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.

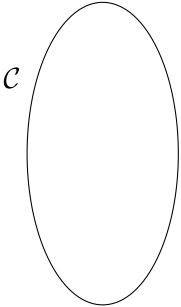
The cost of software failures

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.

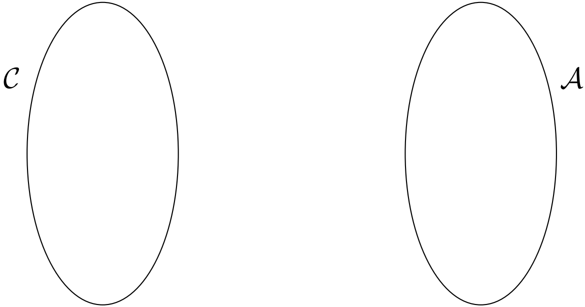
Abstract interpretation

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



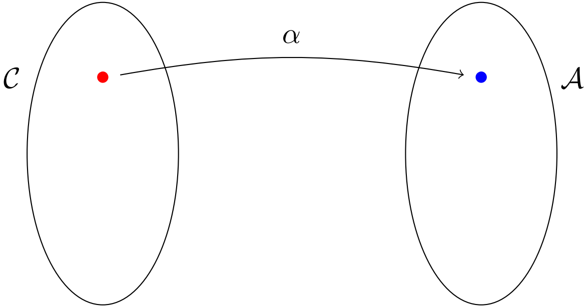
Abstract interpretation

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



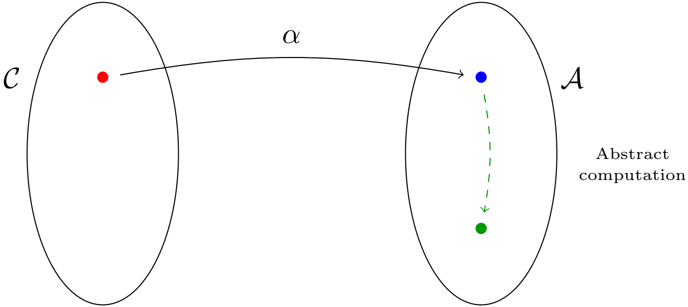
Abstract interpretation

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



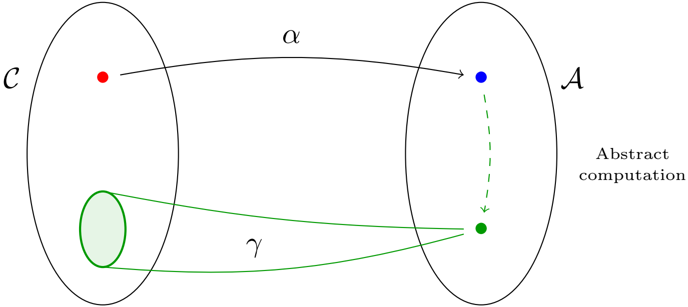
Abstract interpretation

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



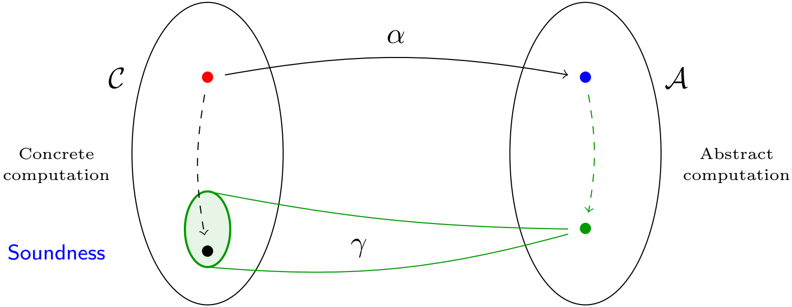
Abstract interpretation

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



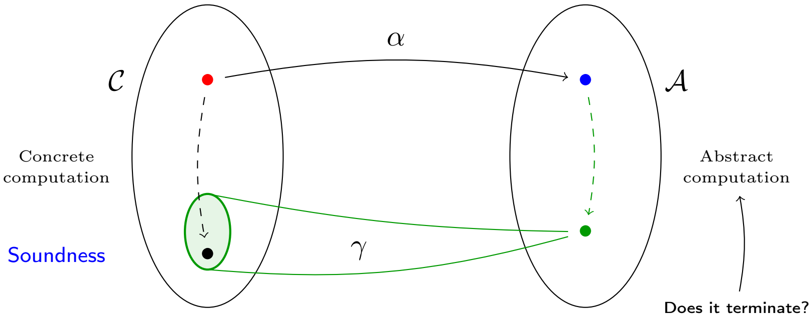
Abstract interpretation

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 simil-C program

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

A concrete semantics that collects variables values diverges

$$[\mathrm{x} \mapsto 0]$$

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

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

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

$$[\mathrm{x} \mapsto 0] \to \{[\mathrm{x} \mapsto 0], [\mathrm{x} \mapsto 1]\} \to^* \{[\mathrm{x} \mapsto n] \mid 0 \leqslant n \leqslant k, k \in \mathbb{N}\}$$

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

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

Interval analaysis would also diverge

$$[\mathrm{x} \mapsto [0, 0]]$$

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

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

Interval analaysis would also diverge

$$[\mathrm{x} \mapsto [0, 0]] \to [\mathrm{x} \mapsto [0, 1]]$$

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

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

Interval analaysis would also diverge

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

Analyzer termination is not guaranteed

Consider the simil-C program

