

# Some decidability questions in abstract program semantics

Luca Zaninotto

Master degree in Computer Science

Università degli studi di Padova

03 Jul 2024

# The cost of software failures



**Figure:** Ariane 5 crash, circa  
370mln\$ in damages

- Testing and careful design might not be enough.
- Formal methods can help, by providing strong mathematical guarantees.

# The cost of software failures

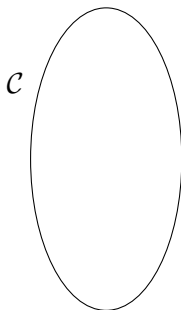


Figure: Ariane 5 crash, circa 370mln\$ in damages

- Testing and careful design might not be enough.
- Formal methods can help, by providing strong mathematical guarantees.
- We focus in particular on **Abstract interpretation**.

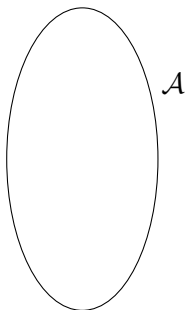
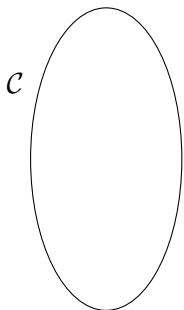
# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



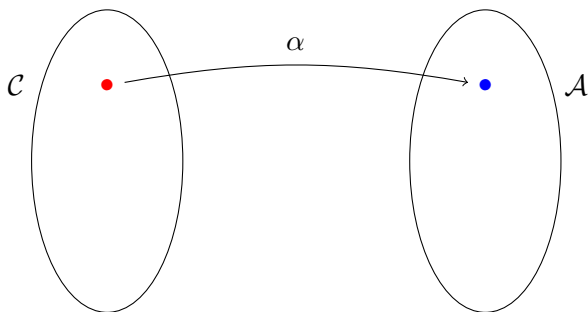
# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



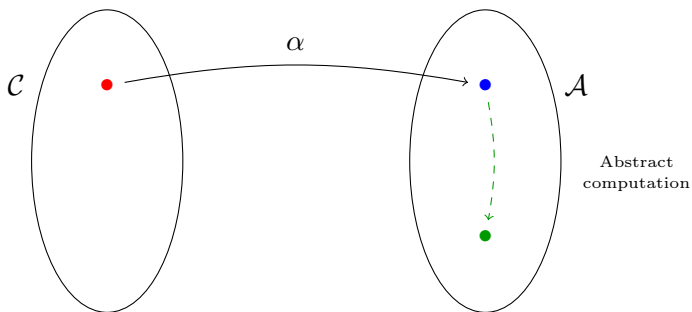
# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



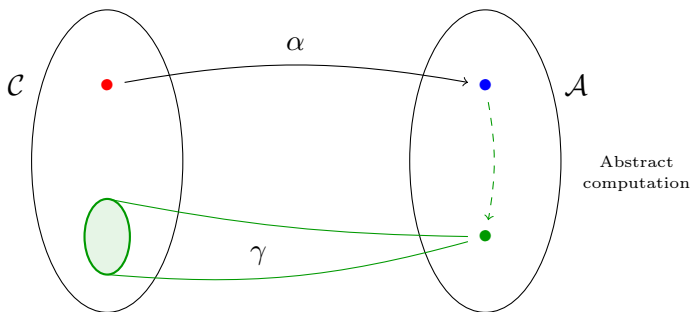
# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



# Abstract interpretation

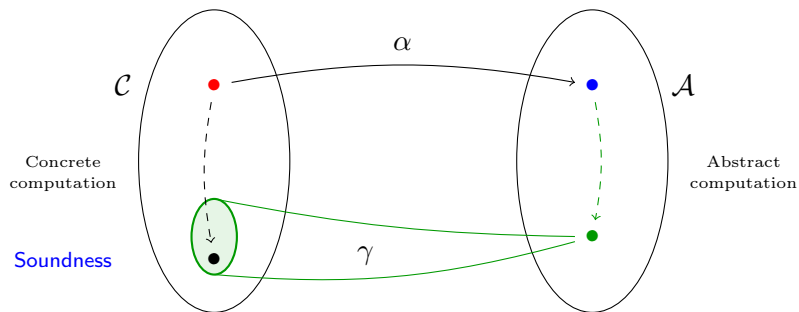
Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*





