Abstract Induction
Concrete Induction
Software correctness proofs

• Any formal proof of a non-trivial program requires a reasoning by **mathematical induction** (e.g., following Turing, on the number of program execution steps):

  • Invent an **inductive argument** (e.g. invariant, variant function), **the hardest part**

  • Prove the **base case and inductive case** (e.g. true on loop entry and preserved by one more loop iteration)

  • Prove that the inductive argument is **strong-enough**, that is, it implies the program property to be verified
Avoiding the difficulties:
(1) finitary methods
Avoiding the difficulty

• **Unsoundness:** not for scientists

• **Model-checking:** finite enumeration, no induction needed

• **Deductive methods** (theorem provers, proof verifiers, SMT solvers): avoid (part of) the difficulty since the inductive argument must be provided by the end-user (⟹ still difficult, shame is on the prover)

• **Finitary abstractions** (predicate abstraction ≡ any finite abstract domain): only finitely many possible statements to be checked to be inductive
Limitations of finite abstractions

• A sound and complete finite abstraction exists to prove any property of any program:

  \[
  \begin{align*}
  x=0; \text{ while } x<1 \text{ do } x++ & \longrightarrow \{ \bot, [0,0], [0,1], [-\infty, \infty] \} \\
  x=0; \text{ while } x<2 \text{ do } x++ & \longrightarrow \{ \bot, [0,0], [0,1], [0,2], [-\infty, \infty] \} \\
  \ldots \\
  x=0; \text{ while } x<n \text{ do } x++ & \longrightarrow \{ \bot, [0,0], [0,1], [0,2], [0,3], \ldots, [0,n], [-\infty, \infty] \} \\
  \ldots
  \end{align*}
  \]

• Not true for a programming language!

• Finite abstractions fail on infinitely many programs on which infinitary abstractions do succeed.
Avoiding the difficulty
(II) Refinement in finite domains
Verification/static analysis by abstract interpretation

• Define the abstraction:

\[ \langle \wp(\mathcal{D}[P]), \subseteq \rangle \xrightarrow{\gamma[P]} \langle \mathcal{A}[P], \subseteq \rangle \xleftarrow{\alpha[P]} \langle \wp(\mathcal{D}[P]), \subseteq \rangle \]

• Calculate the abstract semantics:

\[ S^\#[P] = \alpha[P](\{S[P]\}) \] exact abstraction

\[ S^\#[P] \sqsubseteq \alpha[P](\{S[P]\}) \] approximate abstraction

• Soundness (by construction):

\[ \forall P \in \mathcal{L}: \forall Q \in \mathcal{A}: S^\#[P] \sqsubseteq Q \implies S[P] \in \gamma[P](Q) \]
Refinement: good news

- **Problem**: how to prove a valid abstract property $\alpha(\{\text{lfp } F[\mathcal{P}] \}) \subseteq Q$ when $\alpha \circ F \subseteq F^\# \circ \alpha$ but $\text{lfp } F^\# [\mathcal{P}] \not\subseteq Q$? (i.e. strongest inductive argument too weak)

- It is **always** possible to refine $\langle \mathcal{A}, \subseteq \rangle$ into a most abstract more precise abstraction $\langle \mathcal{A}', \subseteq' \rangle$ such that

$$
\langle \wp(\mathcal{D}), \subseteq \rangle \xleftarrow{\gamma'} \xrightarrow{\alpha'} \langle \mathcal{A}', \subseteq' \rangle
$$

and $\alpha' \circ F = F' \circ \alpha$ with $\text{lfp } F'[\mathcal{P}] \subseteq' \alpha' \circ \gamma(Q)$

(thus proving $\text{lfp } F[\mathcal{P}] \in \gamma'(Q)$ which implies $\text{lfp } F[\mathcal{P}] \in \gamma(Q)$)

Refinement: bad news

- But, refinements of an abstraction can be intrinsically incomplete

- The only complete refinement of that abstraction for the collecting semantics is:

  the identity (i.e. no abstraction at all)

- In that case, the only complete refinement of the abstraction is to the collecting semantics and any other refinement is always imprecise
Example of intrinsic approximate refinement

- Consider executions traces \( \langle i, \sigma \rangle \) with infinite past and future:

![Diagram showing states, time origin, present time, past, and future with \( \sigma_{-2}, \sigma_{-1}, \sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_i \) illustrating the states and time progression.]

