Introduction to Abstract Interpretation

Session 3—Abstract Interpretation Workshop

Pierre Talbot

`pierre.talbot@uni.lu`

19th June 2024

University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

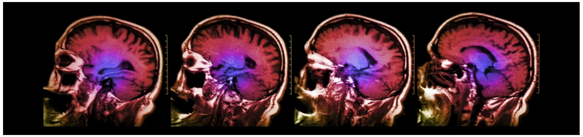
Costly Software Accidents

In 1996, the explosion of Ariane 501, which took ten years and $7 billion to build.

Costly Software Accidents

Bug in fMRI software calls 15 years of research into question

Popular pieces of software for fMRI were found to have false positive rates up to 70%

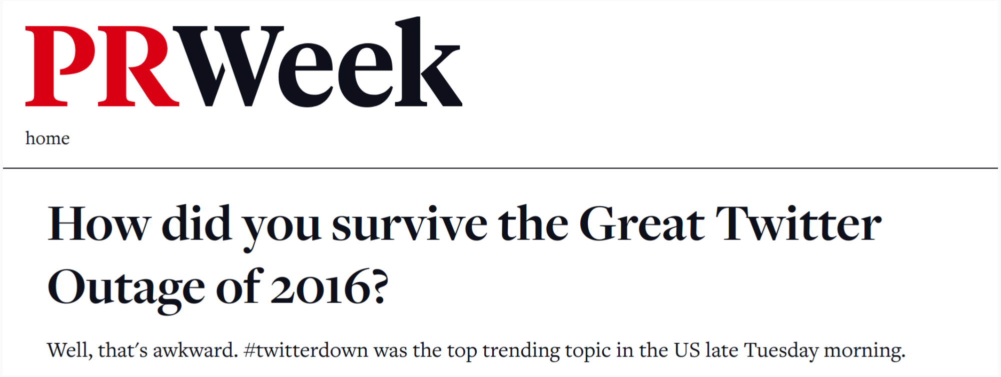


TRENDING NOW

PubMed.gov
The resolution solution: longitudinal examination of New Year's change attempts

Three of the most popular pieces of software for fMRI – SPM, FSL and AFNI – were all found to have false positive rates of up to 70 per cent. These findings could invalidate "up to 40,000 papers", researchers claim.

Costly Software Accidents



PRWeek

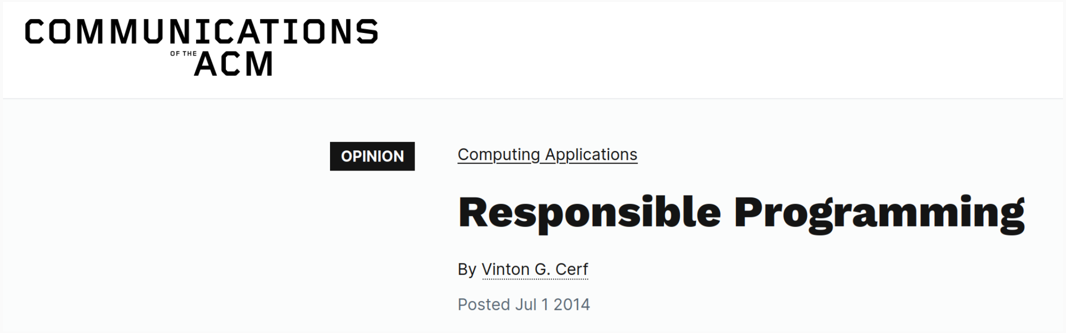
home

How did you survive the Great Twitter Outage of 2016?

Well, that's awkward. #twitterdown was the top trending topic in the US late Tuesday morning.

What Can We Do?

Nothing but it would be irresponsable.



COMMUNICATIONS OF THE ACM

OPINION

Computing Applications

Responsible Programming

By Vinton G. Cerf

Posted Jul 1 2014

"People who write software should have a clear sense of responsibility for its reliable operation and resistance to compromise and error."[1]

[1] https://cacm.acm.org/opinion/responsible-programming/

Know Your Limits...

OK, we should verify software but we should also know our limits...

Undecidability

By Rice's theorem, a static analyzer cannot have all the following properties:

- **General**: works on Turing-complete program.
- **Automated**: does not require human intervention.
- **Sound**: find all bugs.
- **Complete**: all bugs reported are true bugs.

Testing

General, semi-automated, complete but unsound (e.g., unit testing).

JUnit 5

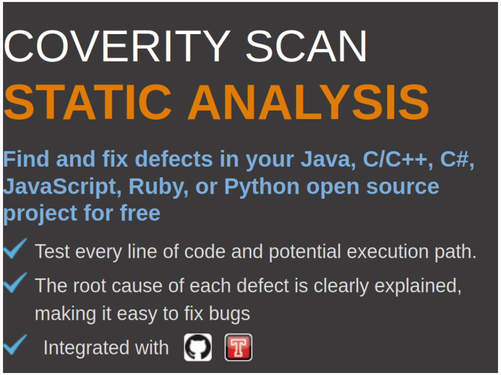
JUnit 4

The 5th major version of the programmer-friendly testing framework for Java and the JVM

"Program testing can be used to show the presence of bugs, but never to show their absence!" (Edsger Dijkstra).

Bug Finding

General, automated, **incomplete and unsound** (e.g. Coverity, CodeSonar).



COVERITY SCAN

STATIC ANALYSIS

Find and fix defects in your Java, C/C++, C#, JavaScript, Ruby, or Python open source project for free

✓ Test every line of code and potential execution path.

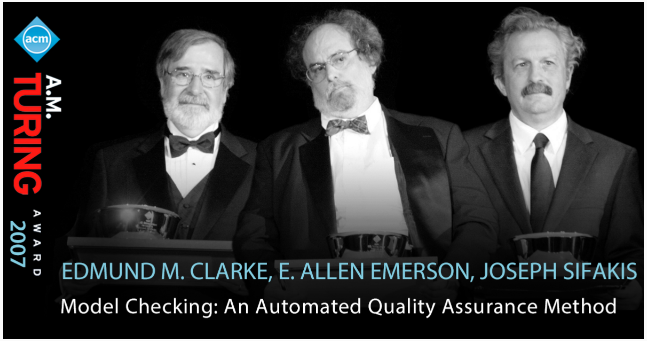
✓ The root cause of each defect is clearly explained, making it easy to fix bugs

✓ Integrated with

Additionally, Synopsys's implementation of static analysis can follow all the possible paths of execution through source code (including interprocedurally) and find defects and vulnerabilities caused by the conjunction of statements that are not errors independent of each other.

Model-Checking

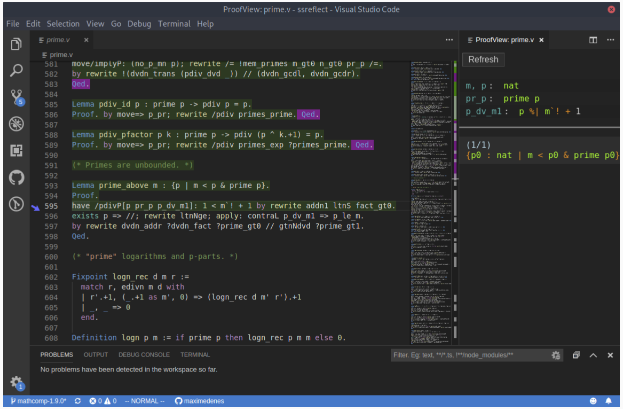
Non-general (finite state model), semi-automated, complete and sound.


acm
A.M. TURING AWARD 2007
EDMUND M. CLARKE, E. ALLEN EMERSON, JOSEPH SIFAKIS
Model Checking: An Automated Quality Assurance Method

Theorem Proving

General, non-automated, complete and sound (e.g., Lean, Coq).
But require human intervention to provide invariants (time consuming and require expertise).

Success story: Compcert, certified C compiler.


ProofView: prime.v - ssreflect - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
prime.v
ProofView: prime.v
Refresh
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Filter. Eg: text, **/*.ts, !**/node_modules/**
mathcomp-1.9.0* -- NORMAL -- maximedenes

```
move/implyP: (no_p_mn p); rewrite /= !mem_primes m_gt0 n_gt0 pr_p /=.
by rewrite !(dvdn_trans (pdiv_dvd _)) // (dvdn_gcdl, dvdn_gcdr).
Qed.

Lemma pdiv_id p : prime p -> pdiv p = p.
Proof. by move=> p_pr; rewrite /pdiv primes_prime. Qed.

Lemma pdiv_pfactor p k : prime p -> pdiv (p ^ k.+1) = p.
Proof. by move=> p_pr; rewrite /pdiv primes_exp ?primes_prime. Qed.

(* Primes are unbounded. *)

Lemma prime_above m : {p | m < p & prime p}.
Proof.
have /pdivP[p pr_p p_dv_m1]: 1 < m`! + 1 by rewrite addn1 ltnS fact_gt0.
exists p => //; rewrite ltnNge; apply: contraL p_dv_m1 => p_le_m.
by rewrite dvdn_addr ?dvdn_fact ?prime_gt0 // gtnNdvd ?prime_gt1.
Qed.

(* "prime" logarithms and p-parts. *)

Fixpoint logn_rec d m r :=
  match r, edivn m d with
  | r'.+1, (_.+1 as m', 0) => (logn_rec d m' r').+1
  | _, _ => 0
  end.

Definition logn p m := if prime p then logn_rec p m m else 0.
```

```
m, p :  nat
pr_p :  prime p
p_dv_m1 :  p %| m`! + 1

(1/1)
{p0 : nat | m < p0 & prime p0}
```

No problems have been detected in the workspace so far.

