Construction of invariance proof methods for parallel programs (with sequential consistency)

Patrick Cousot

NYU, NYC, NY
p.cousot@cims.nyu.edu

History

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1940</td>
<td>Turing</td>
<td>Invents invariance + termination proofs for sequential programs.</td>
</tr>
<tr>
<td>1966</td>
<td>Naur</td>
<td>Re-invents invariance proof.</td>
</tr>
<tr>
<td>1967</td>
<td>Floyd</td>
<td>Re-invents invariance + termination proofs.</td>
</tr>
<tr>
<td>1968</td>
<td>Hoare</td>
<td>Invents structural induction (in HL).</td>
</tr>
</tbody>
</table>

... thousands of (forgotten) publications

Ouchi & Eiyo (1975) | generalize HL to parallel processes with sequential consistency (SC) (complete without auxiliary variables)

Lamport (1977) | generalize Turing/Floyd/Naur for parallel processes with SC (complete thanks to program counters)

DEFINITION OF INVARIANCE
BASED ON AN OPERATIONAL SEMANTICS

History (cont’d)

Radhia Cousot (1980): all this is abstract interpretation.

Today: researchers reinvent everything for weak memory models (WHM)

--> based on Ouchi & Eiyo (incomplete!)

--> empirically, without any methodology.

Objective:

Explain a methodology for designing an invariance proof method by abstract interpretation of an operational semantics of the language.
Operational semantics of a sequential process

- States: \( \langle c, m \rangle \in S \)
  - Memory state \( m(x) \) is the value of (shared) variable \( x \)
  - Control point specifies what remains to be executed in the program
- Transitions: \( \Sigma \subseteq (S \times S) \)

\[ \langle c, m \rangle \xrightarrow{t} \langle c', m' \rangle \]

\( \text{i.e. } \langle c, m \rangle, \langle c', m' \rangle \in t \)

- Execution of a computation step of the process at control point \( c \) in memory state \( m \) moves to control point \( c' \) in new memory state \( m' \).

Example

1: while \( x < 10 \) do
2: \( x := x + 1 \)
3: \( \text{end} \)

\[ \begin{align*}
\langle 1, m \rangle &\xrightarrow{t} \langle 2, m \rangle & \text{if } m(x) < 10 \\
\langle 1, m \rangle &\xrightarrow{t} \langle 3, m \rangle & \text{if } m(x) \geq 10 \\
\langle 2, m \rangle &\xrightarrow{t} \langle 1, m' \rangle \\
& \quad \text{if } m'(x) = m(x) + 1 \\
& \quad m'(y) = m(y) \quad \text{for } y \neq x
\end{align*} \]

\[ \text{Denoted } m' = m \left[ x \leftarrow m(x) + 1 \right] \]

Initial states: \( I \subseteq S \)

\[ I = \{ \langle 1, m \rangle \mid \forall x \in \mathbb{N} : m(x) \leq x \} \]

Transition System

\[ \langle S, I, t \rangle \]

- Transition relation \( t \subseteq (S \times S) \)
- Initial states \( I \subseteq S \)
- States \( S \)

Also called “small-steps operational semantics”

Reflexive transitive closure

\[ t^0 = \{ \langle s, s' \rangle \mid s = s' \} \]

\[ t^{n+1} = t \circ t^n \]

\[ t^* = \bigcup_{n \geq 0} t^n \]
Reachable states
- $<S, I, t>$: transition system
- Reachable states $R$:
  
  $$ R = \{s' \in S \mid \exists s \in I : t^*(s, s') \} $$

Example
- $x < 0$: initial states (by hypothesis)
- $1$: $x \leq 10$
  while $x < 10$ do
    $2$: $x \leftarrow x + 1$
    $3$: $x \leq 11$

Reachable states:
- $R = \{<1, m> \mid m(x) \leq 10\}$
- $\cup \{<2, m> \mid m(x) < 10\}$
- $\cup \{<3, m> \mid m(x) = 10\}$

Invariant:
- $Q = \{<1, m> \mid m(x) \leq 11\}$
- $\cup \{<2, m> \mid m(x) < 10\}$
- $\cup \{<3, m> \mid 10 \leq m(x) \leq 11\}$

Invariance
- $<S, I, t>$: transition system
- $R$: reachable states of $<S, I, t>$
- Invariant:
  - Any superset of the reachable states $Q$ is invariant for $<S, I, t>$
  - $R_{<S, I, t>} \subseteq Q$

