Precision-vs-Cost Tradeoff



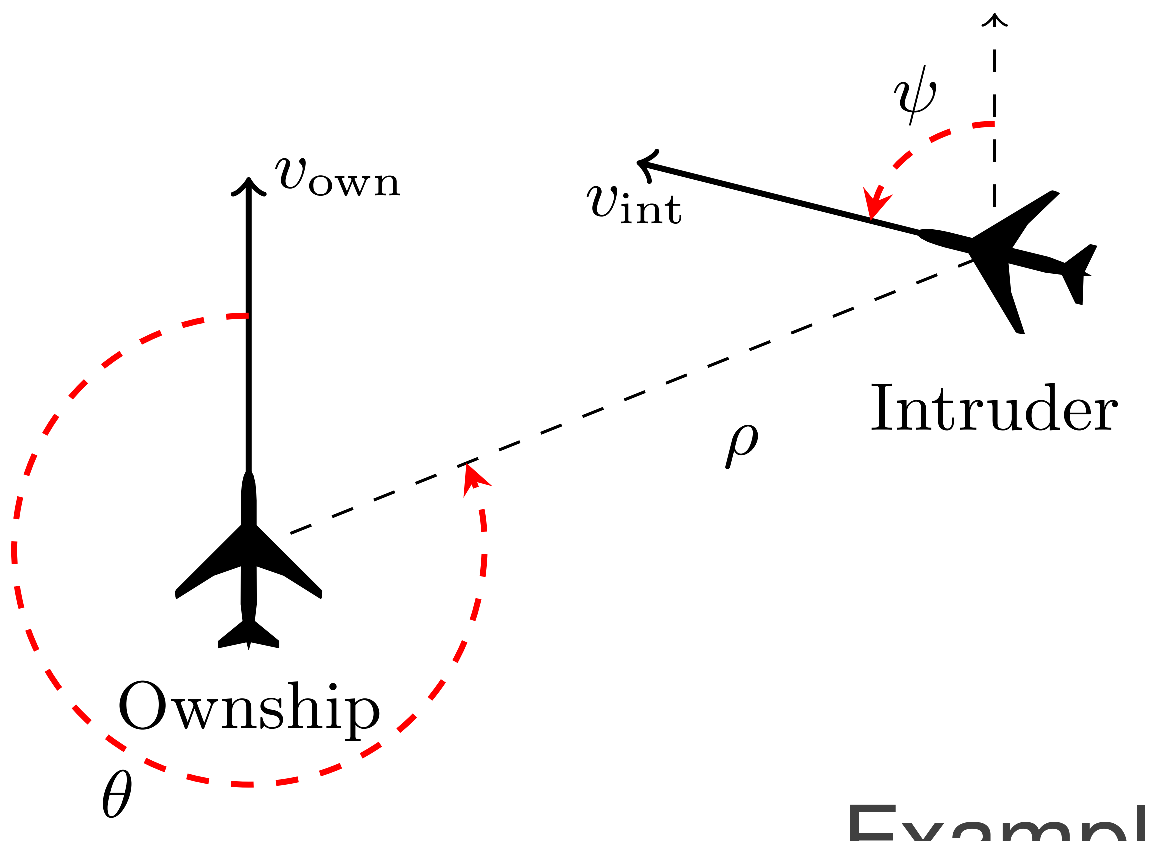
Static Analysis

(Data Science-Related) Examples

Functional Properties

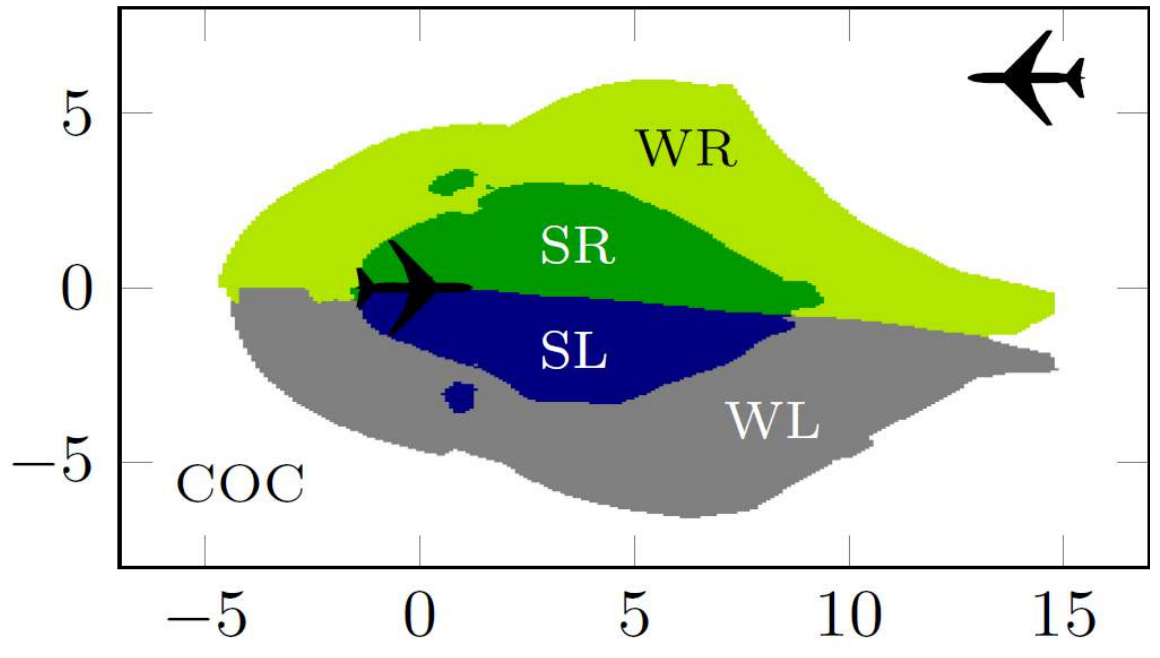
ACAS Xu Neural Networks

collision-avoidance system for drones implemented using **45 feed-forward neural networks**



produce **advisories**:

- **Strong Left**
- **Weak Left**
- **Strong Right**
- **Weak Right**
- **Clear of Conflict**



Example:

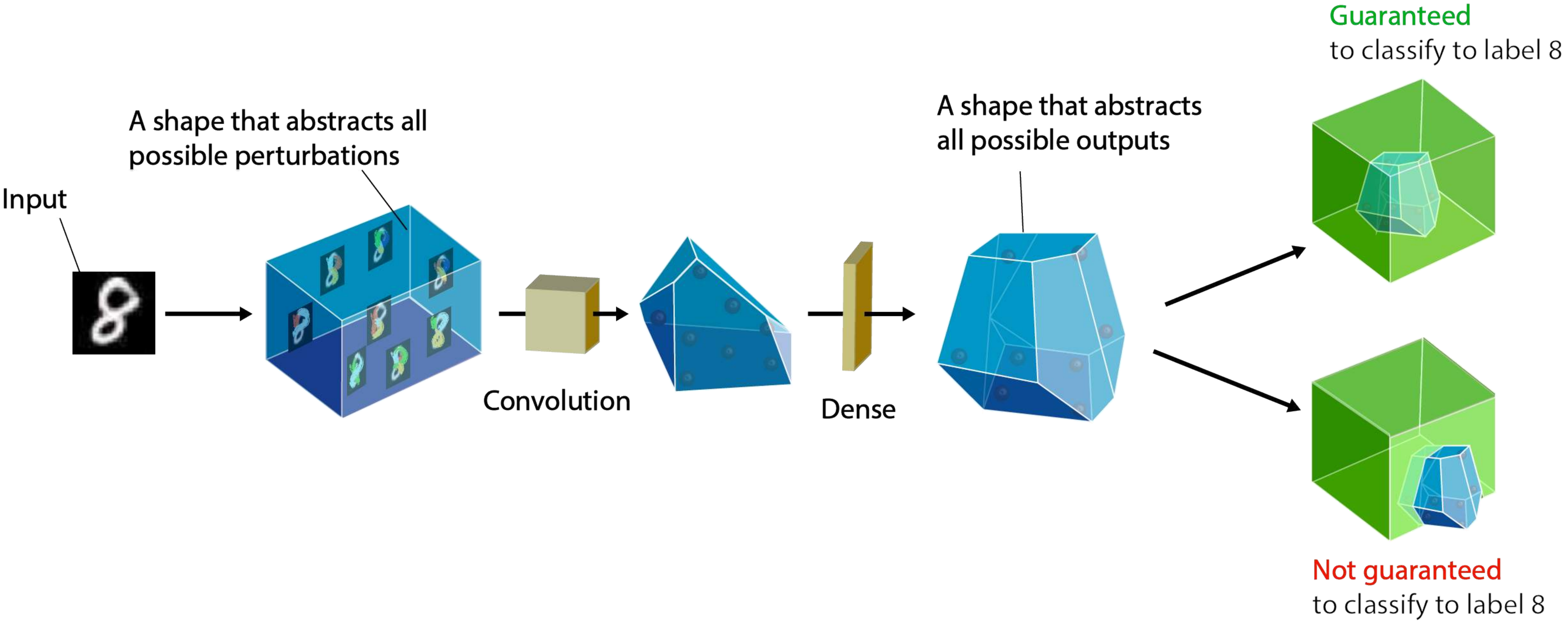
"If the intruder is *near* and approaching *from the left*, the network advises **Strong Right**"

- distance: $12000 \leq \rho \leq 62000$
- angle to intruder: $0.2 \leq \theta \leq 0.4$
- …

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)

Local Robustness

Neural Networks



T. Gehr et al. - AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation (IEEE S&P 2018)

Local Robustness

Support Vector Machines and Random Forests

Robustness Verification of Support Vector Machines

Francesco Ranzato[0000−0003−0159−0068] and Marco Zanella

Dipartimento di Matematica, University of Padova, Italy

Abstract. We study the problem of formally verifying the robustness to adversarial examples of support vector machines (SVMs), a major machine learning model for classification and regression tasks. Following a recent stream of works on formal robustness verification of (deep) neural networks, our approach relies on a sound abstract version of a given SVM classifier to be used for checking its robustness. This methodology is parametric on a given numerical abstraction of real values and, analogously to the case of neural networks, needs neither abstract least upper bounds nor widening operators on this abstraction. The standard interval domain provides a simple instantiation of our abstraction technique, which is enhanced with the domain of reduced affine forms, an efficient abstraction of the zonotope abstract domain. This robustness verification technique has been fully implemented and experimentally evaluated on SVMs based on linear and nonlinear (polynomial and radial basis function) kernels, which have been trained on the popular MNIST dataset of images and on the recent and more challenging Fashion-MNIST dataset. The experimental results of our prototype SVM robustness verifier appear to be encouraging: this automated verification is fast, scalable and shows significantly high percentages of provable robustness on the test set of MNIST, in particular compared to the analogous provable robustness of neural networks.