Abstract Interpretation

General, automated, incomplete and sound.

Success story: Astrée, prove absence of bugs in synchronous control/command aerospace software (Airbus).

Invented by Patrick and Radhia Cousot in the seventies.[2]

Foundations and Trends® in Programming Languages
Vol. 2, No. 2-3 (2015) 71–190
© 2015 J. Bertrane *et al.*
DOI: 10.1561/2500000002

now the essence of knowledge

Static Analysis and Verification of Aerospace Software by Abstract Interpretation

Julien Bertrane
Département d'informatique, École normale supérieure

Patrick Cousot
Département d'informatique, École normale supérieure & Courant Institute of Mathematical Sciences, New York University

Radhia Cousot
CNRS & Département d'informatique, École normale supérieure

Jérôme Feret
INRIA & Département d'informatique, École normale supérieure

Laurent Mauborgne
AbsInt Angewandte Informatik

Antoine Miné
Sorbonne University, University Pierre and Marie Curie, CNRS, LIP6

Xavier Rival
INRIA & Département d'informatique, École normale supérieure

[2] Patrick Cousot and Radhia Cousot. "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints". In: POPL 77'.

Simple Example: Pop Front

```
int pop_front(int* a, size_t& n) {
  int front = a[0];
  for(int i = 0; i < n; ++i) {
    a[i - 1] = a[i];
  }
  n--;
  return front;
}
```

This program has (at least) three bugs.

Simple Example: Pop Front

```
int pop_front(int* a, size_t& n) {
  int front = a[0];
  for(int i = 0; i < n; ++i) {
    a[i - 1] = a[i];
  }
  n--;
  return front;
}
```

This program has (at least) three bugs.

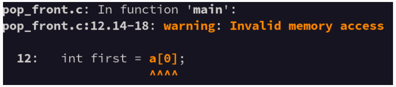
- *Invalid memory access*: `a[0]` when $n = 0$.
- *Invalid memory access*: `a[i - 1]` when $i = 0$.
- *Overflow*: `++i` can overflow since we can have $n >$ `INT_MAX`.

Simple Example: Pop Front

```
int pop_front(int* a, size_t& n) {
  int front = a[0];
  for(int i = 0; i < n; ++i) {
    a[i - 1] = a[i];
  }
  n--;
  return front;
}
```

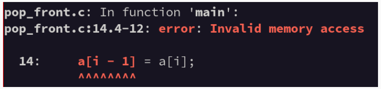
Let's run MOPSA, a static analyzer, on this program:

`mopsa-c pop_front.c`



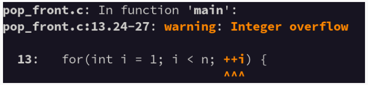
```
pop_front.c: In function 'main':
pop_front.c:12.14-18: warning: Invalid memory access

 12:    int first = a[0];
                    ^^^^
```



```
pop_front.c: In function 'main':
pop_front.c:14.4-12: error: Invalid memory access

 14:      a[i - 1] = a[i];
          ^^^^^^^^
```



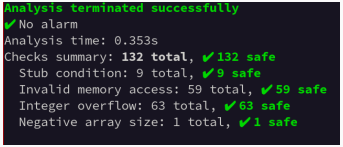
```
pop_front.c: In function 'main':
pop_front.c:13.24-27: warning: Integer overflow

 13:    for(int i = 1; i < n; ++i) {
                              ^^^
```

Simple Example: Push Front

Corrected version:

```
int pop_front(int* a, size_t& n) {
  if(n == 0) return -1;
  int front = a[0];
  for(size_t i = 1; i < n; ++i) {
    a[i - 1] = a[i];
  }
  n--;
  return front;
}
```



```
Analysis terminated successfully
✔ No alarm
Analysis time: 0.353s
Checks summary: 132 total, ✔ 132 safe
  Stub condition: 9 total, ✔ 9 safe
  Invalid memory access: 59 total, ✔ 59 safe
  Integer overflow: 63 total, ✔ 63 safe
  Negative array size: 1 total, ✔ 1 safe
```

Abstract Interpretation

Abstract interpretation answers precisely elementary questions:

- What is a program?
- What is a property of a program?
- What is the verification problem?

We now formally introduce abstract interpretation:

- **Concrete semantics**: answer the questions above.
- **Abstract semantics**: design effective verification algorithm.

Concrete Semantics

Syntax

$$
\begin{array}{rlr}
\langle S \rangle ::= & X \leftarrow \mathit{Expr} & \textit{assignment} \\
\mid & \textbf{if } \mathit{Expr} \circ \mathit{Expr} \textbf{ then } S \textbf{ else } S \textbf{ fi} & \textit{conditional} \\
\mid & \textbf{while } \mathit{Expr} \circ \mathit{Expr} \textbf{ do } S \textbf{ done} & \textit{loop} \\
\mid & S \; ; \; S & \textit{sequence} \\
\langle \mathit{Expr} \rangle ::= & X & \textit{variable} \\
\mid & -\mathit{Expr} & \textit{negation} \\
\mid & \mathit{Expr} \diamond \mathit{Expr} & \textit{arithmetic operation} \\
\mid & c & \textit{constant } c \in \mathbb{Z}
\end{array}
$$

where $\circ \in \{=, \neq, \leq, <, >, \geq, \dots\}$ and $\diamond \in \{+, -, /, *, \%, \dots\}$.

What is a Program?

Let's define:

- X a countable set of variables.
- $Asn : X \to \mathbb{Z}$ the set of assignments (aka. valuation, environments).
- $\mathcal{L} = \{\ell_1, \dots, \ell_n\}$ the set of control points.

At each control point, we look for the set of all possible values of x:

	set of values of x
$^{\ell_1}$	$\mathbb{Z}_{\ell_1}$
$x \leftarrow 1;^{\ell_2}$	$\{1\}_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$\{1\}_{\ell_3}$
$\quad ^{\ell_4}$	$\{1\}_{\ell_4}$
$\quad x \leftarrow x + 2;^{\ell_5}$	$\{3\}_{\ell_5}$
done$^{\ell_6}$	

What is a Program?

Let's define:

- X a countable set of variables.
- $Asn : X \to \mathbb{Z}$ the set of assignments (aka. valuation, environments).
- $\mathcal{L} = \{\ell_1, \dots, \ell_n\}$ the set of control points.

At each control point, we look for the set of all possible values of x:

	set of values of x
$^{\ell_1}$	$\mathbb{Z}_{\ell_1}$
$x \leftarrow 1;\ ^{\ell_2}$	$\{1\}_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$\{1, 3\}_{\ell_3}$
$\quad ^{\ell_4}$	$\{1, 3\}_{\ell_4}$
$\quad x \leftarrow x + 2;\ ^{\ell_5}$	$\{3, 5\}_{\ell_5}$
done$^{\ell_6}$	

What is a Program?

Let's define:

- X a countable set of variables.
- $Asn : X \to \mathbb{Z}$ the set of assignments (aka. valuation, environments).
- $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ the set of control points.

At each control point, we look for the set of all possible values of x:

	set of values of x
$^{\ell_1}$	$\mathbb{Z}_{\ell_1}$
$x \leftarrow 1;\,^{\ell_2}$	$\{1\}_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$\{1,3,5,7,9,11\}_{\ell_3}$
$\quad ^{\ell_4}$	$\{1,3,5,7,9\}_{\ell_4}$
$\quad x \leftarrow x+2;\,^{\ell_5}$	$\{3,5,7,9,11\}_{\ell_5}$
done$^{\ell_6}$	$\{11\}_{\ell_6}$

Property of Programs

	set of values of x
$^{\ell_1}$	$\mathbb{Z}_{\ell_1}$
$x \leftarrow 1;^{\ell_2}$	$\{1\}_{\ell_2}$
while $^{\ell_3} x \leq 10$ **do**	$\{1,3,5,7,9,11\}_{\ell_3}$
$\quad ^{\ell_4}$	$\{1,3,5,7,9\}_{\ell_4}$
$\quad x \leftarrow x + 2;^{\ell_5}$	$\{3,5,7,9,11\}_{\ell_5}$
done$^{\ell_6}$	$\{11\}_{\ell_6}$

- The sets S_{ℓ_i} are called *invariants*.
- They are the strongest possible, there is no set S'_{ℓ_i} such that $S_{\ell_i} \subset S'_{\ell_i}$.
- A property has the same domain than an invariant, for instance: `assert(x >= 11)` after ℓ_6 is the property $\{11, 12, 13, 14, 15, \ldots\}$.
- Clearly this property is validated since $\{11\}_{\ell_6} \subseteq \{11, 12, 13, 14, 15, \ldots\}$ (the program is even more restrictive than the property checked).

How to automatically compute the sets S_{ℓ_i}?

Semantics of Atomic Commands

We define the semantics on expressions and commands $\langle Com \rangle ::= X \leftarrow Expr \mid Expr \circ Expr$.

- Semantics of expressions: $\mathbf{E}[\![.]\!] : Expr \times Asn \to \mathbb{Z}$.
- Semantics of commands: $\mathbf{C}[\![.]\!] : Com \times \mathcal{P}(Asn) \to \mathcal{P}(Asn)$.

Examples

Let $A = \{\{x \mapsto 1, y \mapsto 10\}, \{x \mapsto 2, y \mapsto 11\}\}$.

- Simple arithmetic: $\mathbf{E}[\![x * y]\!]\{x \mapsto 4, y \mapsto 2\} = 8$.
- Assignment: $\mathbf{C}[\![x \leftarrow 1]\!]A = \{\{x \mapsto 1, y \mapsto 10\}, \{x \mapsto 1, y \mapsto 11\}\}$.
- Filtering: $\mathbf{C}[\![x \neq 2]\!]A = \{\{x \mapsto 1, y \mapsto 10\}\}$.