Relational Invariance
- $<S, I, t>$: transition system
- Relational invariant $Q$:
  - $Q \subseteq \mathcal{P}(S \times S)$
  - $\{<s, s'> \mid s \in I \land t^*(s, s') \} \subseteq Q$
**Fixpoints**

\[ * t^* \subseteq \bigcup_{n \geq 0} t^n \]

\[ t^* \text{ is the least fixpoint of } F(x) = t^0 \cup x \cdot g \uparrow \]

**Proof**

\[
F(t^*) \\
= t^0 \cup (\bigcup_{n \geq 0} t^n) \cdot g \uparrow \\
= t^0 \cup t_{n+1} \cdot g \uparrow \\
= t^0 \cup t_{n+1} \cdot m \uparrow \\
= t^0 \cup \bigcup_{m \geq 1} t_m \\
= t^*
\]

\[ t^* \]

**Notation**

\[ t^* = \text{eff } F \]

---

**Example of fixpoint**

\[ t^* \subseteq \bigcup_{n \geq 0} t^n \]

\[ t^* \text{ is a fixpoint of } F(x) = t^0 \cup x \cdot g \uparrow \]

**Proof**

\[
F(t^*) \\
= t^0 \cup (\bigcup_{n \geq 0} t^n) \cdot g \uparrow \\
= t^0 \cup t_{n+1} \cdot g \uparrow \\
= t^0 \cup t_{n+1} \cdot m \uparrow \\
= t^0 \cup \bigcup_{m \geq 1} t_m \\
= t^*
\]

\[ t^* \]

---

**Tarski's fixpoint theorem (I)**

If \( L (\mathbb{E}, \mathbb{I}, \mathbb{T}, \mathbb{U}, \mathbb{N}) \) is a complete lattice and \( F \in L \rightarrow L \) is \( \mathbb{E} \)-increasing then

\[ \text{obj } F = \bigcap \{ x \in L : F(x) \leq x \} \]

**Example:**

\[ \mathbb{E}(\mathbb{E}, \mathbb{I}, \mathbb{S} \times \mathbb{S}, \mathbb{U}, \mathbb{N}) \]

\[ F(x) = t^0 \cup t \cdot g \uparrow \]

\[ t^* = \text{eff } F = \bigcap \{ r : t^0 \cup t \cdot g \uparrow \leq r \} \]
**TarSKI's Fixpoint Theorem (II)**

If \( L(\mathbb{E}, \bot, \top, \sqcup, \sqcap) \) is a complete lattice and \( F : L \rightarrow L \) preserves joins \( \sqcup \) then
\[ \exists \phi F = \sqcup F^n(\bot) \]

**Example:**
\[ \mathbb{F}(x, y) (\in, \emptyset, x \times y, \cup, \cap) \]
\[ F(x) = x \cup x \times x \]
\[ \phi F = \sqcup n \cdot F^n(\bot) \]

**Notes:**

- The wrongly attributed to Kleene
- \( F \) is increasing so
  \[ \bot \leq F(\bot) \leq F^2(\bot) \leq ... \leq F^n(\bot) \leq ... \]
- It is sufficient to assume that \( F \) preserves the sub of increasing chain (Scott continuity).
- Generalizable to increasing functions by considering transfinite iteration.
**Fixpoint Induction**

\[ \text{F}_\text{fp} \in \text{P} \iff \exists I: F(I) \subseteq I \land I \subseteq \text{P} \]

**Proof**

- **Soundness** \( \Rightarrow \):
  
  \[ F(I) \subseteq I \]
  
  \[ I \subseteq \text{P} \]
  
  \[ \Rightarrow F(I) \subseteq \text{P} \]
  
  \[ \Rightarrow \text{F}_\text{fp} \in \text{P} \]

- **Completeness** \( \Rightarrow \):
  
  Assume \( I = \text{F}_\text{fp}(F) \Rightarrow F(I) \subseteq I \)
  
  \[ F(I) \subseteq I \]
  
  \[ \Rightarrow \text{F}_\text{fp} \in \text{P} \]

- **Relative Completeness**:
  
  In a logic (e.g., HFL with first order logic), \( \text{F}_\text{fp}(F) \) might not be expressible in that logic, a source of incompleteness.