1 Introduction

Adversarial machine learning [10,17,38] is an emerging hot topic studying vulnerabilities of machine learning (ML) techniques in adversarial scenarios and whose main objective is to design methodologies for making learning tools robust to adversarial attacks. Adversarial examples have been found in diverse application fields of ML such as image classification, speech recognition and malware detection [10]. Current

Abstract Interpretation of Decision Tree Ensemble Classifiers

Francesco Ranzato, Marco Zanella
Dipartimento di Matematica, University of Padova, Italy
{ranzato, mzanella}@math.unipd.it

Abstract

We study the problem of formally and automatically verifying robustness properties of decision tree ensemble classifiers such as random forests and gradient boosted decision tree models. A recent stream of works showed how abstract interpretation, which is ubiquitously used in static program analysis, can be successfully deployed to formally verify (deep) neural networks. In this work we push forward this line of research by designing a general and principled abstract interpretation-based framework for the formal verification of robustness and stability properties of decision tree ensemble models. Our abstract interpretation-based method may induce complete robustness checks of standard adversarial perturbations and output concrete adversarial attacks. We implemented our abstract verification technique in a tool called *silva*, which leverages an abstract domain of not necessarily closed real hyperrectangles and is instantiated to verify random forests and gradient boosted decision trees. Our experimental evaluation on the MNIST dataset shows that *silva* provides a precise and efficient tool which advances the current state of the art in tree ensembles verification.

1 Introduction

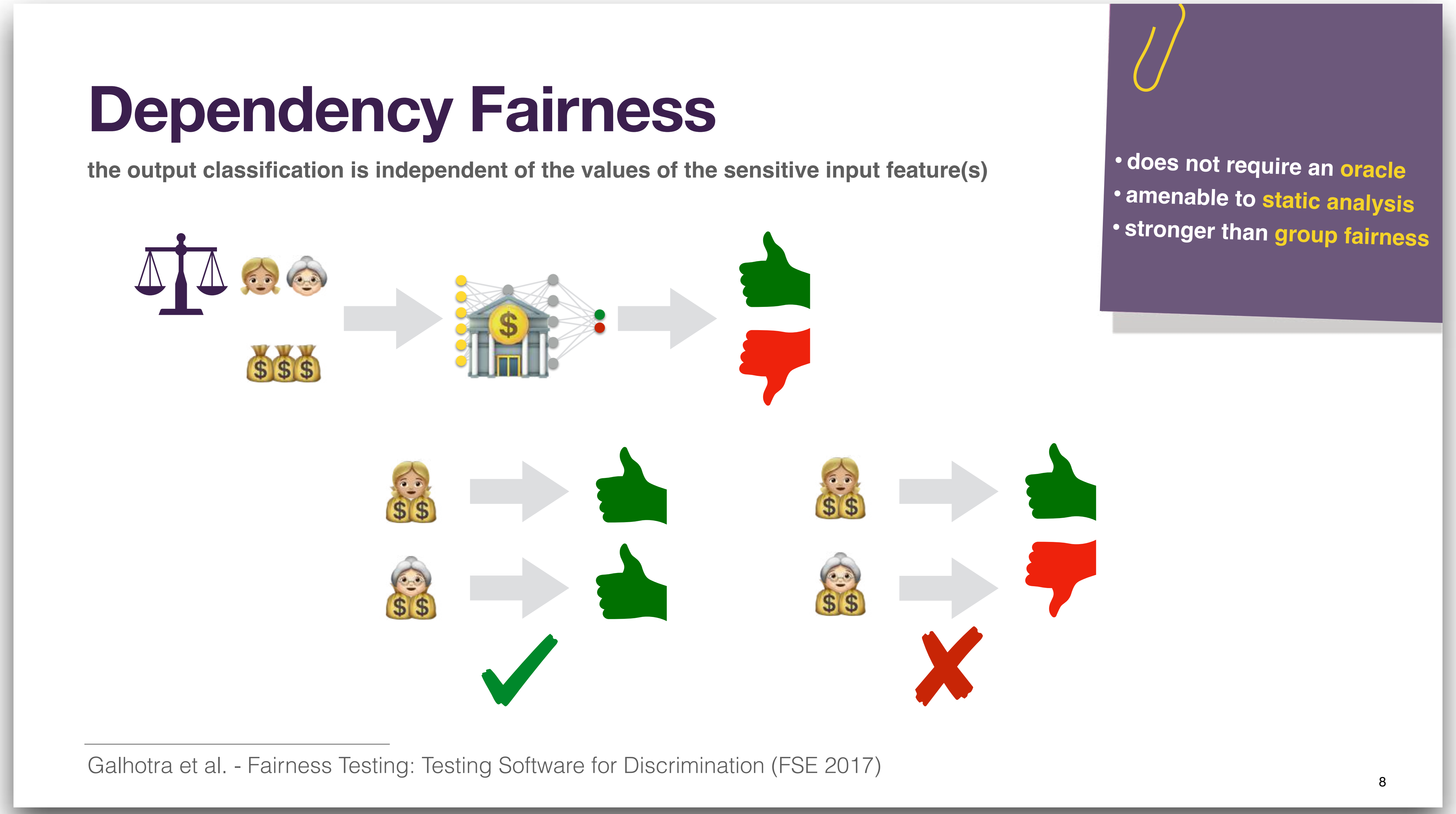
Adversarial machine learning (Goodfellow, McDaniel, and Papernot 2018; Kurakin, Goodfellow, and Bengio 2017) is a hot topic studying vulnerabilities of machine learning (ML) in adversarial scenarios. Adversarial examples have been found in diverse application fields of ML such as image clas-

