Abstract Induction

ETH Zürich
October 2–3, 2015

Patrick Cousot
pcousot@cs.nyu.edu  cs.nyu.edu/~pcousot

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:
  
  \[
  x=0; \text{ while } x<1 \text{ do } x++ \rightarrow \{\bot, [0,0], [0,1], [-\infty, \infty]\}
  \]
  
  \[
  x=0; \text{ while } x<2 \text{ do } x++ \rightarrow \{\bot, [0,0], [0,1], [0,2], [-\infty, \infty]\}
  \]
  
  ... 
  
  \[
  x=0; \text{ while } x<n \text{ do } x++ \rightarrow \{\bot, [0,0], [0,1], [0,2], [0,3], \ldots, [0,n], [-\infty, \infty]\}
  \]
  
  ... 

- Not true for a programming language!
- Finite abstractions fail on infinitely many programs on which infinitary abstractions do succeed

Verification/static analysis by abstract interpretation

- Define the abstraction:
  
  \[
  \langle \forall (\forall \mathcal{D} \mathcal{P}), \subseteq \rangle \xrightarrow{\gamma [\mathcal{P}]} \langle \mathcal{A} \mathcal{P}, \subseteq \rangle \]

- Calculate the abstract semantics:

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

  \[
  S^# [\mathcal{P}] \supseteq \alpha [\mathcal{P}] ([S [\mathcal{P}]]) \quad \text{approximate abstraction}
  \]

- **Soundness** (by construction):

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

- Problem: how to prove a valid abstract property $\alpha(\text{lf} F [\mathcal{P}]) \subseteq Q$ when $\alpha \circ F \subseteq F' \circ \alpha$ but $\text{lf} F'[\mathcal{P}] \not\subseteq Q$? (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

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

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

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

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:

```
+---+---+---+---+---+---+---+---+---+---+
|   |   | i | σ1 | σ2 | σ3 | σ4 | σ5 | σ6 | σ7 |
+---+---+---+---+---+---+---+---+---+---+
```

```
states | time origin | present time | past | future
```

Consider the temporal specification language $\mu$:

(containing LTL, CTL, CTL*, and Kozen’s $\mu$-calculus as fragments):

$$\varphi ::= \sigma_S \quad S \in \wp(S) \quad \text{state predicate}$$

$$\mid \pi_t \quad t \in \wp(S \times S) \quad \text{transition predicate}$$

$$\mid \oplus \varphi_1 \quad \text{next}$$

$$\mid \varphi_1 \lor \varphi_2 \quad \text{disjunction}$$

$$\mid \neg \varphi_1 \quad \text{negation}$$

$$\mid X \quad X \in X \quad \text{variable}$$

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

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

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

Example of intrinsic approximate refinement
Example of intrinsic approximate refinement

- Consider universal model-checking abstraction:
  \[ \text{MC}_M^\forall(\phi) = \alpha_M^\forall([\phi]) \in \varphi(\text{Traces}) \rightarrow \varphi(\text{States}) \]
  \[ = \{ s \in \text{States} \mid \forall (i, \sigma) \in \text{Traces}_M . (\sigma_i = s) \Rightarrow (i, \sigma) \in [\phi] \} \]

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 \(^{(3)}\)

- Any refinement of this abstraction is incomplete (but to the infinite past/future trace semantics itself) \(^{(6)}\)

---

Intrinsic approximate refinement

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

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

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:**

---

\(^{(3)}\) Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25
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 (**) (**) Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. In AIAA Infotech@Aerospace 2010, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.


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)

Facing the difficulties: 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

Fixpoints
- Poset (or pre-order) \(<D, \sqsubseteq, \bot, \sqcup>\)
- Transformer (increasing in the concrete) \(F \in D \mapsto D\)
- Least fixpoint: \(\text{lfp}^c F = \bigcup_{n \in \mathbb{N}} F^n(\bot)\) (under appropriate hypotheses)

Convergence criterion
- By Tarski (or variants)
  \[ F(X) \subseteq X \implies \text{lfp}^c F \subseteq X \]
Convergence acceleration with widening

