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

Patrick Cousot

NYU, NYC, NY

pcousot@cims.nyu.edu

History

Turing (1949) invents invariance + termination proofs for sequential programs.

Naur (1966) re-invents invariance proofs

Floyd (1967) re-invents invariance + termination proofs

Hoare (1969) invents structural induction (in HL)

... thousands of (forgotten) publications

Owicki [and Gries] (1976) generalize HL to parallel processes with sequential consistency (SC) (incomplete without auxiliary variables)

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

History (cont'd)

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

--- thousands of (forgotten) publications

TODAY : researchers reinvent everything for weak memory models (WMM)

→ based on Owicki & Gries (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.

DEFINITION OF INVARIANCE BASED ON AN OPERATIIONAL SEMANTICS

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** : $t \in \wp(S \times S)$

$$\langle c, m \rangle \xrightarrow{t} \langle c', m' \rangle$$ i.e. $\langle\langle c,m\rangle, \langle c', m'\rangle\rangle \in t$

iff 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
    od ;
3 :
```

$$\langle 1, m\rangle \xrightarrow{t} \langle 2, m\rangle \quad \text{if} \quad m(x) < 10$$

$$\langle 1, m\rangle \xrightarrow{t} \langle 3, m\rangle \quad \text{if} \quad m(x) \geqslant 10$$

$$\langle 2, m\rangle \xrightarrow{t} \langle 1, m'\rangle$$

if $m'(x) = m(x) + 1$

$m'(y) = m(y)$ for $y \neq x$

denoted $m' = m[x \leftarrow m(x) + 1]$

Initial states : $I \subseteq S$

$$I = \{\langle 1, m\rangle \mid \forall x \in \mathbb{X}\,.\, m(x) \in \mathbb{Z}\}$$

Transition system

$\langle S, I, t \rangle$

- transition relation $t \in \wp(S \times S)$
- initial states $I \subseteq S$
- states S

Also called "small-steps operational semantics"

Reflexive transitive closure

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

$$t^{n+1} = t \mathbin{;} t^n$$

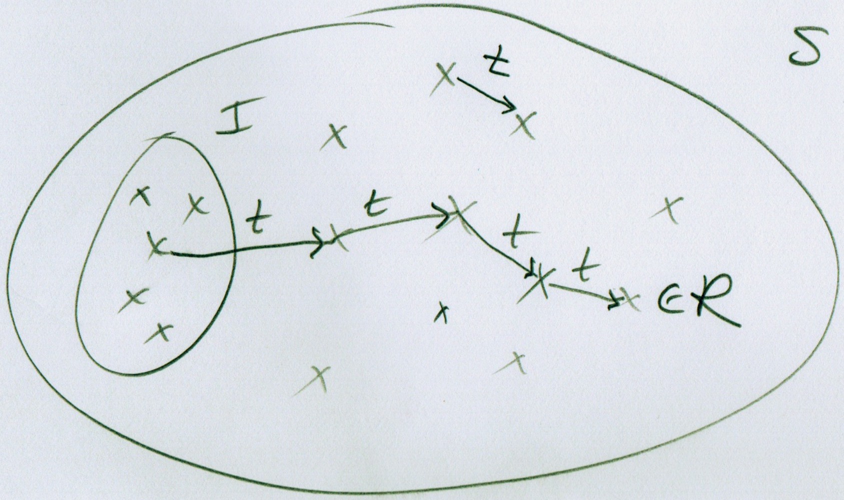
$$= \{\langle s, s''\rangle \mid \exists s' \in S : \langle s, s'\rangle \in t \wedge \langle s', s''\rangle \in t^n\}$$

$$t^* \triangleq \bigcup_{n \geq 0} t^n$$

Reachable states

- $\langle S, I, t \rangle$: transition system
- Reachable states R :

$$R = \{ s' \in S \mid \exists s \in I : t^*(s, s') \}$$



Invariance

- $\langle S, I, t \rangle$: transition system
- R : reachable states of $\langle S, I, t \rangle$
- Invariant :
  - Any superset of the reachable states

Q is invariant for $\langle S, I, t \rangle$

$$\triangleq \quad R_{\langle S, I, t \rangle} \subseteq Q$$

Example

$\{x \leq 0\}$ ← Initial states (by hypothesis)

1 : $\{x \leq 10\}$

while $x < 10$ do

  2 : $\{x < 10\}$

    $x := x + 1$

od

3 : $\{10 \leq x \leq 11\}$

Reachable states :

$$R = \{\langle 1, m\rangle \mid m(x) \leq 10\} \cup \{\langle 2, m\rangle \mid m(x) < 10\} \cup \{\langle 3, m\rangle \mid m(x) = 10\}$$

Invariant :

$$Q = \{\langle 1, m\rangle \mid m(x) \leq 11\} \cup \{\langle 2, m\rangle \mid m(x) < 10\} \cup \{\langle 3, m\rangle \mid 10 \leq m(x) \leq 11\}$$

Relational Invariante

- $\langle S, I, t \rangle$: transition system
- Relational invariant Q :
  - $Q \in \wp(S \times S)$
  - $\{\langle s, s' \rangle \mid s \in I \wedge t^*(s, s')\} \subseteq Q$