neural networks may rely on a number of different techniques: linear approximation of functions (Weng et al. 2018; Zhang et al. 2018), semidefinite relaxations (Raghunathan, Steinhardt, and Liang 2018), logical SMT solvers (Huang et al. 2017; Katz et al. 2017), symbolic interval propagation (Wang et al. 2018a), abstract interpretation (Gehr et al. 2018; Singh et al. 2018; 2019) or hybrid synergistic approaches (Anderson et al. 2019; Wang et al. 2018b). Abstract interpretation (Cousot and Cousot 1977) is a *de facto* standard technique used since forty years for designing static analysers and verifiers of programming languages. Recently, abstract interpretation has been successfully applied for designing precise and scalable robustness verification tools of (deep) neural network models (Gehr et al. 2018; Singh et al. 2018; 2019). While all these verification techniques consider neural networks as ML model, in this work we focus on decision tree ensemble methods, such as random forests and gradient boosted decision tree models, which are widely applied in different fields having sensible adversarial scenarios, notably image classification, malware detection, intrusion detection and spam filtering.

Contributions. Following the aforementioned stream of works applying abstract interpretation for certifying ML models, we design a general abstract interpretation-based framework for the formal verification of stability properties of decision tree ensemble models. Our verification algorithm of ensembles of decision trees: (1) is domain agnostic, since it can be instantiated to any abstract domain which repre-