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

A concrete semantics that collects variables values diverges

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

Interval analaysis would also diverge

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

Goal

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

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.

$$\mathrm{Exp} \ni \mathsf{e} ::= \mathtt{x} \in I \mid \mathtt{x} := k \mid \mathtt{x} := \mathtt{y} + k$$
$$\mathrm{Imp} \ni \mathsf{C} ::= \mathsf{e} \mid \mathsf{C} + \mathsf{C} \mid \mathsf{C}; \mathsf{C} \mid \mathsf{C}^* \mid \mathrm{fix}(\mathsf{C})$$

- while b do C $\implies$ $\mathrm{fix}(b; \mathsf{C}); \neg b$
- if b then C_1 else C_2 $\implies$ $(b; \mathsf{C}_1) + (\neg b; \mathsf{C}_2)$

Concrete semantics

$$
\begin{aligned}
\langle \mathsf{e} \rangle X &\triangleq \{ (\!|\mathsf{e}|\!)\rho \mid \rho \in X, (\!|\mathsf{e}|\!)\rho \neq \bot \} \\
\langle \mathsf{C}_1 + \mathsf{C}_2 \rangle X &\triangleq \langle \mathsf{C}_1 \rangle X \cup \langle \mathsf{C}_2 \rangle X \\
\langle \mathsf{C}_1 ; \mathsf{C}_2 \rangle X &\triangleq \langle \mathsf{C}_2 \rangle (\langle \mathsf{C}_1 \rangle X) \\
\langle \mathsf{C}^* \rangle X &\triangleq \bigcup_{i \in \mathbb{N}} \langle \mathsf{C} \rangle^i X \\
\langle \mathsf{fix}(\mathsf{C}) \rangle X &\triangleq \mathsf{lfp}(\lambda Y \in \wp(\mathsf{Env}).(X \cup \langle \mathsf{C} \rangle\, Y))
\end{aligned}
$$

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

Interval domain

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

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

Infinite chains

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

- Computation does not halt

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

- Analysis does not halt either

$$[\mathrm{x} \mapsto [0,0]] \to [\mathrm{x} \mapsto [0,1]] \to \cdots \to [\mathrm{x} \mapsto [0,\infty]]$$

- 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 ∇
- Widening over-approximates a result. Example