Semantics of Atomic Commands

Semantics of Expressions

- $\mathbf{E}[\![x]\!]\rho \triangleq \rho(x)$
- $\mathbf{E}[\![-e]\!]\rho \triangleq -\mathbf{E}[\![e]\!]\rho$
- $\mathbf{E}[\![e_1 \diamond e_2]\!]\rho \triangleq \mathbf{E}[\![e_1]\!]\rho \diamond \mathbf{E}[\![e_2]\!]\rho \qquad (\diamond \in \{+, -, /, *, \%, \ldots\})$
- $\mathbf{E}[\![c]\!]\rho \triangleq c$

Semantics of Commands

- $\mathbf{C}[\![x \leftarrow e]\!]A \triangleq \{\rho[x \mapsto \mathbf{E}[\![e]\!]\rho] \mid \rho \in A\}$
- $\mathbf{C}[\![e_1 \circ e_2]\!]A \triangleq \{\rho \in A \mid \mathbf{E}[\![e_1]\!]\rho \circ \mathbf{E}[\![e_2]\!]\rho\} \qquad (\circ \in \{=, \neq, \leq, <, >, \geq, \ldots\})$

Semantics of Program

- At each location $\ell \in \mathcal{L}$, we compute its set of reachable environments $\mathcal{X}_\ell$.
- We create an equational system from the program such that its solution is $\{\mathcal{X}_{\ell_1}, \dots, \mathcal{X}_{\ell_n}\}$.

$$\mathbf{eq}(^{\ell_1}\ x \leftarrow e\ ^{\ell_2}) \triangleq \{\mathcal{X}_{\ell_2} = \mathbf{C}[\![x \leftarrow e]\!]\mathcal{X}_{\ell_1}\}$$

$$\mathbf{eq}(^{\ell_1}\ s_1;^{\ell_2}\ s_2\ ^{\ell_3}) \triangleq \mathbf{eq}(^{\ell_1}\ s_1\ ^{\ell_2}) \cup \mathbf{eq}(^{\ell_2}\ s_2\ ^{\ell_3})$$

$$\begin{aligned}&\mathbf{eq}(^{\ell_1}\ \mathbf{if}\ e_1 \circ e_2\ \mathbf{then}\ ^{\ell_2}\ s_1\ ^{\ell_3}\ \mathbf{fi}\ ^{\ell_4}) \triangleq \\ &\qquad \{\mathcal{X}_{\ell_2} = \mathbf{C}[\![e_1 \circ e_2]\!]\mathcal{X}_{\ell_1}\} \ \cup\ \mathbf{eq}(^{\ell_2}\ s_1\ ^{\ell_3})\ \cup\ \{\mathcal{X}_{\ell_4} = \mathcal{X}_{\ell_3} \cup \mathbf{C}[\![e_1 \not\circ e_2]\!]\mathcal{X}_{\ell_1}\}\end{aligned}$$

$$\begin{aligned}&\mathbf{eq}(^{\ell_1}\ \mathbf{while}\ ^{\ell_2}\ e_1 \circ e_2\ \mathbf{do}\ ^{\ell_3}\ s_1\ ^{\ell_4}\ \mathbf{done}\ ^{\ell_5}) \triangleq \\ &\qquad \{\mathcal{X}_{\ell_2} = \mathcal{X}_{\ell_1} \cup \mathcal{X}_{\ell_4},\ \mathcal{X}_{\ell_3} = \mathbf{C}[\![e_1 \circ e_2]\!]\mathcal{X}_{\ell_2}\} \\ &\quad \cup \mathbf{eq}(^{\ell_3}\ s_1\ ^{\ell_4}) \\ &\quad \cup \{\mathcal{X}_{\ell_5} = \mathbf{C}[\![e_1 \not\circ e_2]\!]\mathcal{X}_{\ell_2}\}\end{aligned}$$

Equational Semantics Illustrated

- At each location $\ell \in \mathcal{L}$, we compute its set of reachable environments $\mathcal{X}_\ell \subseteq \mathit{Asn}$.
- We create an equational system from the program such that its solution is $\{\mathcal{X}_{\ell_1}, \dots, \mathcal{X}_{\ell_n}\}$.

```
$^{\ell_1}$ $x \leftarrow 1;$ $^{\ell_2}$
while $^{\ell_3} x \leq 10$ do
    $^{\ell_4}$ $x \leftarrow x + 2$ $^{\ell_5}$
done$^{\ell_6}$
```

$$
\begin{aligned}
\mathcal{X}_{\ell_1} &= \mathit{Asn} \\
\mathcal{X}_{\ell_2} &= \mathbf{C}[\![x \leftarrow 1]\!]\mathcal{X}_{\ell_1} \\
\mathcal{X}_{\ell_3} &= \mathcal{X}_{\ell_2} \cup \mathcal{X}_{\ell_5} \\
\mathcal{X}_{\ell_4} &= \mathbf{C}[\![x \leq 10]\!]\mathcal{X}_{\ell_3} \\
\mathcal{X}_{\ell_5} &= \mathbf{C}[\![x \leftarrow x + 2]\!]\mathcal{X}_{\ell_4} \\
\mathcal{X}_{\ell_6} &= \mathbf{C}[\![x > 10]\!]\mathcal{X}_{\ell_3}
\end{aligned}
$$

Computing the Least Fixpoint

$$
\begin{aligned}
\mathcal{X}_{\ell_1}^{0} &= \{\} \\
\mathcal{X}_{\ell_2}^{0} &= \{\} \\
\mathcal{X}_{\ell_3}^{0} &= \{\} \\
\mathcal{X}_{\ell_4}^{0} &= \{\} \\
\mathcal{X}_{\ell_5}^{0} &= \{\} \\
\mathcal{X}_{\ell_6}^{0} &= \{\}
\end{aligned}
$$

Computing the Least Fixpoint

$$\begin{aligned}
\mathcal{X}^0_{\ell_1} &= \{\} & \mathcal{X}^1_{\ell_1} &= \mathit{Asn} \\
\mathcal{X}^0_{\ell_2} &= \{\} & \mathcal{X}^1_{\ell_2} &= \mathbf{C}[\![x \leftarrow 1]\!]\mathcal{X}^0_{\ell_1} \\
\mathcal{X}^0_{\ell_3} &= \{\} & \mathcal{X}^1_{\ell_3} &= \mathcal{X}^0_{\ell_2} \cup \mathcal{X}^0_{\ell_5} \\
\mathcal{X}^0_{\ell_4} &= \{\} & \mathcal{X}^1_{\ell_4} &= \mathbf{C}[\![x \leq 10]\!]\mathcal{X}^0_{\ell_3} \\
\mathcal{X}^0_{\ell_5} &= \{\} & \mathcal{X}^1_{\ell_5} &= \mathbf{C}[\![x \leftarrow x + 2]\!]\mathcal{X}^0_{\ell_4} \\
\mathcal{X}^0_{\ell_6} &= \{\} & \mathcal{X}^1_{\ell_6} &= \mathbf{C}[\![x > 10]\!]\mathcal{X}^0_{\ell_3}
\end{aligned}$$

Computing the Least Fixpoint

$$\begin{array}{ll}
\mathcal{X}^0_{\ell_1} = \{\} & \mathcal{X}^1_{\ell_1} = \mathit{Asn} \\
\mathcal{X}^0_{\ell_2} = \{\} & \mathcal{X}^1_{\ell_2} = \{\} \\
\mathcal{X}^0_{\ell_3} = \{\} & \mathcal{X}^1_{\ell_3} = \{\} \\
\mathcal{X}^0_{\ell_4} = \{\} & \mathcal{X}^1_{\ell_4} = \{\} \\
\mathcal{X}^0_{\ell_5} = \{\} & \mathcal{X}^1_{\ell_5} = \{\} \\
\mathcal{X}^0_{\ell_6} = \{\} & \mathcal{X}^1_{\ell_6} = \{\}
\end{array}$$

Computing the Least Fixpoint

$$
\begin{array}{lll}
\mathcal{X}_{\ell_1}^{0} = \{\} & \mathcal{X}_{\ell_1}^{1} = Asn & \mathcal{X}_{\ell_1}^{2} = Asn \\
\mathcal{X}_{\ell_2}^{0} = \{\} & \mathcal{X}_{\ell_2}^{1} = \{\} & \mathcal{X}_{\ell_2}^{2} = \mathbf{C}[\![x \leftarrow 1]\!]\mathcal{X}_{\ell_1}^{1} \\
\mathcal{X}_{\ell_3}^{0} = \{\} & \mathcal{X}_{\ell_3}^{1} = \{\} & \mathcal{X}_{\ell_3}^{2} = \mathcal{X}_{\ell_2}^{1} \cup \mathcal{X}_{\ell_5}^{1} \\
\mathcal{X}_{\ell_4}^{0} = \{\} & \mathcal{X}_{\ell_4}^{1} = \{\} & \mathcal{X}_{\ell_4}^{2} = \mathbf{C}[\![x \leq 10]\!]\mathcal{X}_{\ell_3}^{1} \\
\mathcal{X}_{\ell_5}^{0} = \{\} & \mathcal{X}_{\ell_5}^{1} = \{\} & \mathcal{X}_{\ell_5}^{2} = \mathbf{C}[\![x \leftarrow x + 2]\!]\mathcal{X}_{\ell_4}^{1} \\
\mathcal{X}_{\ell_6}^{0} = \{\} & \mathcal{X}_{\ell_6}^{1} = \{\} & \mathcal{X}_{\ell_6}^{2} = \mathbf{C}[\![x > 10]\!]\mathcal{X}_{\ell_3}^{1}
\end{array}
$$