Global Robustness

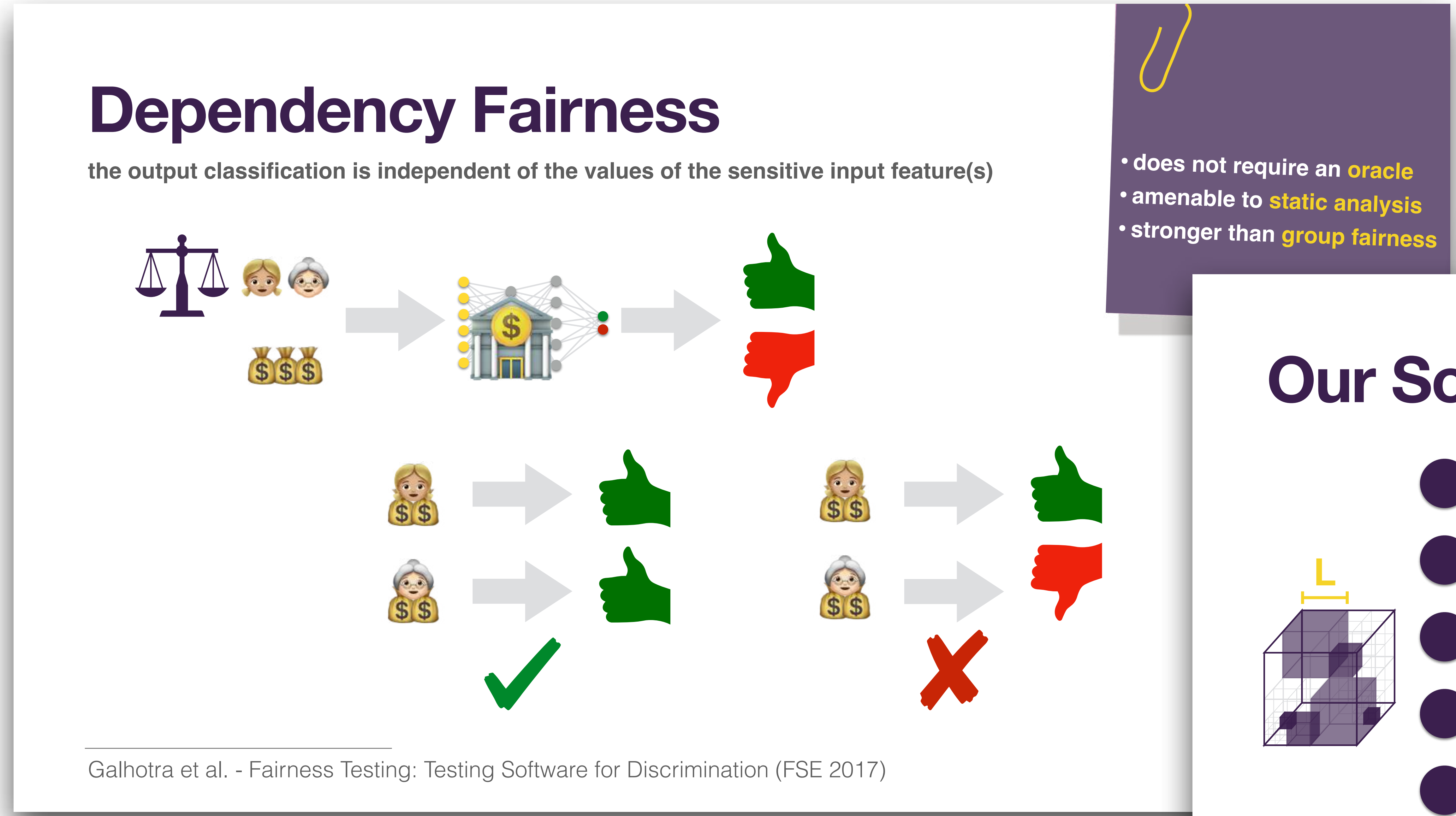
Fairness of Neural Networks

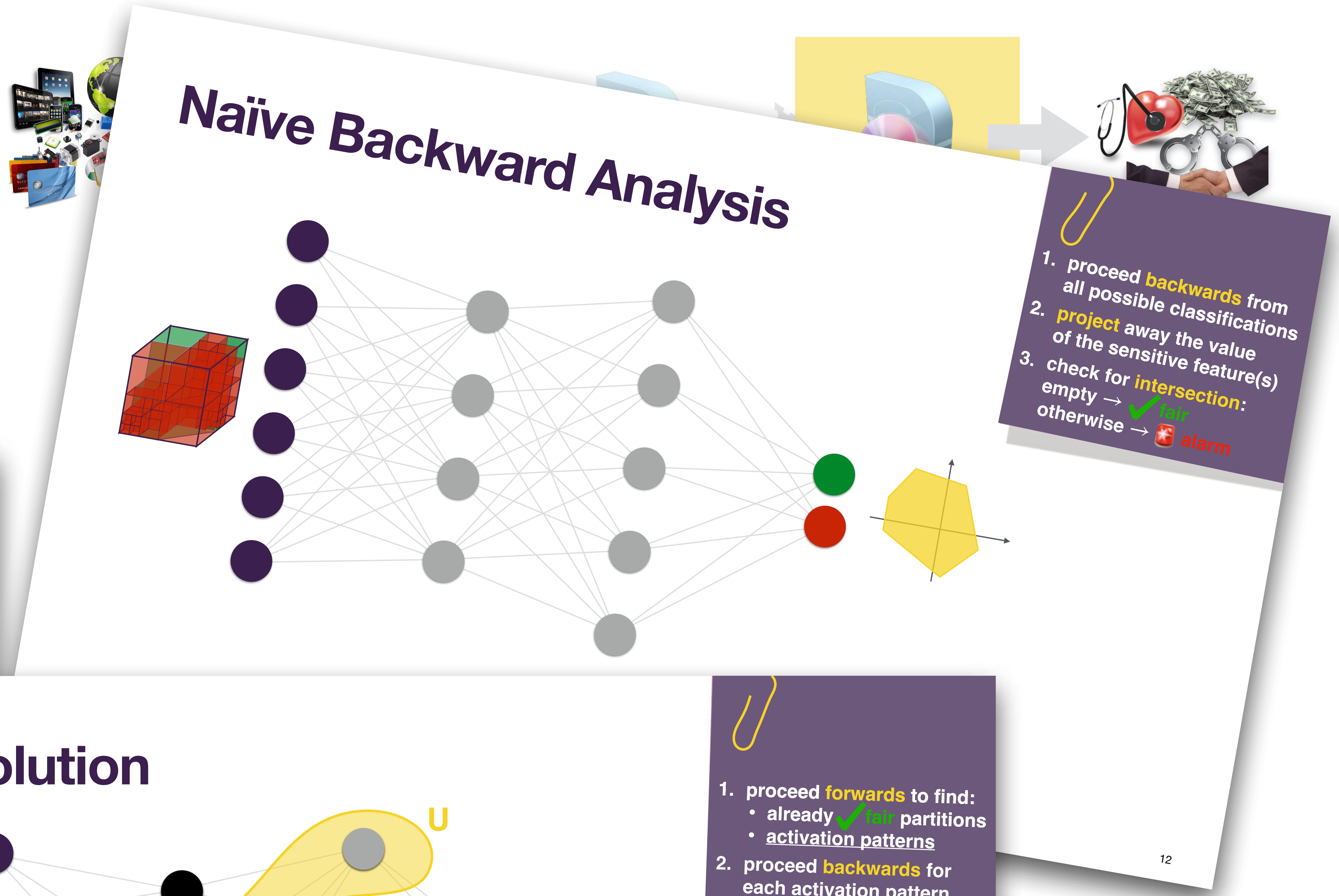