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

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

The problem

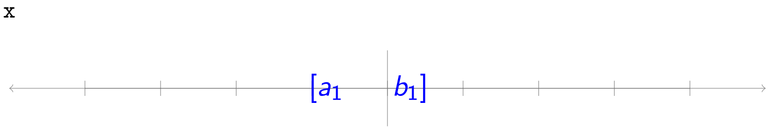
Problem

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

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$

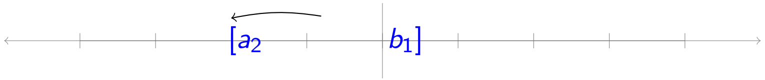


Bounding the interval domain

Consider the behaviour of some variable x while computing

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$

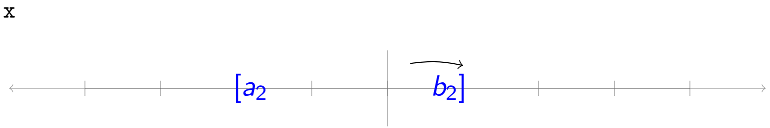
x



Bounding the interval domain

Consider the behaviour of some variable x while computing

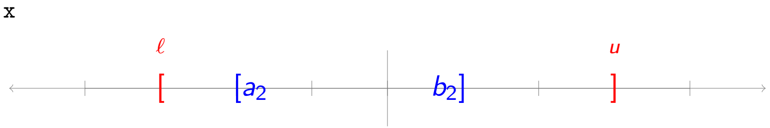
$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$



Bounding the interval domain

Consider the behaviour of some variable x while computing

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$

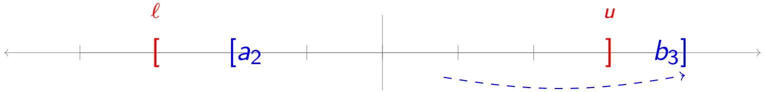


Bounding the interval domain

Consider the behaviour of some variable x while computing

$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$

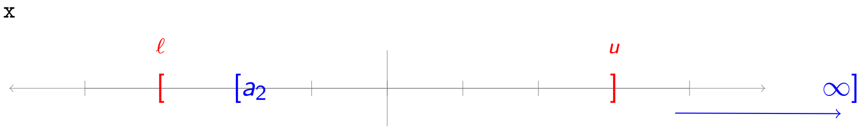
x



Bounding the interval domain

Consider the behaviour of some variable x while computing

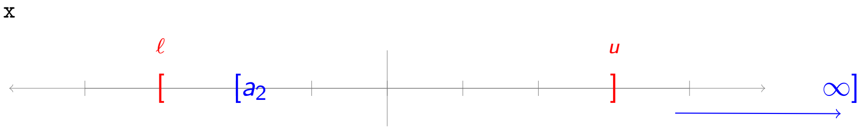
$$[\![\mathsf{fix}(\mathsf{C})]\!]\eta = \mathsf{lfp}(\lambda\mu.(\eta \sqcup [\![\mathsf{C}]\!]\mu))$$



Bounding the interval domain

Consider the behaviour of some variable x while computing

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



- 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 chosing ℓ, u appropriately

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

it holds that

$$
[\![\mathsf{C}]\!]\eta = [\![\mathsf{C}]\!]_\ell^u \eta
$$

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

Non-relational collecting domain

$$\mathbb{C} \triangleq (\mathit{Var} \to \wp(\mathbb{Z})) \cup \{\bot\}$$

- 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

$$\wp(\mathbb{Z})_\ell^u \triangleq \{S \subseteq \mathbb{Z} \mid S \neq \varnothing \land \forall x \in S \quad \ell \leqslant x \leqslant u\}$$
$$\overline{\mathbb{C}}_\ell^u \triangleq (Var \to \wp(\mathbb{Z})_\ell^u) \cup \{\bot, \top\}$$

- Variables mapped to bounded subsets of $\mathbb{Z}$.
- If some variable exceeds the bound than 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

$$[\![\mathsf{C}]\!]^{\dot{\mathbb{I}}}\eta = [\![\mathsf{C}]\!]^{\dot{\mathbb{I}}_\ell^u}\eta$$

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

Future work

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