Computing the Least Fixpoint

$$\begin{array}{lll}
\mathcal{X}^0_{\ell_1} = \{\} & \mathcal{X}^1_{\ell_1} = \mathit{Asn} & \mathcal{X}^2_{\ell_1} = \mathit{Asn} \\
\mathcal{X}^0_{\ell_2} = \{\} & \mathcal{X}^1_{\ell_2} = \{\} & \mathcal{X}^2_{\ell_2} = \{\rho \in \mathit{Asn} \mid \rho(x) = 1\} \\
\mathcal{X}^0_{\ell_3} = \{\} & \mathcal{X}^1_{\ell_3} = \{\} & \mathcal{X}^2_{\ell_3} = \{\} \\
\mathcal{X}^0_{\ell_4} = \{\} & \mathcal{X}^1_{\ell_4} = \{\} & \mathcal{X}^2_{\ell_4} = \{\} \\
\mathcal{X}^0_{\ell_5} = \{\} & \mathcal{X}^1_{\ell_5} = \{\} & \mathcal{X}^2_{\ell_5} = \{\} \\
\mathcal{X}^0_{\ell_6} = \{\} & \mathcal{X}^1_{\ell_6} = \{\} & \mathcal{X}^2_{\ell_6} = \{\}
\end{array}$$

Computing the Least Fixpoint

$$
\begin{array}{llll}
\mathcal{X}_{\ell_1}^0 = \{\} & \mathcal{X}_{\ell_1}^1 = \mathit{Asn} & \mathcal{X}_{\ell_1}^2 = \mathit{Asn} & \mathcal{X}_{\ell_1}^3 = \mathit{Asn} \\
\mathcal{X}_{\ell_2}^0 = \{\} & \mathcal{X}_{\ell_2}^1 = \{\} & \mathcal{X}_{\ell_2}^2 = \{\rho \in \mathit{Asn} \mid \rho(x) = 1\} & \mathcal{X}_{\ell_2}^3 = \mathbf{C}[\![x \leftarrow 1]\!]\mathcal{X}_{\ell_1}^2 \\
\mathcal{X}_{\ell_3}^0 = \{\} & \mathcal{X}_{\ell_3}^1 = \{\} & \mathcal{X}_{\ell_3}^2 = \{\} & \mathcal{X}_{\ell_3}^3 = \mathcal{X}_{\ell_2}^2 \cup \mathcal{X}_{\ell_5}^2 \\
\mathcal{X}_{\ell_4}^0 = \{\} & \mathcal{X}_{\ell_4}^1 = \{\} & \mathcal{X}_{\ell_4}^2 = \{\} & \mathcal{X}_{\ell_4}^3 = \mathbf{C}[\![x \leq 10]\!]\mathcal{X}_{\ell_3}^2 \\
\mathcal{X}_{\ell_5}^0 = \{\} & \mathcal{X}_{\ell_5}^1 = \{\} & \mathcal{X}_{\ell_5}^2 = \{\} & \mathcal{X}_{\ell_5}^3 = \mathbf{C}[\![x \leftarrow x + 2]\!]\mathcal{X}_{\ell_4}^2 \\
\mathcal{X}_{\ell_6}^0 = \{\} & \mathcal{X}_{\ell_6}^1 = \{\} & \mathcal{X}_{\ell_6}^2 = \{\} & \mathcal{X}_{\ell_6}^3 = \mathbf{C}[\![x > 10]\!]\mathcal{X}_{\ell_3}^2
\end{array}
$$

Computing the Least Fixpoint

$$
\begin{array}{llll}
\mathcal{X}^0_{\ell_1} = \{\} & \mathcal{X}^1_{\ell_1} = Asn & \mathcal{X}^2_{\ell_1} = Asn & \mathcal{X}^3_{\ell_1} = Asn \\
\mathcal{X}^0_{\ell_2} = \{\} & \mathcal{X}^1_{\ell_2} = \{\} & \mathcal{X}^2_{\ell_2} = \{\rho \in Asn \mid \rho(x) = 1\} & \mathcal{X}^3_{\ell_2} = \{\rho \in Asn \mid \rho(x) = 1\} \\
\mathcal{X}^0_{\ell_3} = \{\} & \mathcal{X}^1_{\ell_3} = \{\} & \mathcal{X}^2_{\ell_3} = \{\} & \mathcal{X}^3_{\ell_3} = \{\rho \in Asn \mid \rho(x) = 1\} \\
\mathcal{X}^0_{\ell_4} = \{\} & \mathcal{X}^1_{\ell_4} = \{\} & \mathcal{X}^2_{\ell_4} = \{\} & \mathcal{X}^3_{\ell_4} = \{\} \\
\mathcal{X}^0_{\ell_5} = \{\} & \mathcal{X}^1_{\ell_5} = \{\} & \mathcal{X}^2_{\ell_5} = \{\} & \mathcal{X}^3_{\ell_5} = \{\} \\
\mathcal{X}^0_{\ell_6} = \{\} & \mathcal{X}^1_{\ell_6} = \{\} & \mathcal{X}^2_{\ell_6} = \{\} & \mathcal{X}^3_{\ell_6} = \{\}
\end{array}
$$

Computing the Least Fixpoint

The least fixpoint is reached after 10 iterations.
This way of computing the fixpoint is called *Jacobi iterations*.

```
$^{\ell_1}$ $x \leftarrow 1;$ $^{\ell_2}$
**while** $^{\ell_3} x \leq 10$ **do**
    $^{\ell_4}$ $x \leftarrow x + 2$ $^{\ell_5}$
**done**$^{\ell_6}$
```

$$
\begin{aligned}
\mathcal{X}_{\ell_1}^{10} &= Asn \\
\mathcal{X}_{\ell_2}^{10} &= \{\rho \in Asn \mid \rho(x) = 1\} \\
\mathcal{X}_{\ell_3}^{10} &= \{\rho \in Asn \mid \rho(x) \in \{1, 3, \ldots, 11\}\} \\
\mathcal{X}_{\ell_4}^{10} &= \{\rho \in \mathcal{X}_{\ell_3}^{9} \mid \rho(x) \in \{1, 3, \ldots, 9\}\} \\
\mathcal{X}_{\ell_5}^{10} &= \{\rho \in \mathcal{X}_{\ell_4}^{9} \mid \rho(x) \in \{3, \ldots, 11\}\} \\
\mathcal{X}_{\ell_6}^{10} &= \{\rho \in \mathcal{X}_{\ell_3}^{9} \mid \rho(x) = 11\}
\end{aligned}
$$

System of Equations

We create a system of equations over the same domain $\mathcal{L} \rightarrow \mathcal{P}(X \rightarrow \mathbb{Z})$:

$$\begin{aligned} &\mathcal{X}_{\ell_i} = F_i(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_n}\}) \\ &1 \leq i \leq n \end{aligned}$$

where $F_i \in (\mathcal{L} \rightarrow \mathcal{P}(X \rightarrow \mathbb{Z})) \rightarrow (\mathcal{L} \rightarrow \mathcal{P}(X \rightarrow \mathbb{Z}))$ to obtain a system of equation of the form:

Example

From $\mathcal{X}_{\ell_2} = \mathbf{C}(i \leftarrow 1, \mathcal{X}_{\ell_1})$ to $\mathcal{X}_{\ell_2} = F_2(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_6}\})$ with F_2 defined as:

$$F_2(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_6}\}) \triangleq \{\mathcal{X}_{\ell_1}, \mathbf{C}(i \leftarrow 1, \mathcal{X}_{\ell_1}), \ldots, \mathcal{X}_{\ell_6}\}$$

System of Equations

We create a system of equations over the same domain $\mathcal{L} \to \mathcal{P}(X \to \mathbb{Z})$:

$$\mathcal{X}_{\ell_i} = F_i(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_n}\})$$
$$1 \leq i \leq n$$

where $F_i \in (\mathcal{L} \to \mathcal{P}(X \to \mathbb{Z})) \to (\mathcal{L} \to \mathcal{P}(X \to \mathbb{Z}))$ to obtain a system of equation of the form:

Example

From $\mathcal{X}_{\ell_2} = \mathbf{C}(i \leftarrow 1, \mathcal{X}_{\ell_1})$ to $\mathcal{X}_{\ell_2} = F_2(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_6}\})$ with F_2 defined as:

$$F_2(\{\mathcal{X}_{\ell_1}, \ldots, \mathcal{X}_{\ell_6}\}) \triangleq \{\mathcal{X}_{\ell_1}, \mathbf{C}(i \leftarrow 1, \mathcal{X}_{\ell_1}), \ldots, \mathcal{X}_{\ell_6}\}$$

Then, the fixpoint of $F_n \circ F_{n-1} \circ \ldots \circ F_1$ starting at $\{\{\}_{\ell_1}, \ldots, \{\}_{\ell_n}\}$ is

the unique least fixpoint.

(by Kleene theorem and continuity of all F_i).

Summary

Abstract interpretation answers precisely the questions we raised at the beginning:

- What is a program? The least fixpoint point of $\mathbf{eq}(S)$.
- What is a property? A subset of the environment $P \in \mathcal{P}(Asn)$.
  Example: $x < 12$ is the property $\{\rho \in Asn \mid \rho(x) \in \{1, 2, \ldots, 11\}\}$.
- What is the verification problem? An inclusion check: $(\mathbf{lfp}\ \mathbf{eq}(S))_{\ell_i} \subseteq P$.
  Example: $\mathcal{X}_{\ell_6} = \{\rho \in Asn \mid \rho(x) = 11\} \subseteq \{\rho \in Asn \mid \rho(x) \in \{1, 2, \ldots, 11\}\}$

Note: We have focussed on a particular semantics called *assertional forward reachability semantics*, but there exists other concrete semantics which are more or less precise (e.g. relational semantics, trace semantics).