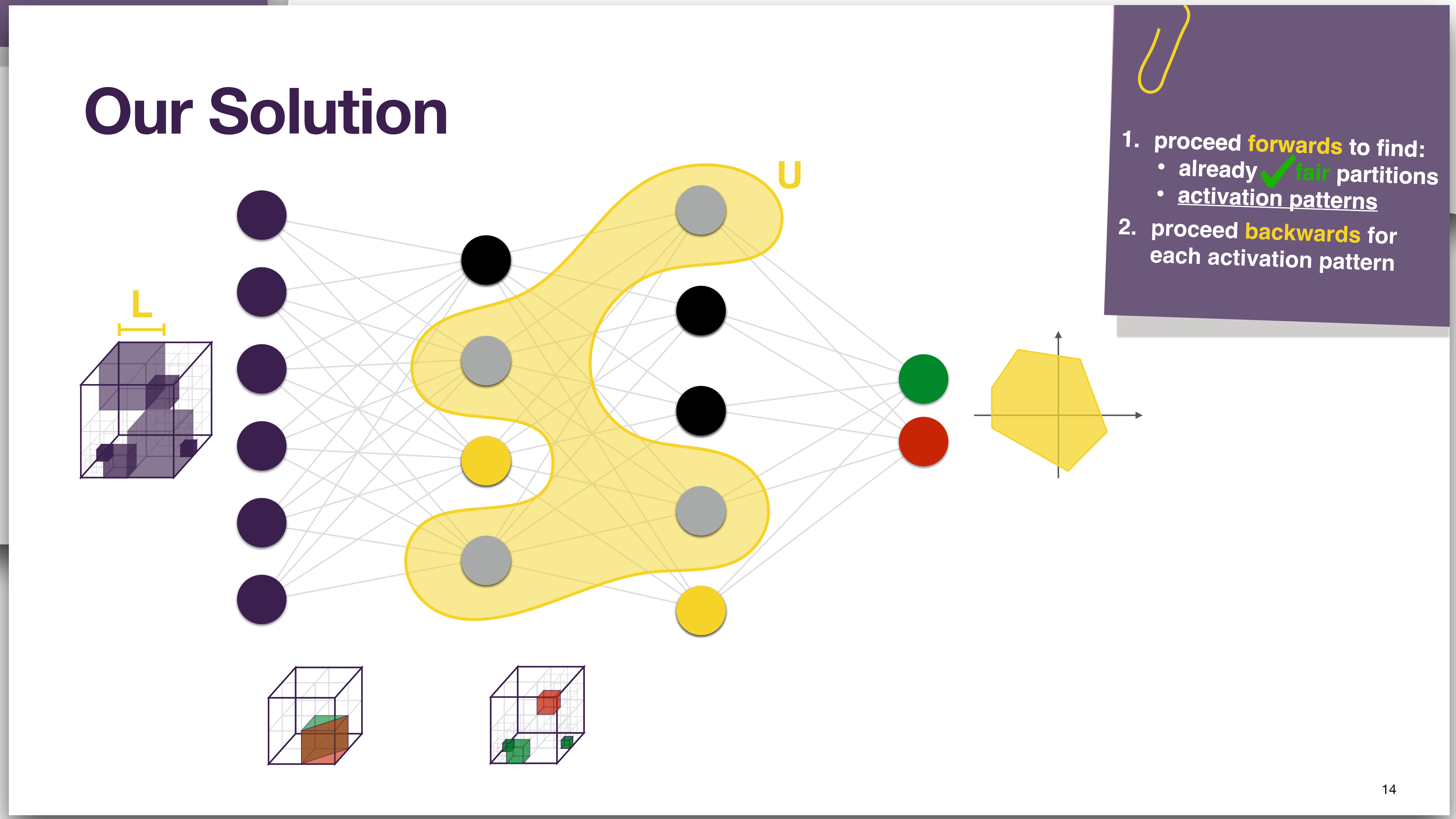
C. **Urban** et al. - Perfectly Parallel Fairness Certification of Neural Networks (OOPSLA 2020)