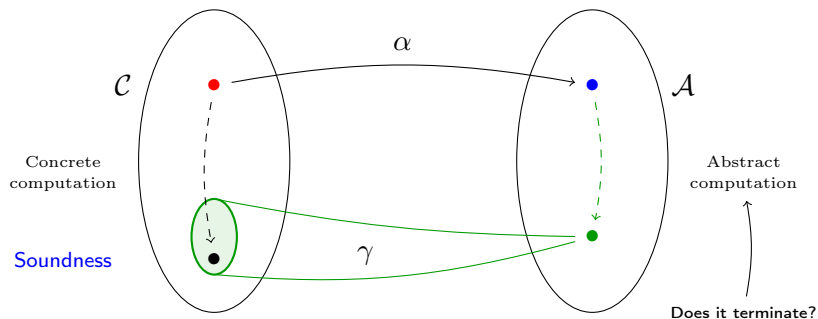
# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



# Abstract interpretation

Given a program semantics, abstracts its behaviour and provide a *sound* property of the program, also called *invariant*



# Analyzer termination is not guaranteed

Consider the siml-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0]$$

# Analyzer termination is not guaranteed

Consider the siml-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\}$$

# Analyzer termination is not guaranteed

Consider the siml-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow^* \{[x \mapsto n] \mid 0 \leq n \leq k, k \in \mathbb{N}\}$$

# Analyzer termination is not guaranteed

Consider the siml-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow^* \{[x \mapsto n] \mid 0 \leq n \leq k, k \in \mathbb{N}\} \rightarrow \dots$$

Interval analysis would also diverge

$$[x \mapsto [0, 0]]$$

# Analyzer termination is not guaranteed

Consider the simil-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow^* \{[x \mapsto n] \mid 0 \leq n \leq k, k \in \mathbb{N}\} \rightarrow \dots$$

Interval analysis would also diverge

$$[x \mapsto [0, 0]] \rightarrow [x \mapsto [0, 1]]$$

# Analyzer termination is not guaranteed

Consider the simil-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow^* \{[x \mapsto n] \mid 0 \leq n \leq k, k \in \mathbb{N}\} \rightarrow \dots$$

Interval analysis would also diverge

$$[x \mapsto [0, 0]] \rightarrow [x \mapsto [0, 1]] \rightarrow^* [x \mapsto [0, k]] \text{ with } k \in \mathbb{N}$$



# Analyzer termination is not guaranteed

Consider the simil-C program

```
int x = 0;
while(true) {
    x++;
}
```

A concrete semantics that collects variables values diverges

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow^* \{[x \mapsto n] \mid 0 \leq n \leq k, k \in \mathbb{N}\} \rightarrow \dots$$

Interval analysis would also diverge

$$[x \mapsto [0, 0]] \rightarrow [x \mapsto [0, 1]] \rightarrow^* [x \mapsto [0, k]] \text{ with } k \in \mathbb{N} \rightarrow \dots$$

# Goal

- Establish if some abstract semantics are computable.
- Focus on non-relational domains:
  - Interval domain:  $\dot{\mathbb{I}} \triangleq (Var \mapsto \mathbb{I})$
  - Non-relational collecting domain:  $\mathbb{C} \triangleq (Var \mapsto \wp(\mathbb{Z}))$

# Goal

- Establish if some abstract semantics are computable.
- Focus on non-relational domains:
  - Interval domain:  $\dot{\mathbb{I}} \triangleq (Var \mapsto \mathbb{I})$  **Computable**
  - Non-relational collecting domain:  $\mathbb{C} \triangleq (Var \mapsto \wp(\mathbb{Z}))$  **Partial results**

# Outline

- 1 Imp language and its semantics
- 2 Non relational abstract domains
- 3 Computing the abstract semantics
- 4 Results and future work

# Grammar

- Minimal core of an imperative language;
- Turing complete;
- Based on Kleene algebras with tests.

$$\begin{aligned}\text{Exp} \ni e &::= x \in I \mid x := k \mid x := y + k \\ \text{Imp} \ni C &::= e \mid C + C \mid C; C \mid C^* \mid \text{fix}(C)\end{aligned}$$

- $\text{while } b \text{ do } C \implies \text{fix}(b; C); \neg b$
- $\text{if } b \text{ then } C_1 \text{ else } C_2 \implies (b; C_1) + (\neg b; C_2)$

# Concrete semantics

$$\begin{aligned}
 \langle e \rangle X &\triangleq \{ \langle e \rangle \rho \mid \rho \in X, \langle e \rangle \rho \neq \perp \} \\
 \langle C_1 + C_2 \rangle X &\triangleq \langle C_1 \rangle X \cup \langle C_2 \rangle X \\
 \langle C_1; C_2 \rangle X &\triangleq \langle C_2 \rangle (\langle C_1 \rangle X) \\
 \langle C^* \rangle X &\triangleq \bigcup_{i \in \mathbb{N}} \langle C \rangle^i X \\
 \langle \text{fix}(C) \rangle X &\triangleq \text{lfp}(\lambda Y \in \wp(\text{Env}). (X \cup \langle C \rangle Y))
 \end{aligned}$$

- Collecting semantics.
- Finiteness and termination are undecidable because of Rice.