FixPoints

Example of fixpoint

- $t^* \triangleq \bigcup_{n\geqslant 0} t^n$
- t^* is a fixpoint of $F(x) = t^0 \cup x \mathbin{;} t$

Proof

$$
\begin{aligned}
& F(t^*) \\
&= t^0 \cup \left(\bigcup_{n\geqslant 0} t^n\right) \mathbin{;} t \\
&= t^0 \cup \bigcup_{n\geqslant 0} (t^n \mathbin{;} t) \\
&= t^0 \cup \bigcup_{n\geqslant 0} t^{n+1} \\
&= t^0 \cup \bigcup_{m\geqslant 1} t^m \qquad (m = n+1) \\
&= \bigcup_{n\geqslant 0} t^n \\
&= t^*
\end{aligned}
$$

□

- t^* is the **least** fixpoint of $F(x) = t^0 \cup x \mathbin{;} t$

Proof Assume $r = F(r)$ is a fixpoint of F

- $t^0 \subseteq r$
- $t^n \subseteq r$ induction hypothesis
- t^{n+1}
  $= t^n \mathbin{;} t$
  $\subseteq r \mathbin{;} t$ (ind. hyp.)
  $\subseteq t^0 \cup r \mathbin{;} t$
  $= F(r) = r$
- $\forall n : t^n \subseteq r$ (by recurrence)

$\Rightarrow t^* = \bigcup_{n \geq 0} t^n \subseteq r$ (def. least upper bound $\cup$)

$\square$

Notation $t^* = \mathsf{lfp}\ F$

least fixed point

Tarski's fixpoint theorem (I)

If $L(\sqsubseteq, \bot, \top, \sqcup, \sqcap)$ is a complete lattice and $F \in L \to L$ is $\sqsubseteq$-increasing then

$$\mathrm{lfp}\, F = \sqcap \{x \in L : F(x) \sqsubseteq x\}.$$

Example : $\wp(S \times S)(\subseteq, \emptyset, S \times S, \cup, \cap)$

$F(x) = t^0 \cup t \circ x$

$t^* = \mathrm{lfp}\, F = \cap \{r : t^0 \cup r \circ t \subseteq r\}$

Proof

- $P \triangleq \{x \in L : F(x) \sqsubseteq x\}$ ($P \neq \emptyset$ since $\top \in P$)
  $a \triangleq \sqcap P$ (greatest lower bound, glb)
- $\forall x \in P$:

$a = \sqcap P \sqsubseteq x$ (def. glb)
$\Rightarrow F(a) \sqsubseteq F(x)$ (F increasing)
$\Rightarrow F(a) \sqsubseteq x$ ($x \in P$ so $F(x) \sqsubseteq x$)
$\Rightarrow F(a)$ is a lower bond of P
$\Rightarrow F(a) \sqsubseteq a$ (a is the glb of P)
$\Rightarrow F(F(a)) \sqsubseteq F(a)$ (F increasing)
$\Rightarrow F(a) \in P$ (def. P)
$\Rightarrow a \sqsubseteq F(a)$ (a is the glb of P)
$\Rightarrow a = F(a)$ (antisymmetry of $\sqsubseteq$)

- If x is any fixpoint of F (which has at least one : a)

$F(x) = x$ (def. fixpoint)
$\Rightarrow F(x) \sqsubseteq x$ ($\sqsubseteq$ is reflexive)
$\Rightarrow x \in P$ (def. P)
$\Rightarrow a \sqsubseteq x$ (a is the glb of P)

- $a = \mathrm{lfp}\,(F)$

$\square$

Tarski's fixpoint theorem (II)

If $L(\sqsubseteq, \bot, \top, \sqcup, \sqcap)$ is a complete lattice and $F \in L \longrightarrow L$ preserves joins $\sqcup$ then

$$\mathrm{lfp}\, F = \bigsqcup_{n \geqslant 0} F^n(\bot)$$

Example : $\mathcal{P}(S \times S)\ (\subseteq, \emptyset, S \times S, \cup, \cap)$

$$F(x) = t^0 \cup x \circ t$$

$$t^* = \mathrm{lfp}\, F = \bigcup_{n \geqslant 0} t^n$$

– $F^0(x) = x$ iterates of F

$F^{n+1}(x) = F(F^n(x))$

– $F(\bigsqcup_{i \in \Delta} x_i) = \bigsqcup_{i \in \Delta} F(x_i)$ join preservation

$F(\sqcup X) = \sqcup \{F(x) \mid x \in X\}$

Proof.

- $a \triangleq \bigsqcup_{n \geq 0} F^n(\bot)$

- $F(a)$

$= F(\bigsqcup_{n \geq 0} F^n \bot)$ (def. a)

$= \bigsqcup_{n \geq 0} F(F^n(\bot))$ (F preserves join $\sqcup$)

$= \bigsqcup_{n \geq 0} F^{n+1}(\bot)$ (def iterates)

$= \bot \sqcup \bigsqcup_{n \geq 1} F^n(\bot)$ ($\bot$ is the infimum

$= \bigsqcup_{n \geq 0} F^n(\bot)$ (def iterates $F^0(\bot) = \bot$)

$= a$ (def. a)

- If x is any fixpoint of F