**Example**

\[ t^* \in \mathbb{R} \]

\[ \Rightarrow \text{F}_\text{fp} \in \mathbb{R} \]

\[ \Rightarrow \exists I: F(I) \subseteq I \land I \subseteq \mathbb{R} \]

\[ \Rightarrow \exists I: t^* \subseteq I \land I \subseteq \mathbb{R} \]

\( t^* \) is called the “inductive argument.” (is invariant in the specific case of wronskian proofs.)
Abstract Interpretation

Abstract:
- Properties: $x \in S$ with property $P$ if $x$ is in the set of elements with this property.
- Example: Even $(x) \iff x \in \{2n | n \in \mathbb{N}\}$

Abstraction: A correspondence between properties.

$\alpha : \mathcal{P}(S) \rightarrow \mathcal{P}(\mathcal{A})$

Exemples

Reachable States

$R = \{ s' \in S | \exists s \in I : \langle s, s' \rangle \in t^* \}$

$= \alpha(t^*)$

where $\alpha(x) = \{ s' \in S | \exists s \in I : \langle s, s' \rangle \in x \}$

$\alpha \in \mathcal{P}(S \times S) \rightarrow \mathcal{P}(S)$

Galois Connection

$\langle \mathcal{P}(S), \subseteq \rangle \xleftarrow{\alpha} \langle \mathcal{P}(\mathcal{A}), \subseteq \rangle$

$\forall x \in \mathcal{P}(S), \forall \alpha \in \mathcal{P}(\mathcal{A}) :$ $\alpha(P) \subseteq Q \iff P \subseteq \alpha(P)$
Example

\[ \alpha(P) \subseteq Q \]
\[ \Rightarrow \forall \lambda' : (\exists c : <\lambda', \lambda' > \in P) \Rightarrow \lambda' \subseteq Q \]
\[ \forall \lambda' : (\exists c : <\lambda', \lambda' > \in P) \Rightarrow \lambda' \subseteq Q \]
\[ \forall : (\forall \lambda' : <\lambda', \lambda' > \in P \Rightarrow \exists c : \lambda' \subseteq Q) \]
\[ \exists : <\lambda', \lambda' > \in P \Rightarrow \forall \lambda' : \lambda' \subseteq Q \]
\[ P \subseteq \forall : (\exists c : \lambda' \subseteq Q) \]
\[ \forall : \alpha'(Q) \subseteq \forall : \lambda' \subseteq Q \]

Intuition for Galois connections

- \( \alpha(P) \) is an over-approximation of \( P \)
- \( \forall(Q) \) is the meaning of \( Q \).

- \( P \subseteq \forall(Q) \) i.e. \( P \) is over-approximated by \( Q \) with meaning \( \forall(Q) \).
- \( \alpha(P) \subseteq Q \) i.e. \( \alpha(P) \) is a more precise approximant of \( P \) than \( Q \).
- \( \alpha(P) \subseteq Q \) i.e. \( Q \) is an over-approximation of the best approximant \( \alpha(P) \) of \( P \).
- \( P \subseteq \forall(Q) \) i.e. \( P \) is over-approximated by \( Q \) with meaning \( \forall(Q) \).

Properties of G.C.

- \( \alpha \) is increasing

- \( \alpha(y) = \alpha(y) \)
  \[ \Rightarrow y \subseteq \alpha(y) \]
  (def. G.C.)

- \( \alpha(\alpha(y)) = \alpha(y) \)
  (def. G.C.)

- \( \alpha(c) \subseteq \alpha(y) \)
  (hypothese)

- \( \alpha(\alpha(y)) = \alpha(y) \)
  (def. G.C.)

- Duality principle

- \[ \langle L, E \rangle \subseteq \langle M, E \rangle \]
  \[ \Rightarrow \alpha(x) \leq y \Rightarrow x \leq \alpha(y) \]
  \[ \Rightarrow \forall(y) \Rightarrow \exists(x) \]
  \[ \Rightarrow \forall(x) \Rightarrow \exists(y) \]

C.e. If a theorem is true of \( L, E, M, \alpha, \xi, \kappa \), ... then its dual for \( L, Z, M, \alpha, \xi, \kappa \), ... is also true.

E.g. \( \forall \) is increasing.
Property of G.C.