Global Robustness

Fairness of Neural Networks



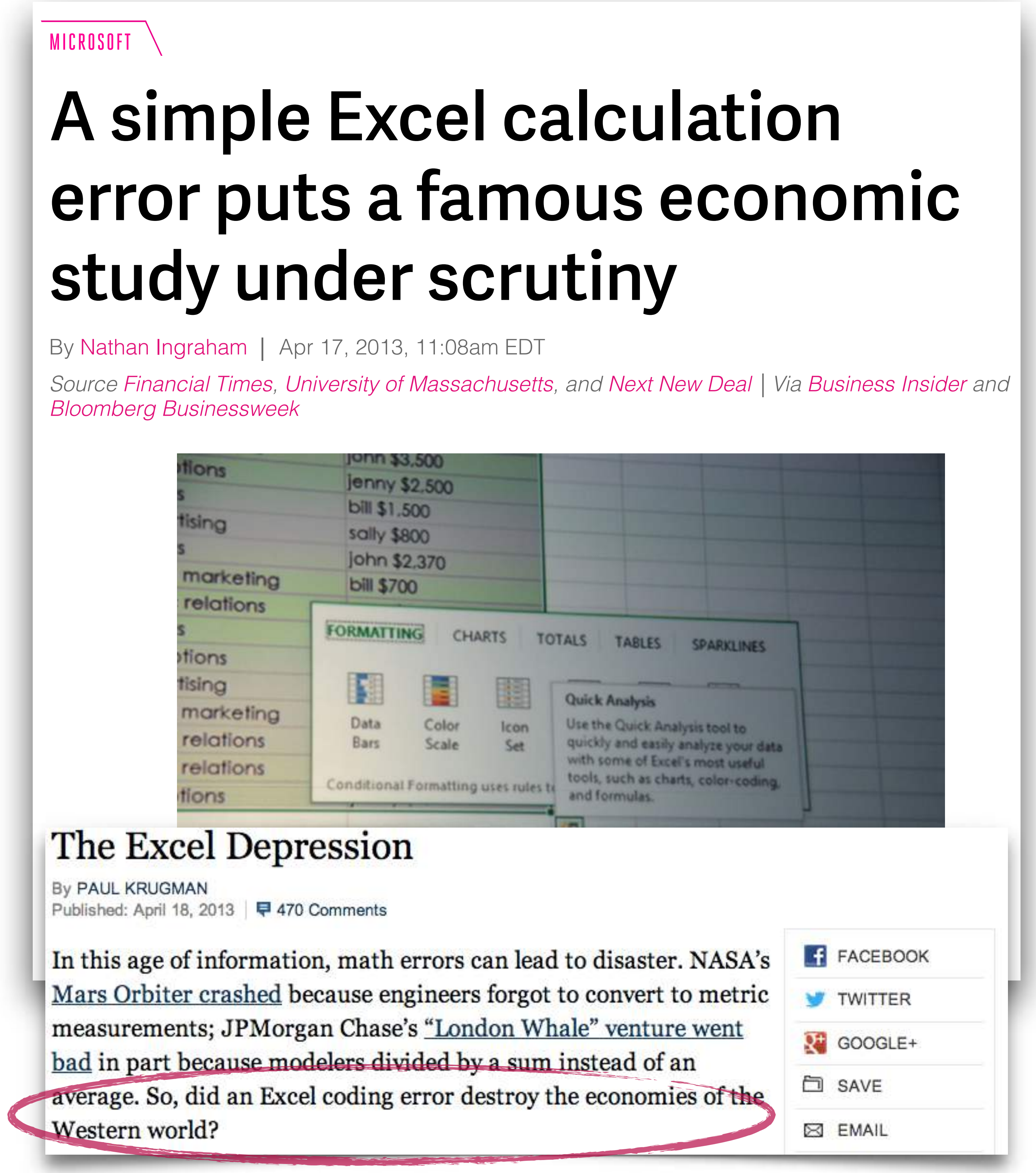




C. **Urban** et al. - Perfectly Parallel Fairness Certification of Neural Networks (OOPSLA 2020)

Input Data Usage

Accidentally Unused Data



MICROSOFT

A simple Excel calculation error puts a famous economic study under scrutiny

By Nathan Ingraham | Apr 17, 2013, 11:08am EDT

Source Financial Times, University of Massachusetts, and Next New Deal | Via Business Insider and Bloomberg Businessweek

The Excel Depression

By PAUL KRUGMAN
Published: April 18, 2013 | 470 Comments

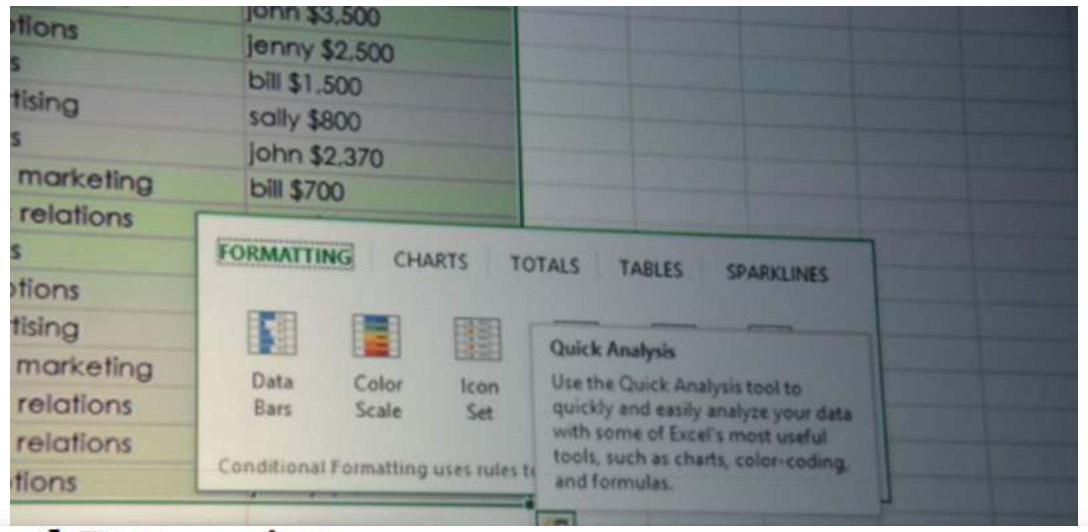
In this age of information, math errors can lead to disaster. NASA's Mars Orbiter crashed because engineers forgot to convert to metric measurements; JPMorgan Chase's "London Whale" venture went bad in part because modelers divided by a sum instead of an average. So, did an Excel coding error destroy the economies of the Western world?

FACEBOOK
TWITTER
GOOGLE+
SAVE
EMAIL

C. **Urban** and P. Müller - An Abstract Interpretation Framework for Data Usage (ESOP 2018)

Input Data Usage

Accidentally Unused Data



MICROSOFT

A simple Excel calculation error puts a famous economic study under scrutiny

By Nathan Ingraham | Apr 17, 2013, 11:08am EDT

Source Financial Times, University of Massachusetts, and Next New Deal | Via Business Insider and Bloomberg Businessweek

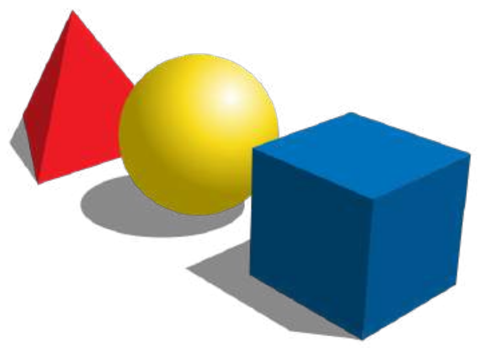
The Excel Depression

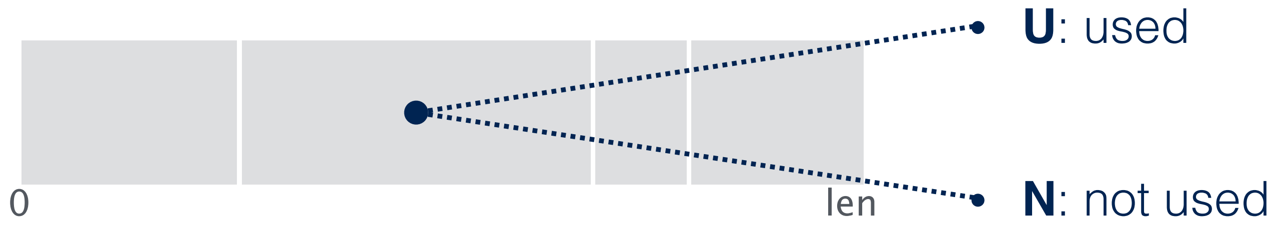
By PAUL KRUGMAN
Published: April 18, 2013 | 470 Comments