---

Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25
Example of intrinsic approximate refinement

- Consider the temporal specification language \( \mu^* \) (containing LTL, CTL, CTL*, and Kozen’s \( \mu \)-calculus as fragments):

  \[
  \varphi ::= \sigma_S \quad S \in \varphi(S) \quad \text{state predicate}
  \]

  \[
  \mid \pi_t \quad t \in \varphi(S \times S) \quad \text{transition predicate}
  \]

  \[
  \mid \oplus \varphi_1 \quad \text{next}
  \]

  \[
  \mid \varphi_1 \circ \quad \text{reversal}
  \]

  \[
  \mid \varphi_1 \lor \varphi_2 \quad \text{disjunction}
  \]

  \[
  \mid \neg \varphi_1 \quad \text{negation}
  \]

  \[
  \mid X \quad X \in X \quad \text{variable}
  \]

  \[
  \mid \mu X \cdot \varphi_1 \quad \text{least fixpoint}
  \]

  \[
  \mid \nu X \cdot \varphi_1 \quad \text{greatest fixpoint}
  \]

  \[
  \mid \forall \varphi_1 : \varphi_2 \quad \text{universal state closure}
  \]

Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25
Example of intrinsic approximate refinement

- Consider universal model-checking abstraction:

\[
MC^\forall_M(\phi) = \alpha^\forall_M(\lbrack \phi \rbrack) \in \varphi(\text{Traces}) \rightarrow \varphi(\text{States})
\]

\[
= \{ s \in \text{States} \mid \forall \langle i, \sigma \rangle \in \text{Traces}_M. (\sigma_i = s) \Rightarrow \langle i, \sigma \rangle \in \lbrack \phi \rbrack \}
\]

where \( M \) is defined by a transition system

(and dually the existential model-checking abstraction)
Example of intrinsic approximate refinement

- The abstraction from a set of traces to a trace of sets is sound but *incomplete*, even for finite systems (*)

\[ \begin{array}{c}
\alpha \\
\gamma
\end{array} \]

- Any refinement of this abstraction is *incomplete* (but to the infinite past/future trace semantics itself) (**)

(*) Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Intrinsic approximate refinement

\[ \forall P \in \mathcal{P}(\mathcal{D}) \cdot \mathcal{D} \]

\[ \lambda P \in \mathcal{P}(\mathcal{D}) \cdot P \]

poset of abstractions

complete abstractions

refinements of \( \alpha \)

abstraction poset
In general refinement does not terminate

- **Example:** filter invariant abstraction:

2nd order filter:

Unstable polyhedral abstraction:

Counter-example guided refinement will indefinitely add missing points according to the execution trace:

Stable ellipsoidal abstraction:

In general refinement does not terminate

- Narrowing is needed to stop infinite iterated automatic refinements:
  
  e.g. SLAM stops refinement after 20mn, now abandoned (despite complete success claimed in 98% of studied cases (*)

- Intelligence is needed for refinement:
  
  e.g. human-driven refinement of Astrée (**)


Facing the difficulties: Abstract induction
Sound software static analysis

• The **mathematical induction** must be performed in the **abstract** (e.g. the inductive argument must belong to an abstract domain with a finite computer representation)

• (and imply the mathematical induction in the **concrete**)**
Abstract induction

• The inductive argument must be expressible in the abstract domain (complex abstract domains favored)

• It must be strong enough to imply the program property (complex abstract domains favored)