# Interval domain

$$\mathbb{I} \triangleq \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\} \wedge a \leq b\}$$

- Variables map to an interval  $[x \mapsto [-1, 1], y \mapsto [0, 0], \dots]$
- **Non-relational**, relations between variables (e.g.,  $x = 3 * y$ ) are not modelled
- Computation of fixpoints non trivial

# Infinite chains

```
x := 0; fix(true; x++)
```

- Computation does not halt

$$[x \mapsto 0] \rightarrow \{[x \mapsto 0], [x \mapsto 1]\} \rightarrow \cdots \rightarrow \{[x \mapsto n] \mid n \in \mathbb{N}\}$$

- Analysis does not halt either

$$[x \mapsto [0, 0]] \rightarrow [x \mapsto [0, 1]] \rightarrow \cdots \rightarrow [x \mapsto [0, \infty]]$$

- Problem: iterating over an infinite chain in the domain

$$[0, 0] \sqsubseteq [0, 1] \sqsubseteq \cdots \sqsubseteq [0, \infty]$$



# Widening and narrowing

- Common approach: widening  $\nabla$
- Widening over-approximates a result. Example

```
x := 0; fix(x < 10; x++);
```

- Precise analysis (not guaranteed to halt):  $[x \mapsto [0, 10]]$
- Analysis with widening (halts):  $[x \mapsto [0, \infty]]$

# The problem

## Problem

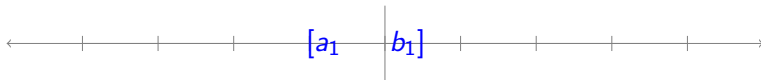
*Can we compute the precise interval semantics while ensuring the termination of the analyzer?*

# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$

$x$

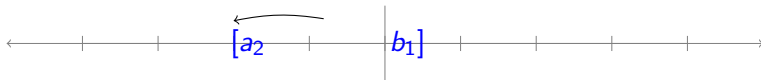


# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$

$x$

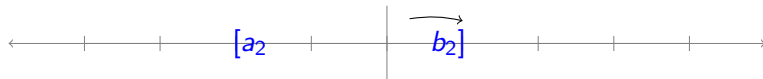


# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$

$x$



# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$

$x$



# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$



# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$





# Bounding the interval domain

Consider the behaviour of some variable  $x$  while computing

$$\llbracket \text{fix}(C) \rrbracket \eta = \text{lfp}(\lambda \mu. (\eta \sqcup \llbracket C \rrbracket \mu))$$



- Bounds are determined by the program  $C$  and the initial environment
- If a variable exceeds a bound the corresponding side of the interval is pushed to infinity

# Bounding the interval domain

By choosing  $\ell, u$  appropriately

$$\begin{aligned} \mathbb{I}_{\ell}^u &\triangleq \{[a, b] \mid a, b \in \mathbb{Z} \wedge \ell \leq a \leq b \leq u\} \\ &\cup \{[a, +\infty] \mid a \geq \ell\} \\ &\cup \{[-\infty, b] \mid b \leq u\} \end{aligned}$$

it holds that

$$\llbracket C \rrbracket_{\eta} = \llbracket C \rrbracket_{\ell}^u \eta$$

Since  $\mathbb{I}_{\ell}^u$  does not contain infinite chains, the termination trivializes.

# Non-relational collecting domain

$$\mathbb{C} \triangleq (Var \rightarrow \wp(\mathbb{Z})) \cup \{\perp\}$$

- Variables map to a generic subset of integers;
- Variable images are no longer convex;
- We could only prove some partial results.

# Bounding the non-relational collecting domain

$$\wp(\mathbb{Z})_\ell^u \triangleq \{S \subseteq \mathbb{Z} \mid S \neq \emptyset \wedge \forall x \in S \quad \ell \leq x \leq u\}$$

$$\overline{\mathbb{C}}_\ell^u \triangleq (\text{Var} \rightarrow \wp(\mathbb{Z})_\ell^u) \cup \{\perp, \top\}$$

- Variables mapped to bounded subsets of  $\mathbb{Z}$ .
- If some variable exceeds the bound then the whole analysis results in the smashed  $\top$  element.
- This way we can only infer analysis termination and not the most precise abstract invariant.

# Results

- Interval analysis can be computed precisely in finite time

$$\llbracket C \rrbracket^i_\eta = \llbracket C \rrbracket^{i^u}_\eta$$

- For non-relational collecting semantics we can decide termination of the analyzer.

# Future work

- Expand the language to include non-linear variable assignment  
 $x := y * k$
- Expand on precise non-relational collecting semantics