Small Issues...

- **lfp eq**(S) might only exists after an infinite number of iterations.
- Even if finite, the sets $\mathcal{X}_{\ell_i}$ can grow exponentially, and the number of iterations can be very big.

Abstract Semantics

Ingredients of Abstract Interpretation

Let S be a program.
We want a mechanical procedure approximating $\textbf{lfp eq}(S)$.
The ingredients are:

1. An *abstract representation* $A^\sharp$ of $\mathcal{P}(X \to \mathbb{Z})$ such that the elements of $A^\sharp$ are finitely representable in a machine.
2. An *abstract set of equations* $\textbf{eq}^\sharp(S)$ such that $\textbf{lfp eq}^\sharp(S)$ is computable in a finite number of steps.
3. *Soundness*: $\textbf{lfp eq}(S) \subseteq \gamma(\textbf{lfp eq}^\sharp(S))$ where $\gamma : \mathcal{P}(X \to \mathbb{Z}) \to A^\sharp$.
   $\Rightarrow$ We overapproximate the least fixpoint, meaning that we find all bugs but potentially have false-positives due to the overapproximation.

Abstract Domain

Abstract Domain

$\Rightarrow$ The key of abstract interpretation is to work with *abstractions* of the concrete semantics.

Definition

An abstract domain is a lattice $\langle A^\sharp, \sqsubseteq, \sqcup, \sqcap, \bot, \top, \mathbf{C}^\sharp[\![.]\!]\rangle$ such that:

- Every element of $A^\sharp$ is representable in a machine.
- The operations on $A^\sharp$ are efficiently computable.
- $\mathbf{C}^\sharp[\![.]\!]$ is order-preserving.

The concrete and abstract semantics are connected by a Galois connection:

$$\langle \mathcal{P}(X \to \mathbb{Z}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A^\sharp, \sqsubseteq\rangle$$

Interval Lattice

Definition

The lattice of interval $\langle \mathcal{I}, \sqsubseteq, \sqcup, \sqcap, \bot, [-\infty, \infty] \rangle$ is defined as:

$$\mathcal{I} \triangleq \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{\infty\}, a \sqsubseteq b\} \cup \{\bot\}$$

with the following operations:

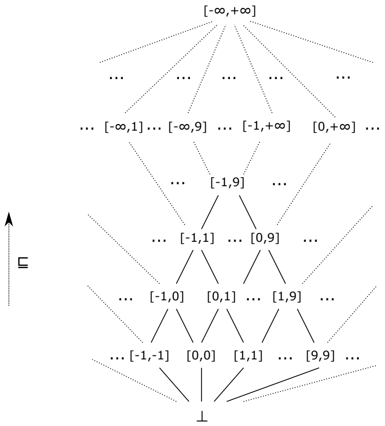
- $[a, b] \sqsubseteq [c, d] \Leftrightarrow a \geq c \wedge b \leq d$.
- $[a, b] \sqcup [c, d] \triangleq [\min(a, c), \max(b, d)]$.
- $[a, b] \sqcap [c, d] \triangleq [\max(a, c), \min(b, d)]$.

We also define projection functions $\lfloor [a, b] \rfloor \triangleq a$ and $\lceil [a, b] \rceil \triangleq b$.

The interval lattice

Introduced by [Cous76].

$$\mathcal{B}^\sharp \overset{\text{def}}{=} \{\, [a,b] \mid a \in \mathbb{I} \cup \{-\infty\},\ b \in \mathbb{I} \cup \{+\infty\},\ a \leq b \,\} \cup \{\, \perp_b^\sharp \,\}$$



Note: intervals are open at infinite bounds $+\infty$, $-\infty$.

Abstract Interval Static Analysis

At each control point, we look for the set of all possible values of x:

	abstract set of values of x
$^{\ell_1}$	$\top_{\ell_1}$
$x \leftarrow 1;^{\ell_2}$	$[1,1]_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$[1,1]_{\ell_3}$
$\quad ^{\ell_4}$	$[1,1]_{\ell_4}$
$\quad x \leftarrow x+2;^{\ell_5}$	$[3,3]_{\ell_5}$
done$^{\ell_6}$	

Abstract Interval Static Analysis

At each control point, we look for the set of all possible values of x:

	abstract set of values of x
$^{\ell_1}$	$\top_{\ell_1}$
$x \leftarrow 1;^{\ell_2}$	$[1,1]_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$[1,1] \sqcup [3,3] = [1,3]_{\ell_3}$
$\quad ^{\ell_4}$	$[1,3]_{\ell_4}$
$\quad x \leftarrow x + 2;^{\ell_5}$	$[3,5]_{\ell_5}$
done$^{\ell_6}$	

Abstract Interval Static Analysis

At each control point, we look for the set of all possible values of x:

	abstract set of values of x
ℓ_1	$\top_{\ell_1}$
$x \leftarrow 1;^{\ell_2}$	$[1,1]_{\ell_2}$
while $^{\ell_3}x \leq 10$ **do**	$[1,11]_{\ell_3}$
$\quad \ell_4$	$[1,9]_{\ell_4}$
$\quad x \leftarrow x+2;^{\ell_5}$	$[3,11]_{\ell_5}$
done$^{\ell_6}$	$[11,11]_{\ell_6}$

Loss of precision

Working in the abstract can result in weaker invariants:

- The first time we reach ℓ_5, we have $x \mapsto [1..3]$.
- But $2 \in [1..3]$ although it is not a possible value!
- This interval analysis would be unable to prove that $x \neq 2$ at location ℓ_5.

Interval Abstract Domain

The abstract domain of interval is
$\mathcal{I}^\sharp \triangleq \langle X \to \mathcal{I}, \dot\sqsubseteq, \dot\sqcup, \dot\sqcap, x \in X \mapsto \bot, x \in X \mapsto [-\infty, \infty], \mathbf{C}_I^\sharp[\![.]\!]\rangle$ where $\dot\sqsubseteq, \dot\sqcup, \dot\sqcap$ are pointwise interval operations.

The Galois connection with the concrete domain is given by:

- $\gamma_I(A) \triangleq x \in X \mapsto \bigsqcup_{\rho \in A}[\rho(x), \rho(x)]$.
- $\alpha_I(\sigma) \triangleq \{\rho \in Asn \mid \forall x \in X,\ \rho(x) \in \sigma(x)\}$.

Loss of precision

Let $A = \{\{x \mapsto 0, y \mapsto 1\}, \{x \mapsto 1, y \mapsto 0\}\}$, then:

$$\gamma_I(A) = \{x \mapsto [0,1], y \mapsto [0,1]\}$$

All relationships among variables are forgotten (this is called a *Cartesian abstraction*).

Interval Abstract Semantics

Abstract Semantics of Expressions

Let $\mathbf{E}_I^\sharp[\![.]\!] : \mathit{Expr} \times (X \to \mathcal{I}) \to \mathcal{I}$ and $\sigma \in X \to \mathcal{I}$.

- $\mathbf{E}_I^\sharp[\![x]\!]\sigma \triangleq \sigma(x)$
- $\mathbf{E}_I^\sharp[\![c]\!]\sigma \triangleq [c, c]$
- $\mathbf{E}_I^\sharp[\![-e]\!]\sigma \triangleq \mathit{let}\ [a, b] = \mathbf{E}_I^\sharp[\![e]\!]\sigma\ \mathit{in}\ [-b, -a]$
- $\mathbf{E}_I^\sharp[\![e_1 + e_2]\!]\sigma \triangleq \mathit{let}\ [a, b] = \mathbf{E}_I^\sharp[\![e_1]\!]\sigma\ \mathit{in}$
  $\mathit{let}\ [c, d] = \mathbf{E}_I^\sharp[\![e_2]\!]\sigma\ \mathit{in}\ [a + b, c + d]$

Abstract Semantics of Commands

Let $\mathbf{C}_I^\sharp[\![.]\!] : \mathit{Expr} \times (X \to \mathcal{I}) \to (X \to \mathcal{I})$ and $\sigma \in X \to \mathcal{I}$.