• It must be inferable in the abstract (simple abstract domains favored)
Abstract induction in infinite domains
Abstract Interpreters

• **Transitional abstract interpreters**: proceed by induction on program steps

• **Structural abstract interpreters**: proceed by induction on the program syntax

• **Common main problem**: over/under-approximate fixpoints in non-Noetherian\(^*\) abstract domains \(^**\)

\(^*\) Iterative fixpoint computations may not converge in finitely many steps

\(^**\) Or convergence may be guaranteed but too slow.
Fixpoints

- **Poset** (or pre-order) \(< D, \subseteq, \bot, \cup >\)

- **Transformer** (increasing in the concrete) \( F \in D \mapsto D \)

- **Least fixpoint**: \( \text{lfp} \subseteq F = \bigsqcup_{n \in \mathbb{N}} F^n(\bot) \) (under appropriate hypotheses)
Convergence criterion

• By Tarski (or variants)

\[ F(X) \subseteq X \quad \Rightarrow \quad \text{lfp} \subseteq F \subseteq X \]
Widening
Convergence acceleration with widening

Infinite iteration
Convergence acceleration with widening

Infinite iteration

Accelerated iteration with widening
(e.g. with a widening based on the derivative as in Newton-Raphson method)

Extrapolation by Widening

- \( X^0 = \bot \) (increasing iterates with widening)

\[
X^{n+1} = X^n \triangledown F(X^n) \quad \text{when} \quad F(F(X^n)) \not\subseteq F(X^n)
\]

\[
X^{n+1} = F(X^n) \quad \text{when} \quad F(F(X^n)) \subseteq F(X^n)
\]

- Widening \( \triangledown \), two independent hypotheses:
  - \( Y \subseteq X \triangledown Y \) (extrapolation)
  - Enforces convergence of increasing iterates with widening (to a limit \( X^\ell \))
The oldest widenings

- **Primitive widening** [1,2]

\[(x \wedge y) = \begin{cases} 
\text{cas } x \in V, y \in V \text{ dans} \\
\text{if } x \text{ then } \infty \text{ else } x \\
\text{if } x \text{ then } x \text{ else } \infty \\
[n_1, m_1], [n_2, m_2] \Rightarrow \\
\text{si } n_2 < n_1 \text{ alors } \infty \text{ sinon } n_1 \text{ fsi} \\
\text{si } m_2 > m_1 \text{ alors } \infty \text{ sinon } m_1 \text{ fsi} \\
\text{fincas} 
\end{cases} \]

\[[a_1, b_1] \wedge [a_2, b_2] = \\
\begin{cases} 
\text{if } a_2 < a_1 \text{ then } \infty \text{ else } a_1 \text{ fsi,} \\
\text{if } b_2 > b_1 \text{ then } \infty \text{ else } b_1 \text{ fsi} 
\end{cases} \]

- **Widening with thresholds** [3]

\[\forall x \in \mathcal{L}_2, \bot \uparrow \mathcal{V}_2(j) \downarrow = x \]

\[[l_1, u_1] \mathcal{V}_2(j) [l_2, u_2] \\
= \begin{cases} 
\text{if } 0 \leq l_2 < l_1 \text{ then } 0 \text{ elsif } l_2 < l_1 \text{ then } -b - 1 \text{ else } l_1 \text{ fsi,} \\
\text{if } u_1 < u_2 \leq 0 \text{ then } 0 \text{ elsif } u_1 < u_2 \text{ then } b \text{ else } u_1 \text{ fsi} 
\end{cases} \]


Extrapolation with widening

\[ X \sqsubseteq F(X) \]

\[ F(X) \sqsubseteq X \]

\[ \frac{X}{F(X)} \]

\[ X^0 \]

\[ X^1 \]

\[ \cdots \]

\[ lfp^\subseteq F \]

\[ F(X) = X \]

\[ Y^\lambda \]

\[ X^\ell \]

\[ T \]
Widenings are not increasing

- A well-known fact
  
  \([1,1] \subseteq [1,2] \) but \([1,1] \vee [1,2] = [1,\infty] \subseteq [1,2] \vee [1,2] = [1,2] \)