- $\alpha$ preserves $
abla$-bds.
  - Let $UX$ be the $
abla$ of $X \cup L$.
  - Does $\psi(X) = \{ y : x \in X \}$ have a $
abla$ $\psi(x) \cup M$?
  - Yes: Thi $\alpha(UX) = \psi(x)$.

Proof:

- $\forall x \in X, x \in UX$
- $\forall x \in X, \alpha(x) \leq \alpha(UX)$ (by definition)
- $\psi(UX)$ is an upper bound of $\psi(x)$
- Let $m$ be any upper bound of $\psi(x)$
- $\forall x \in X, \alpha(x) \leq m$ (definition of upper bound)
- $\forall x \in X, \psi(x) \leq m$ (by definition of $G.C.$)
- $\psi(UX) \leq m$ (definition of upper bound)
- $\alpha(UX) \leq m$ (by definition of $G.C.$)
- $\alpha(UX)$ is the greatest upper bound of $\psi(x)$

- $\psi$ preserves $
abla$-bds (by duality)

Fixpoint abstraction theorem:

$$\begin{align*}
(L, \leq, _, V) & \frac{\text{complete lattices}}{\rightarrow} (M, \leq, V) \\
\langle L, \leq \rangle & \frac{\text{Galois connection}}{\psi} \langle M, \leq \rangle \quad \psi \; \text{preserves bds} \\
F \in L & \rightarrow L \; (\text{commutativity}) \\
F \alpha = \alpha \circ F & \\
\psi(F) = \psi(F) &
\end{align*}$$

Intuition: commutative abstracting preserve fixpoints
Numerous weaker versions.
**Proof**

- \( \alpha(F^n(z)) = \alpha(z) \) which is the uniform of \( M \) (since \( z \in F(x) \) so \( \alpha(z) \in \alpha \) for all \( x \in M \))
- \( \alpha(F^n(z)) = F^n(\alpha(z)) \) induct hypothesis
- \( \alpha(F^n(\alpha(z))) = F^n(\alpha(z)) \) commutativity
- \( \alpha(F^n(z)) = F^n(\alpha(z)) \) induct hypothesis
- \( \alpha(F^{n+1}(z)) \)
- \( \forall z : \alpha(F^n(z)) = F^n(\alpha(z)) \)
- \( \forall z : \alpha(F^n(\alpha(z))) = F^n(\alpha(z)) \)
- \( \forall x : \alpha(F^n(\alpha(z))) = F^n(\alpha(z)) \)

**Example**

\[ t^* = \text{def} \ F \text{ where } F(x) = \exists y \in \mathbb{R} : t_0 + y \geq t \]

\[ \alpha(x) = \{ y | \exists z \in I : y \leq z \} \]

\[ \alpha(F(x)) = \alpha(I \cup \exists y \in \mathbb{R} : y \leq z) \]

\[ \alpha(I \cup \exists y \in \mathbb{R} : y \leq z) \]

\[ \alpha(F(x)) = \alpha(I \cup \exists y \in \mathbb{R} : y \leq z) \]

\[ \alpha(F(I \cup \exists y \in \mathbb{R} : y \leq z)) \]

\[ \alpha(I \cup \exists y \in \mathbb{R} : y \leq z) \]

\[ \alpha(F(I \cup \exists y \in \mathbb{R} : y \leq z)) \]

\[ \alpha(F(I \cup \exists y \in \mathbb{R} : y \leq z)) \]

**Parallel Programs**

\[ \text{States : } S = \mathcal{C}_0 \times \ldots \times \mathcal{C}_n \times \mathcal{M} \]

\[ \text{Control of processes : } \mathcal{C}_i \subseteq \mathcal{C}_i \times \mathcal{M} \]

\[ \text{Variables in shared memory : } \mathcal{M} \]

**Transition relation of processes**

\[ t \in \mathcal{S} \times S \]

\[ t^i \subseteq \mathcal{S}_i \times \mathcal{S} \]

**Transition relation of the parallel program**

\[ t \in \mathcal{S} \times S \]

\[ t^i \subseteq \mathcal{S}_i \times \mathcal{S}_i \]

\[ t^i \subseteq \mathcal{S}_i \times \mathcal{S}_i \]

\[ t^i \subseteq \mathcal{S}_i \times \mathcal{S}_i \]

\[ t = \bigcup_{i=1}^{n} t^i \]
**Principle of the design**

- This is the basic induction principle.
- Applying further fixpoint preserving abstractions we get:
  - Numerous variants of the inductive principle.
    - \( \alpha(p) = \neg p \) proofs by reduction ad absurdum.
    - \( \alpha(t) = t^{-1} \) backward proof method (e.g., subgoal induction, wp, etc.).
- Language specific invariance proof methods.