- $\mathbf{C}_I^\sharp[\![x \leftarrow e]\!]\sigma \triangleq \sigma[x \mapsto \mathbf{E}_I^\sharp[\![e]\!]\sigma]$
- $\mathbf{C}_I^\sharp[\![x \leq y]\!]\sigma \triangleq \sigma[x \mapsto \sigma(x) \sqcap [-\infty, \lceil\sigma(y)\rceil]]$
  $\dot{\sqcap}\ \sigma[y \mapsto \sigma(y) \sqcap [\lfloor\sigma(x)\rfloor, \infty]]$

Abstract Equational Semantics

Abstract Semantics of Program

- At each location $\ell \in \mathcal{L}$, we compute its set of reachable environments $\mathcal{X}^\sharp_\ell \in A^\sharp$.
- We create an equational system from the program such that its solution is $\{\mathcal{X}^\sharp_{\ell_1}, \ldots, \mathcal{X}^\sharp_{\ell_n}\}$.

$$\mathbf{eq}^\sharp({}^{\ell_1}\ x \leftarrow e\ {}^{\ell_2}) \triangleq \{\mathcal{X}^\sharp_{\ell_2} = \mathbf{C}^\sharp[\![x \leftarrow e]\!]\mathcal{X}^\sharp_{\ell_1}\}$$

$$\mathbf{eq}^\sharp({}^{\ell_1}\ s_1;{}^{\ell_2}\ s_2\ {}^{\ell_3}) \triangleq \mathbf{eq}^\sharp({}^{\ell_1}\ s_1\ {}^{\ell_2}) \cup \mathbf{eq}^\sharp({}^{\ell_2}\ s_2\ {}^{\ell_3})$$

$$\begin{aligned}&\mathbf{eq}^\sharp({}^{\ell_1}\ \mathbf{if}\ e_1 \circ e_2\ \mathbf{then}\ {}^{\ell_2}\ s_1\ {}^{\ell_3}\ \mathbf{fi}\ {}^{\ell_4}) \triangleq \\ &\quad \{\mathcal{X}^\sharp_{\ell_2} = \mathbf{C}^\sharp[\![e_1 \circ e_2]\!]\mathcal{X}^\sharp_{\ell_1}\} \cup \mathbf{eq}^\sharp({}^{\ell_2}\ s_1\ {}^{\ell_3}) \cup \{\mathcal{X}^\sharp_{\ell_4} = \mathcal{X}^\sharp_{\ell_3} \sqcup \mathbf{C}^\sharp[\![e_1 \not\circ e_2]\!]\mathcal{X}^\sharp_{\ell_1}\}\end{aligned}$$

$$\begin{aligned}&\mathbf{eq}^\sharp({}^{\ell_1}\ \mathbf{while}\ {}^{\ell_2}\ e_1 \circ e_2\ \mathbf{do}\ {}^{\ell_3}\ s_1\ {}^{\ell_4}\ \mathbf{done}\ {}^{\ell_5}) \triangleq \\ &\quad \{\mathcal{X}^\sharp_{\ell_2} = \mathcal{X}^\sharp_{\ell_1} \sqcup \mathcal{X}^\sharp_{\ell_4},\ \mathcal{X}^\sharp_{\ell_3} = \mathbf{C}^\sharp[\![e_1 \circ e_2]\!]\mathcal{X}^\sharp_{\ell_2}\} \\ &\quad \cup\ \mathbf{eq}^\sharp({}^{\ell_3}\ s_1\ {}^{\ell_4}) \\ &\quad \cup\ \{\mathcal{X}^\sharp_{\ell_5} = \mathbf{C}^\sharp[\![e_1 \not\circ e_2]\!]\mathcal{X}^\sharp_{\ell_2}\}\end{aligned}$$

Abstract Fixpoint

Instead of working on the set of concrete values, we work on intervals.

```
$^{\ell_1}$ $i \leftarrow 1;$ $^{\ell_2}$
while $^{\ell_3} i \leq 10$ do
    $^{\ell_4}$ $i \leftarrow i + 2$ $^{\ell_5}$
done$^{\ell_6}$
```

$$\begin{aligned}
\mathcal{X}^{\sharp}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp}_{\ell_2} &= \mathbf{C}^{\sharp}_{I}[\![i \leftarrow 1]\!] \mathcal{X}^{\sharp}_{\ell_1} \\
\mathcal{X}^{\sharp}_{\ell_3} &= \mathcal{X}^{\sharp}_{\ell_2} \cup \mathcal{X}^{\sharp}_{\ell_5} \\
\mathcal{X}^{\sharp}_{\ell_4} &= \mathbf{C}^{\sharp}_{I}[\![i \leq 10]\!] \mathcal{X}^{\sharp}_{\ell_3} \\
\mathcal{X}^{\sharp}_{\ell_5} &= \mathbf{C}^{\sharp}_{I}[\![i \leftarrow i + 2]\!] \mathcal{X}^{\sharp}_{\ell_4} \\
\mathcal{X}^{\sharp}_{\ell_6} &= \mathbf{C}^{\sharp}_{I}[\![i > 10]\!] \mathcal{X}^{\sharp}_{\ell_3}
\end{aligned}$$

Computing the Abstract Least Fixpoint

$$
\begin{aligned}
\mathcal{X}_{\ell_1}^{\sharp 0} &= \bot \\
\mathcal{X}_{\ell_2}^{\sharp 0} &= \bot \\
\mathcal{X}_{\ell_3}^{\sharp 0} &= \bot \\
\mathcal{X}_{\ell_4}^{\sharp 0} &= \bot \\
\mathcal{X}_{\ell_5}^{\sharp 0} &= \bot \\
\mathcal{X}_{\ell_6}^{\sharp 0} &= \bot
\end{aligned}
$$

Computing the Abstract Least Fixpoint

$$
\begin{aligned}
\mathcal{X}^{\sharp 0}_{\ell_1} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp 0}_{\ell_2} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_2} &= \mathbf{C}^{\sharp}_{I}[\![x \leftarrow 1]\!]\mathcal{X}^{\sharp 0}_{\ell_1} \\
\mathcal{X}^{\sharp 0}_{\ell_3} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_3} &= \mathcal{X}^{\sharp 0}_{\ell_2} \cup \mathcal{X}^{\sharp 0}_{\ell_5} \\
\mathcal{X}^{\sharp 0}_{\ell_4} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_4} &= \mathbf{C}^{\sharp}_{I}[\![x \leq 10]\!]\mathcal{X}^{\sharp 0}_{\ell_3} \\
\mathcal{X}^{\sharp 0}_{\ell_5} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_5} &= \mathbf{C}^{\sharp}_{I}[\![x \leftarrow x + 2]\!]\mathcal{X}^{\sharp 0}_{\ell_4} \\
\mathcal{X}^{\sharp 0}_{\ell_6} &= \bot & \qquad \mathcal{X}^{\sharp 1}_{\ell_6} &= \mathbf{C}^{\sharp}_{I}[\![x > 10]\!]\mathcal{X}^{\sharp 0}_{\ell_3}
\end{aligned}
$$

Computing the Abstract Least Fixpoint

$$\begin{array}{ll}
\mathcal{X}^{\sharp 0}_{\ell_1} = \bot & \mathcal{X}^{\sharp 1}_{\ell_1} = \top \\
\mathcal{X}^{\sharp 0}_{\ell_2} = \bot & \mathcal{X}^{\sharp 1}_{\ell_2} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_3} = \bot & \mathcal{X}^{\sharp 1}_{\ell_3} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_4} = \bot & \mathcal{X}^{\sharp 1}_{\ell_4} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_5} = \bot & \mathcal{X}^{\sharp 1}_{\ell_5} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_6} = \bot & \mathcal{X}^{\sharp 1}_{\ell_6} = \bot
\end{array}$$

Computing the Abstract Least Fixpoint

$$\begin{array}{lll}
\mathcal{X}^{\sharp 0}_{\ell_1} = \bot & \mathcal{X}^{\sharp 1}_{\ell_1} = \top & \mathcal{X}^{\sharp 2}_{\ell_1} = \top \\
\mathcal{X}^{\sharp 0}_{\ell_2} = \bot & \mathcal{X}^{\sharp 1}_{\ell_2} = \bot & \mathcal{X}^{\sharp 2}_{\ell_2} = \mathbf{C}^\sharp_I[\![x \leftarrow 1]\!]\mathcal{X}^{\sharp 1}_{\ell_1} \\
\mathcal{X}^{\sharp 0}_{\ell_3} = \bot & \mathcal{X}^{\sharp 1}_{\ell_3} = \bot & \mathcal{X}^{\sharp 2}_{\ell_3} = \mathcal{X}^{\sharp 1}_{\ell_2} \cup \mathcal{X}^{\sharp 1}_{\ell_5} \\
\mathcal{X}^{\sharp 0}_{\ell_4} = \bot & \mathcal{X}^{\sharp 1}_{\ell_4} = \bot & \mathcal{X}^{\sharp 2}_{\ell_4} = \mathbf{C}^\sharp_I[\![x \leq 10]\!]\mathcal{X}^{\sharp 1}_{\ell_3} \\
\mathcal{X}^{\sharp 0}_{\ell_5} = \bot & \mathcal{X}^{\sharp 1}_{\ell_5} = \bot & \mathcal{X}^{\sharp 2}_{\ell_5} = \mathbf{C}^\sharp_I[\![x \leftarrow x + 2]\!]\mathcal{X}^{\sharp 1}_{\ell_4} \\
\mathcal{X}^{\sharp 0}_{\ell_6} = \bot & \mathcal{X}^{\sharp 1}_{\ell_6} = \bot & \mathcal{X}^{\sharp 2}_{\ell_6} = \mathbf{C}^\sharp_I[\![x > 10]\!]\mathcal{X}^{\sharp 1}_{\ell_3}
\end{array}$$

Computing the Abstract Least Fixpoint

$$\begin{array}{lll}
\mathcal{X}^{\sharp 0}_{\ell_1} = \bot & \mathcal{X}^{\sharp 1}_{\ell_1} = \top & \mathcal{X}^{\sharp 2}_{\ell_1} = \top \\
\mathcal{X}^{\sharp 0}_{\ell_2} = \bot & \mathcal{X}^{\sharp 1}_{\ell_2} = \bot & \mathcal{X}^{\sharp 2}_{\ell_2} = \bot[x \mapsto [1,1]] \\
\mathcal{X}^{\sharp 0}_{\ell_3} = \bot & \mathcal{X}^{\sharp 1}_{\ell_3} = \bot & \mathcal{X}^{\sharp 2}_{\ell_3} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_4} = \bot & \mathcal{X}^{\sharp 1}_{\ell_4} = \bot & \mathcal{X}^{\sharp 2}_{\ell_4} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_5} = \bot & \mathcal{X}^{\sharp 1}_{\ell_5} = \bot & \mathcal{X}^{\sharp 2}_{\ell_5} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_6} = \bot & \mathcal{X}^{\sharp 1}_{\ell_6} = \bot & \mathcal{X}^{\sharp 2}_{\ell_6} = \bot
\end{array}$$