- A widening cannot both:
  - Be increasing in its first parameter
  - Enforce termination of the iterates
  - Avoid useless over-approximations as soon as a solution is found(*)

(*) A counter-example is \(x \vee y = \top\)
Narrowing
Interpolation with narrowing

- \( Y^0 = X^\ell \) (decreasing iterates with narrowing)
  
  \[ Y^{n+1} = Y^n \triangle F(Y^n) \quad \text{when} \quad F(F(Y^n)) \sqsubseteq F(Y^n) \]

  \[ Y^{n+1} = F(Y^n) \quad \text{when} \quad F(F(Y^n)) = F(Y^n) \]

- Narrowing \( \triangle \), two independent hypotheses:

- \( Y \subseteq X \implies Y \subseteq X \triangle Y \subseteq X \) \quad (interpolation)

- Enforces *convergence* of decreasing iterates with narrowing (to a limit \( Y^\lambda \))
The oldest narrowing

- [2]

\[ [a_1, b_1] \Delta [a_2, b_2] = \]

\[ \begin{align*}
&\text{if } a_1 = -\infty \text{ then } a_2 \text{ else MIN } (a_1, a_2), \\
&\text{if } b_1 = +\infty \text{ then } b_2 \text{ else MAX } (b_1, b_2) \end{align*} \]
Interpolation with narrowing

Could stop when $F(X) \not\sqsubseteq X \land F(F(X)) \sqsubseteq F(X)$ but not the current practice.
Duality
Duality

- **Extrapolators:**

- **Interpolators:**

<table>
<thead>
<tr>
<th></th>
<th>Convergence above the limit</th>
<th>Convergence below the limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing iteration</td>
<td>Widening $\nabla$</td>
<td>Dual-narrowing $\widetilde{\Delta}$</td>
</tr>
<tr>
<td>Decreasing iteration</td>
<td>Narrowing $\Delta$</td>
<td>Dual widening $\widetilde{\nabla}$</td>
</tr>
</tbody>
</table>

Extrapolators $(\nabla, \widetilde{\nabla})$ and interpolators $(\Delta, \widetilde{\Delta})$
Extrapolators, Interpolators, and Duals

\[ X \subseteq F(X) \quad X \not\subseteq F(X) \]

\[ X = F(X) \]

\[ F(X) \subseteq X \]

\[ \perp \]

\[ \top \]

\[ F \]
Multi-step extrapolators/interpolators

- The extrapolators/interpolators can be on
  - the last two iterates
  - a bounded number of previous iterates
  - all previous iterates

- Examples:
  - loop unrolling
  - delayed widening
  - etc
Dual narrowing
Interpolation with dual narrowing

- \( Z^0 = \perp \) (increasing iterates with dual-narrowing)

\[
Z^{n+1} = F(Z^n) \sim Y^\lambda \quad \text{when} \quad F(F(Z^n)) \not\subseteq F(Z^n)
\]

\[
Z^{n+1} = F(Z^n) \quad \text{when} \quad F(F(Z^n)) \subseteq F(Z^n)
\]

- Dual-narrowing \( \sim \), two independent hypotheses:

  - \( X \subseteq Y \implies X \subseteq Y \sim X \subseteq Y \) (interpolation)

- Enforces convergence of increasing iterates with dual-narrowing
Example of dual-narrowing

- \([a, b] \tilde{\Delta} [c, d]\)

- \([a, b] \tilde{\Delta} [c, d] \triangleq ([c = -\infty \land a \land (a + c)/2], [d = \infty \land b \land (b + d)/2])\)

- The first method we tried in the late 70’s with Radhia
  - Slow
  - Does not easily generalize (e.g. to pointer analysis)
Interpolation with dual-narrowing

- Refine widening/narrowing iterations $Y^\lambda$
- Refine a user-defined specification (Craig interpolation)
Craig interpolation

• Craig interpolation:

Given $P \implies Q$ find $I$ such that $P \implies I \implies Q$ with

\[ \text{var}(I) \subseteq \text{var}(P) \cap \text{var}(Q) \]

is a dual narrowing (already observed by Vijay D’Silva and Leopold Haller as a narrowing [indeed inversed!])