---

**Example: Eng, Naur, Floyd**

\[ S = C \times M \quad \text{control state} \]
\[ m \in M \quad \text{memory state} \]
\[ \alpha(c) = \Pi_{c \in C} \{ m \mid <c, m> \in P \} \]

i.e., projection on the program control points to get local invariants on variables attached to program points.

---

**Application to parallel processes (with sequential consistency).**
The Ascroft - Manna method

- Apply the further abstraction (which is also an isomorphism)

\[ \alpha_{AM}(P) = \prod_{c_i \in C_1, c_2 \in C_2, \ldots, c_n \in C_n} \{ m \mid \langle c_1 \ldots c_m \rangle \in P \} \]

Ascroft - Manna verification conditions

- obtained by the commutation condition of the fixed point abstraction theorem

\[ \forall c_i \in C_1, c_2 \in C_2, \ldots, c_n \in C_n : \forall m \in M : \forall i \in [1, n] : \]

- \[ \langle c_1 \ldots c_i-1 c_i+1 \ldots c_m \rangle \in P_{c_1 \ldots c_i \ldots c_n} \]

- \[ \langle c_i, m \rangle \rightarrow \langle c'_i, m \rangle \]

- \[ \Rightarrow \langle c_1 \ldots c_i-1 c'_i c_{i+1} \ldots c_m \rangle \in P_{c_1 \ldots c_i \ldots c_n} \]

- Too many invariants |c_1|x|c_2|x|...|c_n|

The Lamport method

- Apply the further isomorph ic abstraction:

\[ \alpha_L(P) = \prod_{c_i \in C_1, c_2 \in C_2, \ldots, c_n \in C_n} \{ m \mid \langle c_1 \ldots c_i \rangle \in P \} \]

To each process \( P_i \):
- to each program point of that process:
- attach an invariant
- on the control points of the other processes:
- on the shared memory states \( m \)

Lamport’s verification conditions

- obtained by the commutation condition of the fixed point abstraction for \( \alpha_L \)

\[ \forall c_i \in C_1, c_2 \in C_2, \ldots, c_n \in C_n : \forall m \in M : \forall i \in [1, n] \]

- \[ \langle c_1 \ldots c_i-1 c_i+1 \ldots c_m \rangle \in P_{c_1 \ldots c_i \ldots c_n} \]

- \[ \forall c \in [1, n] \]

- \[ \langle c_i, m \rangle \rightarrow \langle c'_i, m \rangle \]

- \[ \Rightarrow \langle c_1 \ldots c_i-1 c'_i c_{i+1} \ldots c_m \rangle \in P_{c_1 \ldots c_i \ldots c_n} \]

- \[ \forall j \in [1, n] \]

- \[ \langle c_j, m \rangle \rightarrow \langle c'_j, m \rangle \]

- \[ \Rightarrow \langle c_1 \ldots c_{j-1} c'_j c_{j+1} \ldots c_m \rangle \in P_{c_1 \ldots c_j \ldots c_n} \]

- Note: The precondition can be strengthened e.g.

\[ \langle c_1 \ldots c_i a_{c_{i+1}} \ldots c_j c_{j+1} \ldots c_m \rangle \in P_{c_1 \ldots c_j \ldots c_n} \]
Example

\[ \begin{align*}
\{ x = 0 \} & : c_2 = 3 \land x = 0 \\
& \quad \forall c_2 \geq 3 \land x = 0 \\
& \quad \forall c_2 \geq 4 \land x = 0 \\
2 : & c_2 = 3 \land x = 1 \\
& \quad \forall c_2 \geq 4 \land x = 1 \\
& \quad \forall c_2 \geq 5 \land x = 1 \\
\{ x = 2 \} - \\
\end{align*} \]

- Initialisation: \( \{ x = 0 \} \land c_2 \geq 3 \implies P_1 \)
- Soundness proof
- Axioms of termination: \( P_1 \)
- Termination: \( c_2 = 1 \land P_2 \land c_2 = 4 \land P_1 \implies x = 2 \).