Computing the Abstract Least Fixpoint

$$\begin{aligned}
\mathcal{X}^{\sharp 0}_{\ell_1} &= \bot \\
\mathcal{X}^{\sharp 0}_{\ell_2} &= \bot \\
\mathcal{X}^{\sharp 0}_{\ell_3} &= \bot \\
\mathcal{X}^{\sharp 0}_{\ell_4} &= \bot \\
\mathcal{X}^{\sharp 0}_{\ell_5} &= \bot \\
\mathcal{X}^{\sharp 0}_{\ell_6} &= \bot
\end{aligned}
\qquad
\begin{aligned}
\mathcal{X}^{\sharp 1}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp 1}_{\ell_2} &= \bot \\
\mathcal{X}^{\sharp 1}_{\ell_3} &= \bot \\
\mathcal{X}^{\sharp 1}_{\ell_4} &= \bot \\
\mathcal{X}^{\sharp 1}_{\ell_5} &= \bot \\
\mathcal{X}^{\sharp 1}_{\ell_6} &= \bot
\end{aligned}
\qquad
\begin{aligned}
\mathcal{X}^{\sharp 2}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp 2}_{\ell_2} &= \bot[x \mapsto [1,1]] \\
\mathcal{X}^{\sharp 2}_{\ell_3} &= \bot \\
\mathcal{X}^{\sharp 2}_{\ell_4} &= \bot \\
\mathcal{X}^{\sharp 2}_{\ell_5} &= \bot \\
\mathcal{X}^{\sharp 2}_{\ell_6} &= \bot
\end{aligned}
\qquad
\begin{aligned}
\mathcal{X}^{\sharp 3}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp 3}_{\ell_2} &= \mathbf{C}^{\sharp}_{I}[\![x \leftarrow 1]\!]\mathcal{X}^{\sharp 2}_{\ell_1} \\
\mathcal{X}^{\sharp 3}_{\ell_3} &= \mathcal{X}^{\sharp 2}_{\ell_2} \cup \mathcal{X}^{\sharp 2}_{\ell_5} \\
\mathcal{X}^{\sharp 3}_{\ell_4} &= \mathbf{C}^{\sharp}_{I}[\![x \leq 10]\!]\mathcal{X}^{\sharp 2}_{\ell_3} \\
\mathcal{X}^{\sharp 3}_{\ell_5} &= \mathbf{C}^{\sharp}_{I}[\![x \leftarrow x + 2]\!]\mathcal{X}^{\sharp 2}_{\ell_4} \\
\mathcal{X}^{\sharp 3}_{\ell_6} &= \mathbf{C}^{\sharp}_{I}[\![x > 10]\!]\mathcal{X}^{\sharp 2}_{\ell_3}
\end{aligned}$$

Computing the Abstract Least Fixpoint

$$\begin{array}{llll}
\mathcal{X}^{\sharp 0}_{\ell_1} = \bot & \mathcal{X}^{\sharp 1}_{\ell_1} = \top & \mathcal{X}^{\sharp 2}_{\ell_1} = \top & \mathcal{X}^{\sharp 3}_{\ell_1} = \top \\
\mathcal{X}^{\sharp 0}_{\ell_2} = \bot & \mathcal{X}^{\sharp 1}_{\ell_2} = \bot & \mathcal{X}^{\sharp 2}_{\ell_2} = \bot[x \mapsto [1,1]] & \mathcal{X}^{\sharp 3}_{\ell_2} = \bot[x \mapsto [1,1]] \\
\mathcal{X}^{\sharp 0}_{\ell_3} = \bot & \mathcal{X}^{\sharp 1}_{\ell_3} = \bot & \mathcal{X}^{\sharp 2}_{\ell_3} = \bot & \mathcal{X}^{\sharp 3}_{\ell_3} = \bot[x \mapsto [1,1]] \\
\mathcal{X}^{\sharp 0}_{\ell_4} = \bot & \mathcal{X}^{\sharp 1}_{\ell_4} = \bot & \mathcal{X}^{\sharp 2}_{\ell_4} = \bot & \mathcal{X}^{\sharp 3}_{\ell_4} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_5} = \bot & \mathcal{X}^{\sharp 1}_{\ell_5} = \bot & \mathcal{X}^{\sharp 2}_{\ell_5} = \bot & \mathcal{X}^{\sharp 3}_{\ell_5} = \bot \\
\mathcal{X}^{\sharp 0}_{\ell_6} = \bot & \mathcal{X}^{\sharp 1}_{\ell_6} = \bot & \mathcal{X}^{\sharp 2}_{\ell_6} = \bot & \mathcal{X}^{\sharp 3}_{\ell_6} = \bot
\end{array}$$

Computing the Least Fixpoint

Similarly to the concrete fixpoint, the abstract fixpoint is reached after 10 iterations.

```
^{ℓ1} x ← 1; ^{ℓ2}
while ^{ℓ3} x ≤ 10 do
    ^{ℓ4} x ← x + 2 ^{ℓ5}
done^{ℓ6}
```

$$\begin{aligned}
\mathcal{X}_{\ell_1}^{10} &= \top \\
\mathcal{X}_{\ell_2}^{10} &= \bot[x \mapsto [1,1]] \\
\mathcal{X}_{\ell_3}^{10} &= \bot[x \mapsto [1,11]] \\
\mathcal{X}_{\ell_4}^{10} &= \mathcal{X}_{\ell_3}^{9}[x \mapsto [1,9]] \\
\mathcal{X}_{\ell_5}^{10} &= \mathcal{X}_{\ell_4}^{9}[x \mapsto [3,11]] \\
\mathcal{X}_{\ell_6}^{10} &= \mathcal{X}_{\ell_3}^{9}[x \mapsto [11,11]]
\end{aligned}$$

Unbounded Loop

The previous computation of the fixpoint terminates in a finite number of steps, but that is not the case in general.

Suppose we bound the loop by n:

```
$^{\ell_1}$ $x \leftarrow 1;$ $^{\ell_2}$
while $^{\ell_3} x \leq n$ do
    $^{\ell_4}$ $x \leftarrow x + 2$ $^{\ell_5}$
done$^{\ell_6}$
```

$$\begin{aligned}
\mathcal{X}^m_{\ell_1} &= \top \\
\mathcal{X}^m_{\ell_2} &= \bot[x \mapsto [1, 1]] \\
\mathcal{X}^m_{\ell_3} &= \bot[x \mapsto [1, n+1]] \\
\mathcal{X}^m_{\ell_4} &= \mathcal{X}^{m-1}_{\ell_3}[x \mapsto [1, n-1]] \\
\mathcal{X}^m_{\ell_5} &= \mathcal{X}^{m-1}_{\ell_4}[x \mapsto [3, n+1]] \\
\mathcal{X}^m_{\ell_6} &= \mathcal{X}^{m-1}_{\ell_3}[x \mapsto [n+1, n+1]]
\end{aligned}$$

Unbounded Loop

The previous computation of the fixpoint terminates in a finite number of steps, but that is not the case in general.

Suppose we bound the loop by n:

```
$^{\ell_1}$ $x \leftarrow 1;$ $^{\ell_2}$
while $^{\ell_3} x \leq n$ do
    $^{\ell_4}$ $x \leftarrow x + 2$ $^{\ell_5}$
done$^{\ell_6}$
```

$$
\begin{aligned}
\mathcal{X}_{\ell_1}^m &= \top \\
\mathcal{X}_{\ell_2}^m &= \bot[x \mapsto [1,1]] \\
\mathcal{X}_{\ell_3}^m &= \bot[x \mapsto [1,n+1]] \\
\mathcal{X}_{\ell_4}^m &= \mathcal{X}_{\ell_3}^{m-1}[x \mapsto [1,n-1]] \\
\mathcal{X}_{\ell_5}^m &= \mathcal{X}_{\ell_4}^{m-1}[x \mapsto [3,n+1]] \\
\mathcal{X}_{\ell_6}^m &= \mathcal{X}_{\ell_3}^{m-1}[x \mapsto [n+1,n+1]]
\end{aligned}
$$

This example supposes we know n

- What if n is a very large constant? Slow convergence
- Worst, what if n is a variable such that $n \mapsto [-\infty, \infty]$ in the environment? Convergence at infinity only
- Question: What condition on $A^\sharp$ would allow to always converge in finitely many steps?

Widening

Definition

Let $\langle A^\sharp, \sqsubseteq\rangle$ be an abstract domain.
A widening is a function $\nabla : A^\sharp \times A^\sharp \to A^\sharp$ such that for all $x, y \in A^\sharp$:

$$x \sqsubseteq x \nabla y \qquad y \sqsubseteq x \nabla y$$

We say that ∇ is *terminating* if for any increasing sequence $x^1 \sqsubseteq x^2 \sqsubseteq \ldots$ and arbitrary sequence $y^1, y^2, \ldots$ such that $\forall k \in \mathbb{N},\ x^{k+1} = x^k \nabla y^k$, there exists $i \in \mathbb{N}$ such that $x^{i+1} = x^i$.