• May not be unique

• May not terminate
Relationship between narrowing and dual-narrowing

- $\tilde{\Delta} = \Delta^{-1}$

- $Y \subseteq X \implies Y \subseteq X \Delta Y \subseteq X$ (narrowing)

- $Y \subseteq X \implies Y \subseteq Y \tilde{\Delta} X \subseteq X$ (dual-narrowing)
Bounded widening
Dual-narrowing versus bounded widening

- **Dual-narrowing:**

  \[ F(X) \subseteq B \iff F(X) \subseteq F(X) \Delta \tilde{B} \subseteq B \]

  Induction on \( F(X) \) and \( B \)

- **Bounded widening:**

  \[ X \subseteq F(X) \subseteq B \iff F(X) \subseteq X \vee_{B} F(X) \subseteq B \]

  Induction on \( X \), \( F(X) \), and \( B \)
Example of widenings (cont’d)

- Bounded widening (in $[\ell, h]$):

$$[a, b] \triangledown_{[\ell, h]} [c, d] \triangleq \left[ \frac{c + a - 2\ell}{2}, \frac{b + d + 2h}{2} \right]$$
Soundness
Soundness

• Fixpoint approximation soundness theorems can be expressed with minimalist hypotheses (*):

• No need for complete lattices, complete partial orders (CPO’s):
  • The concrete domain is a poset
  • The abstract domain is a pre-order
  • The concretization is defined for the abstract iterates only.

Soundness (cont’d)

• No need for increasingness/monotony hypotheses for fixpoint theorems (Tarski, Kleene, etc)

  • The concrete transformer is increasing and the limit of the iterations does exist in the concrete domain

  • No monotonicity hypotheses on the abstract transformer (no need for fixpoints in the abstract)

  • Soundness hypotheses on the extrapolators/interpolators with respect to the concrete

• In addition, the independent termination hypotheses on the extrapolators/interpolators ensure convergence in finitely many steps
The challenge of verification

- Infer the *inductive argument*

- Without *deep knowledge* about the program (e.g. very precise, quasi-inductive, quasi-strong enough specification)

- *Scale*
Infer the abstract inductive argument

typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;

void filter () {
    static float E[2], S[2];
    if (INIT) { S[0] = X; P = X; E[0] = X; }
    else { P = ((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
                + (S[0] * 1.5)) - (S[1] * 0.7)); }
    E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
    /* S[0], S[1] in ?????????????????????????????????????????????????? */
}

void main () { X = 0.2 * X + 5; INIT = TRUE;
    while (1) {
        X = 0.9 * X + 35; /* simulated filter input */
        filter (); INIT = FALSE; }
}
Infer the abstract inductive argument

typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;

void filter () {
    static float E[2], S[2];
    if (INIT) { S[0] = X; P = X; E[0] = X; }
    else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
        + (S[0] * 1.5)) - (S[1] * 0.7)); }
    E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
    /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
    while (1) {
        X = 0.9 * X + 35; /* simulated filter input */
        filter (); INIT = FALSE; }
}
Extrapolation/Interpolation

- Abstract interpretation in infinite domains is traditionally by iteration with widening/narrowing.
- We have shown how to use iteration with dual-narrowing.
- These ideas of the 70's generalize Craig interpolation from logic to arbitrary abstract domains.
- Can be used to improve precision when a fixpoint is reached after the widening/narrowing iterations.
The End, Thank You