The Owuchi & Gries abstraction

\[ \chi_{\text{OG}} (P) = \prod_{i=1}^{n} \prod_{c_i \in C_i} \{ m | \exists c_i \ldots c_{i-1} c_{i+1} \ldots c_m : c_i \ldots c_{i-1} c_{i+1} \ldots c_m \implies \}
\]

- to each process \( P_i \)
- to each program pair \( c_i \) of process
- attach an invariant on
- The shared memory state
- i.e. same as Floyd for \( n = 1 \) but incomplete for \( n > 1 \).

Proof of incompleteness

- To make the proof, we need an invariant.
- The strongest one is the left of the verification result.
- Here is an example of strongest invariant:

\[ \begin{align*}
\{ x = 0 \} & : (x \geq 0) \\
1 : & (x \geq 0) \\
& \quad x = x + 1 \\
2 : & (x \geq 1) \\
& \quad x = x + 1 \\
\{ x \geq 2 \}
\end{align*} \]

\( \Rightarrow \) Impossible to prove that \( x = 2 \) on exit.

Auxiliary variables

- Add auxiliary variables to the program, prove the modified program, this implies the correctness of the original program.

- Example

\[ \begin{align*}
1 : & c_3 = 1 \\
& \quad x = x + 1 \\
2 : & c_4 = 2 \\
& \quad x = x + 1 \\
3 : & c_5 = 3 \\
& \quad x = x + 1 \\
4 : & c_6 = 4 \\
& \quad x = x + 1
\end{align*} \]

- Owuchi & Gries provide no clue on how to discover auxiliary variables.
The completeness proof

- Choose auxiliary variables that simulate the program counters
- Show that the abstraction eliminating these auxiliary counters provides the semantics of the original method
- Conclude by completeness of Langard's method

See details in:

R. Conuat. Reasoning about program invariants proof methods. Res. rep. CRIS-86-P08A

WHAT ABOUT JONES' REPLY / GUARANTEE?

Reachable states

\[ R = \text{Eff} F \]

\[ F(X) = \bigcup \{ A' \mid \exists a : A \xrightarrow{a} A' \} = \bigcup \{ A' \mid \exists a : A \xrightarrow{\text{Eff}} A' \} = \bigcup \left( \bigcup \{ A' \mid \exists a : A \xrightarrow{\text{Eff}} A' \} \right) \]

\[ = R(G)X \]

where \[ G(X) = \bigcup \{ A' \mid \exists a : A \xrightarrow{\text{Eff}} A' \} \leftarrow \text{guarantee} \]

\[ R(G)X = \bigcup \{ A' \mid \exists a : A \xrightarrow{\text{Eff}} A' \} \leftarrow \text{rely (assuming guarantee)} \]

\[ \text{Theorem} \]

\[ \text{Eff} F = \text{Eff} \forall x. R(G(x))x \]

\[ \text{proof} \]

The least fixpoint of \[ X = F(X) \]

is the same as the least fixpoint of the right side of equation

\[ X = \text{Eff}(Y)X \]

\[ Y = G(X) \]

by the theorem of asynchronous iteration with memory (Conuat & Conuat, 1977)
JONES Rely/Guarantee
- Apply the Hoare post condition principle to
  \[ \text{efp } \langle X, Y \rangle < R(Y) X, G(X) \]
- up to the Lamport abstraction GL, assigning
  to each control point an assertion on
  - the shared variable
  - the control point of the other process
  (as Ouakli & Gnesi with auxiliary variables)

- Cliff B. Jones:
  Tentative Steps Toward a Development Method for
  596-619 (1983)

- Joey W. Coleman, Cliff B. Jones:
  A Structural Proof of the Soundness of Rely/guarantee

Astriée A
- Astriée: a static analyser of C for
  synchronous control-command
  embedded software

- Astriée A: idem, for parallel programs

⇒ a further abstraction of
- an invariant at each point of each
  process on the shared variables and
  program counter of other processes
- rely-guarantee fixpoint computation
- widening/narrowing convergence
  acceleration

APPLICATIONS

CONCLUSION
Conclusion

- Too many computer scientists are tinker[wo]men (bricolleurs/sectors).
- If you want to understand what you do go to basic principles.
- For reasoning on program semantics this is A.I. =)

PS: this approach generalizes to termination (Courc & Cousot, POPL 2012)