Interval widening

Let's define a widening over intervals (push unstable bounds to infinities):

- $\bot \nabla x \triangleq x \nabla \bot \triangleq x$
- $[a, b] \nabla [c, d] \triangleq [\llparenthesis\ a > c\ ?\ -\infty\ :\ a\ \rrparenthesis, \llparenthesis\ b < d\ ?\ \infty\ :\ b\ \rrparenthesis]$

Interval Widening

```
$^{\ell_1}$ $i \leftarrow 1;$ $^{\ell_2}$
while $^{\ell_3} i \leq n$ do
    $^{\ell_4}$ $i \leftarrow i + 2$ $^{\ell_5}$
done$^{\ell_6}$
```

$$\begin{aligned}
\mathcal{X}^{\sharp k+1}_{\ell_1} &= \top \\
\mathcal{X}^{\sharp k+1}_{\ell_2} &= \mathbf{C}^{\sharp}_{I}[\![i \leftarrow 1]\!] \mathcal{X}^{\sharp k}_{\ell_1} \\
\mathcal{X}^{\sharp k+1}_{\ell_3} &= \mathcal{X}^{\sharp k}_{\ell_3} \ \nabla \ \mathcal{X}^{\sharp k}_{\ell_2} \cup \mathcal{X}^{\sharp k}_{\ell_5} \\
\mathcal{X}^{\sharp k+1}_{\ell_4} &= \mathbf{C}^{\sharp}_{I}[\![i \leq 10]\!] \mathcal{X}^{\sharp k}_{\ell_3} \\
\mathcal{X}^{\sharp k+1}_{\ell_5} &= \mathbf{C}^{\sharp}_{I}[\![i \leftarrow i + 2]\!] \mathcal{X}^{\sharp k}_{\ell_4} \\
\mathcal{X}^{\sharp k+1}_{\ell_6} &= \mathbf{C}^{\sharp}_{I}[\![i > 10]\!] \mathcal{X}^{\sharp k}_{\ell_3}
\end{aligned}$$

Focus on $\mathcal{X}^{\sharp}_{\ell_3}$

$$\begin{aligned}
\mathcal{X}^{\sharp 2}_{\ell_3} &= \bot \nabla \bot = \bot \\
\mathcal{X}^{\sharp 3}_{\ell_3} &= \mathcal{X}^{\sharp 2}_{\ell_3} \nabla \bot[x \mapsto [1,1]] = \bot[x \mapsto [1,1]] \\
\mathcal{X}^{\sharp 4}_{\ell_3} &= \mathcal{X}^{\sharp 3}_{\ell_3} \nabla \bot[x \mapsto [1,3]] = \bot[x \mapsto [1,\infty]] \\
\mathcal{X}^{\sharp 5}_{\ell_3} &= \mathcal{X}^{\sharp 4}_{\ell_3} \nabla \bot[x \mapsto [1,\infty]] = \bot[x \mapsto [1,\infty]]
\end{aligned}$$

Widening helps to enforce convergence at the cost of a loss of precision.

Soundness

Soundness

Let $\langle C, \leq \rangle$ be the concrete domain and $\langle A, \sqsubseteq \rangle$ the abstract domain.

Definition

- A *transformer* is an order-preserving function $f : C \to C$ (e.g., $\mathbf{C}[\![.]\!]$ or $\mathbf{eq}(.)$).
- An *abstract transformer* is an order-preserving function $\overline{f} : A \to A$ (e.g., $\mathbf{C}^\sharp[\![.]\!]$ or $\mathbf{eq}^\sharp(.)$).

$$\text{Soundness: } \mathbf{lfp}^{\leq}\, f \leq \gamma(\mathbf{lfp}^{\sqsubseteq}\, \overline{f})$$

(we say $\mathbf{lfp}^{\sqsubseteq}\, \overline{f}$ is a sound fixpoint overapproximation of $\mathbf{lfp}^{\leq}\, f$.)

What are the conditions required on A and its abstract transformers to satisfy soundness?

Soundness

Theorem (Sound transformer abstraction [Cou21] Th. 18.3)

If $\langle C, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \sqsubseteq \rangle$ *then* $\langle C \xrightarrow{\nearrow} C, \dot{\leq} \rangle \underset{\overrightarrow{\alpha}}{\overset{\overrightarrow{\gamma}}{\leftrightarrows}} \langle A \xrightarrow{\nearrow} A, \dot{\sqsubseteq} \rangle$ *with:*

$$\overrightarrow{\alpha}(f) \triangleq \alpha \circ f \circ \gamma$$
$$\overrightarrow{\gamma}(\overline{f}) \triangleq \gamma \circ \overline{f} \circ \alpha$$

To abstract a least fixpoint $\alpha(\mathbf{lfp}^{\leq} f)$, we abstract its transformer into an abstract transformer $\alpha \circ f \circ \gamma \in A \to A$.

From concrete to abstract transformers

We could define $\mathbf{C}^{\sharp}_{I}[\![x \leq y]\!] \triangleq \alpha \circ \mathbf{C}[\![x \leq y]\!] \circ \gamma$.

Soundness

Theorem (Least fixpoint overapproximation in a complete lattice [Cou21] Th. 18.10)

Let $\langle C, \leq\rangle$ *and* $\langle A, \sqsubseteq\rangle$ *be complete lattices,* $\langle C, \leq\rangle \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \langle A, \sqsubseteq\rangle$ *and* $f \in C \to C$ *order-preserving.*
Then $\mathbf{lfp}^{\leq}\, f \leq \gamma(\mathbf{lfp}^{\sqsubseteq}\, \alpha \circ f \circ \gamma)$.

Proof.

$$\begin{array}{cll}
 & \mathbf{lfp}^{\leq}\, f & \\
= & \bigwedge\{x \in C \mid f(x) \leq x\} & \textit{(by Tarski's fixpoint theorem)} \\
\leq & \bigwedge\{\gamma(\overline{x}) \mid f(\gamma(\overline{x})) \leq \gamma(\overline{x})\} & \\
= & \gamma(\bigsqcap\{\overline{x} \in A \mid f(\gamma(\overline{x})) \leq \gamma(\overline{x})\}) & \textit{(}\gamma\textit{ preserves arbitrary meet)} \\
= & \gamma(\bigsqcap\{\overline{x} \in A \mid (\alpha \circ f \circ \gamma)(\overline{x}) \sqsubseteq \overline{x}\}) & \textit{(by } \langle C, \leq\rangle \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \langle A, \sqsubseteq\rangle\textit{)} \\
= & \gamma(\mathbf{lfp}^{\sqsubseteq}\, \alpha \circ f \circ \gamma) & \textit{(by Tarski's fixpoint theorem)}
\end{array}$$

□

Soundness

$\alpha \circ f \circ \gamma$ is convenient from a mathematical perspective but not usuable in practice as α, γ and f might not be computable.
Hence, we approximate this definition.

Theorem ([Cou21] Th. 18.7)

Let $\langle C, \leq \rangle$ be a complete lattice and $f, g \in C \to C$ order-preserving.
If $f \;\dot{\leq}\; g$ then $\mathbf{lfp}^{\leq}\, f \;\sqsubseteq\; \mathbf{lfp}^{\leq}\, g$.

Corollary

Let $\alpha \circ f \circ \gamma \;\dot{\sqsubseteq}\; \overline{f}$. Then $\mathbf{lfp}^{\leq}\, f \;\leq\; \gamma(\mathbf{lfp}^{\sqsubseteq}\, \overline{f})$.

Proof.

By Th. 18.7 and Th. 18.10. □

Other Concepts of Abstract Interpretation

Many techniques to improve precision

- **Various abstract domains** with different precision/efficiency tradeoff (replacing intervals in the previous example).
- **Various products of abstract domains** to combine their strengths.
- More efficient fixpoint algorithms (narrowing, chaotic iterations, . . .).
- . . .

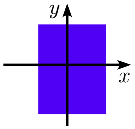
Course plan (2/8)

Bricks of abstraction: numerical domains

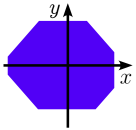
simple domains

relational domains

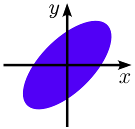
specific domains



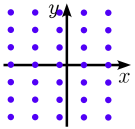
Intervals
$x \in [a, b]$



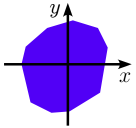
Octagons
$\pm x \pm y \leq c$



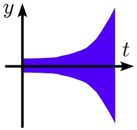
Ellipsoids
digital filters



Congruences
$x \in a\mathbb{Z} + b$



Polyhedra
$\sum_i \alpha_i x_i \leq \beta$



Exponentials
rounding errors

Conclusion

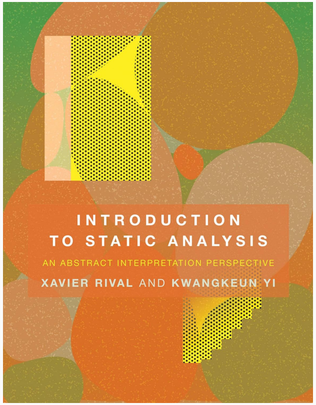
Universality of Lattice Theory and Abstract Interpretation

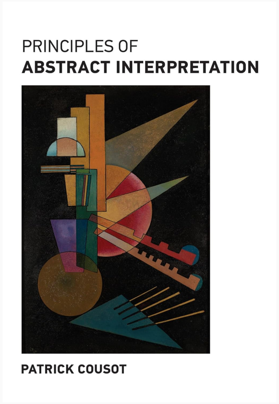
Abstraction and approximation are two central concepts in computer science. Abstract interpretation captures those precisely, thus has many applications beyond program analysis:

- Constraint reasoning.
- Neural network verification.
- (Gradual) typing.
- Conflict-free replicated data types (CRDTs).
- Parallel computing.

Resources

- MPRI class of Antoine Miné: `https://www-apr.lip6.fr/~mine/enseignement/mpri/2023-2024/` (two slides stolen from this class).
- Two recent books:


INTRODUCTION TO STATIC ANALYSIS
AN ABSTRACT INTERPRETATION PERSPECTIVE
XAVIER RIVAL AND KWANGKEUN YI



References

[Cou21] Patrick Cousot. **Principles of abstract interpretation.** MIT Press, 2021.