- **Extrapolation by Widening**
  - \( X^0 = \bot \) (increasing iterates with widening)
  - \( X^{n+1} = X^n \triangleleft F(X^n) \) when \( F(F(X^n)) \not\subseteq F(X^n) \)
  - \( X^{n+1} = F(X^n) \) when \( F(F(X^n)) \subseteq F(X^n) \)
  - **Widening** \( \triangleleft \), two independent *hypotheses*:
    - \( Y \subseteq X \triangleleft Y \) (extrapolation)
  - Enforces *convergence* of increasing iterates with widening (to a limit \( X^\ell \))

The oldest widenings

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

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

- **Widening with thresholds** [3]

\[ \forall x \in \mathbb{L}_2, \perp \mathbb{V}_2(j)x = x \mathbb{V}_2(j) \perp = x \]

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

Extrapolation with widening

Widenings are not increasing

- A well-known fact

\[ [1,1] \subseteq [1,2] \text{ but } [1,1] \mathbb{V} [1,2] = [1,\infty] \subseteq [1,2] \mathbb{V} [1,2] = [1,2] \]

- 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 \( x \mathbb{V} y = T \)
Interpolation with narrowing

- \( Y^0 = X^\ell \) (decreasing iterates with narrowing)

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

- Narrowing \( \Delta \), two independent hypotheses:
  - \( Y \subseteq X \implies Y \subseteq X \Delta Y \subseteq X \) (interpolation)
  - Enforces convergence of decreasing iterates with narrowing (to a limit \( Y^\lambda \) )

Could stop when \( F(X) \not\subseteq X \wedge F(F(X)) \not\subseteq F(X) \) but not the current practice.
Duality

<table>
<thead>
<tr>
<th>Convergence above the limit</th>
<th>Convergence below the limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing iteration</td>
<td>Widening $\bigvee$</td>
</tr>
<tr>
<td>Decreasing iteration</td>
<td>Narrowing $\Delta$</td>
</tr>
</tbody>
</table>

Extrapolators ($\bigvee$, $\bigvee$) and interpolators ($\Delta$, $\Delta$)

- **Extrapolators:**

- **Interpolators:**

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
**Interpolation with dual narrowing**

- $Z^0 = \bot$ (increasing iterates with dual-narrowing)
  
  $Z^{n+1} = F(Z^n) \Delta Y^n$ when $F(F(Z^n)) \not\subseteq F(Z^n)$
  
  $Z^{n+1} = F(Z^n)$ when $F(F(Z^n)) \subseteq F(Z^n)$
  
- Dual-narrowing $\Delta$, two independent hypotheses:
  
  - $X \subseteq Y \implies X \subseteq Y \Delta X \subseteq Y$ (interpolation)
  
- Enforces convergence of increasing iterates with dual-narrowing

**Example of dual-narrowing**

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

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

**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 inverted narrowing!])

  - May not be unique
  - May not terminate

- Refine widening/narrowing iterations $Y^\lambda$

- Refine a user-defined specification (Craig interpolation)
Relationship between narrowing and dual-narrowing

- $\Delta = \Delta^{-1}$
- $Y \subseteq X \Rightarrow Y \subseteq X \Delta Y \subseteq X$ (narrowing)
- $Y \subseteq X \Rightarrow Y \subseteq Y \Delta X \subseteq X$ (dual-narrowing)

Note: effectiveness and termination conditions may be different

Dual-narrowing versus bounded widening

- Dual-narrowing $\tilde{\Delta}$:
  \[ F(X) \subseteq B \quad \Rightarrow \quad F(X) \subseteq F(X) \tilde{\Delta} B \subseteq B \]
  Induction on $F(X)$ and $B$
- Bounded widening $\nabla_B$:
  \[ X \subseteq F(X) \subseteq B \quad \Rightarrow \quad F(X) \subseteq X \nabla_B F(X) \subseteq B \]
  Induction on $X$, $F(X)$, and $B$

Example of widenings (cont’d)

- Bounded widening (in $[\ell, h]$):
  \[ [a, b] \nabla_{[\ell, h]} [c, d] \triangleq \frac{[c+a-2\ell, b+d+2h]}{2} \]
Soundness

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

Conclusion
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; }
}
The End, Thank You