In this age of information, math errors can lead to disaster. NASA's Mars Orbiter crashed because engineers forgot to convert to metric measurements; JPMorgan Chase's "London Whale" venture went bad in part because modelers divided by a sum instead of an average. So, did an Excel coding error destroy the economies of the Western world?

FACEBOOK
TWITTER
GOOGLE+
SAVE
EMAIL

Piecewise Unused Input Data Analysis






```
grades = list(map(int, input().split()))   ◄···· INPUT VARIABLE

count = 0
●····································· grades ··→ 0 | N | 1 | U | len(grades)
i = 1   ◄····························· ERROR: 1 SHOULD BE 0
●····································· grades ··→ 0 | N | i | U | i+1 | U | len(grades)
while i < len(grades):
    ●································· grades ··→ 0 | N | i | U | i+1 | U | len(grades)
    if grades[i] < 4:
        count = count + 1
    ●································· grades ··→ 0 | N | i+1 | U | i+2 | U | len(grades)
    i = i + 1
    ●································· grades ··→ 0 | N | i | U | i+1 | U | len(grades)
●····································· grades ··→ 0 | N | len(grades)
if 2 * count < len(grades):
    passing = True
else:
    passing = False
●····································· grades ··→ 0 | N | len(grades)

print(passing)   ◄···················· OUTPUT VARIABLE
```

10

C. Urban and P. Müller - An Abstract Interpretation Framework for Data Usage (ESOP 2018)

Ongoing Work

Implicit Assumptions on the Input Data

Implicit Assumptions

```
import sys

grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
with open(sys.argv[1]) as file:
    for line in file:
        data = line.strip().split(' ')
        grades = int(data[1])
        gpa = 0.0
        for i in range(2, grades + 2):
            gpa += grade2gpa[data[i]]
        result = gpa / grades

        print('{}: {}'.format(data[0], result))
```

```
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
[^\n ] [1-9][0-9]* (A|B|C|D|F){ }
```

Examples

- each line contains a certain number of characters or words
- all characters are uppercase or lowercase
- values can only be in a certain range

```
dkd 2 A C
ndd 1 F
dle 3 C C C
bk2 1 B
wwb 2 D F
wbd 1 D
```

✔

```
dkd 2 A C
ndd 1 F
dle 3.0 C C C
bk2 1 B
wwb 2 D F
wbd 1 D
```

✘

```
dkd 2 A C
ndd 1 F
dle 3 C C C
bk2 1 B+
wwb 2 D F
wbd 1 D
```

✘

```
dkd 2 A C
ndd 1 F
dle 3 C C C
bk2 1 B
wwb 2 D
wbd 1 D
```

✘

```
dkd 1 A C
ndd 1 F
dle 3 C C C
bk2 1 B
wwb 1 D
wbd 2 D F
```

⚠

6

C. Urban - What Programs Want: Automatic Inference of Input Data Specifications (arXiv 2020)

Input Data Usage
Accidentally Unused Data
A simple Excel calculation error puts a famous economic study under scrutiny
The Excel Depression
Piecewise Unused Input Data Analysis
INPUT VARIABLE
OUTPUT VARIABLE
C. Urban and P. Müller - An Abstract Interpretation Framework for Data Usage (ESOP 2018)
Global Robustness
Fairness of Neural Networks
Dependency Fairness
Naïve Backward Analysis
Our Solution
C. Urban et al. - Perfectly Parallel Fairness Certification of Neural Networks (OOPSLA 2020)
Functional Properties
ACAS Xu Neural Networks
collision-avoidance system for drones implemented using 45 feed-forward neural networks
produce advisories:
Strong Left
Weak Left
Strong Right
Weak Right
Clear of Conflict
Ownship
Intruder
Example:
"If the intruder is near and approaching from the left, the network advises Strong Right"
distance: $12000 \leq \rho \leq 62000$
angle to intruder: $0.2 \leq \theta \leq 0.4$
...
z et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)
.LOG
pre-processing
training
data analysis
Ongoing Work
Implicit Assumptions on the Input Data
Implicit Assumptions
Examples
C. Urban - What Programs Want: Automatic Inference of Input Data Specifications (arXiv 2020)
Local Robustness
Neural Networks
Input
A shape that abstracts all possible perturbations
Convolution
Dense
A shape that abstracts all possible outputs
Guaranteed to classify to label 8
Not guaranteed to classify to label 8
T. Gehr et al. - AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation (IEEE S&P 2018)
Local Robustness
Support Vector Machines and Random Forests
Robustness Verification of Support Vector Machines
Abstract Interpretation of Decision Tree Ensemble Classifiers
QUESTIONS?