  - $F^0(\bot) = \bot \sqsubseteq x$ (def. infimum $\bot$)
  - $F^n(\bot) \sqsubseteq x$ (induction hypothesis)
  - $F^{n+1}(\bot) = F(F^n(\bot)) \sqsubseteq F(x) = x$ (F preserves joins hence increasing)
  - $\forall n : F^n(\bot) \sqsubseteq x$ (by recurrence)
  - $a = \bigsqcup_{n \geq 0} F^n(\bot) \sqsubseteq x$ (def a and $\sqcup$ is the glb)

- $a = \mathrm{lfp}\, F$ (def. lfp). ∎

Notes

- Th. wrongly attributed to Kleene
- F is increasing so
  $$\perp \sqsubseteq F(\perp) \sqsubseteq F^2(\perp) \sqsubseteq \ldots \sqsubseteq F^n(\perp) \sqsubseteq \ldots$$
- It is sufficient to assume that F preserves the lub of increasing chain (Scott continuity).
- Generalizable to increasing functions by considering transfinite iterates.

FIXPOINT INDUCTION

FIXPOINT OVER-APPROXIMATION

Prove that $\mathrm{lfp}\, F \sqsubseteq P$

(under the hypothesis of Tarski's fixpoint theorem)

FIXPOINT INDUCTION

$$\mathrm{lfp}\, F \sqsubseteq P \iff \exists I : F(I) \sqsubseteq I \wedge I \sqsubseteq P$$

<u>Proof</u>

Soundness $\Leftarrow$: $F(I) \sqsubseteq I$

$\Rightarrow I \in \{X \mid F(X) \sqsubseteq X\}$

$\Rightarrow \mathrm{lfp}\, F = \sqcap \{X \mid F(X) \sqsubseteq X\} \sqsubseteq I$

(Tarski & def. glb $\sqcap$)

$\Rightarrow \mathrm{lfp}\, F \sqsubseteq I$ ($I \sqsubseteq P$ and transitivity)

Completeness $\Rightarrow$: choose $I = \mathrm{lfp}(F)$ so $F(I) = I$ implies $F(I) \sqsubseteq I$ by reflexivity and $I \sqsubseteq P$ by hypothesis

Relative completeness : in a logic (e.g. HL with first order logic), $\mathrm{lfp}(F)$ might not be expressible in that logic, a source of incompleteness.

Example

$$t^* \subseteq r$$

$\Longleftrightarrow \mathsf{lfp}\ F \subseteq r$ where $F(x) = t^0 \cup x \mathbin{;} t$

$\Longleftrightarrow \exists I : F(I) \subseteq I \wedge I \subseteq r$

$\Longleftrightarrow \exists I : t^0 \subseteq I \wedge I \mathbin{;} t \subseteq I \wedge I \subseteq r$

I is called the "inductive argument" (or invariant in the specific case of invariance proofs)

ABSTRACT INTERPRETATION

ABSTRACTION

Properties : $x \in S$ a la propriété P

$\iff$ x appartient à l'ensemble des éléments qui ont cette propriété

$\iff$ une propriété est un élément de $\mathcal{P}(S)$

Exemple : $\text{even}(x) \iff x \in \{2n \mid n \in \mathbb{N}\}$

Abstraction : A correspondance between properties.

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

abstraction — propriétés concrètes — propriétés abstraites

Exemples

Reachable states

$$\mathcal{R} = \{ s' \in S \mid \exists s \in \mathcal{I} : \langle s, s' \rangle \in t^* \}$$
$$= \alpha(t^*)$$

where $\alpha(X) = \{ s' \in S \mid \exists s \in \mathcal{I} : \langle s, s' \rangle \in X \}$

$$\alpha \in \wp(S \times S) \longrightarrow \wp(S)$$

relation = properties of pairr of stater

properties of states

Galois connection

$$\langle \wp(S), \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle \wp(S'), \subseteq \rangle$$

$$\triangleq$$

$\forall P \in \wp(S) : \forall Q \in \wp(S') :$

$$\alpha(P) \subseteq Q \iff P \subseteq \gamma(Q)$$

Example

$$\alpha(P) \subseteq Q$$

$$\Longleftrightarrow \{s' \in S \mid \exists s \in I : \langle s, s' \rangle \in P\} \subseteq Q \qquad \{\text{def } \alpha\}$$

$$\Longleftrightarrow \forall s' : (\exists s \in I : \langle s, s' \rangle \in P) \Rightarrow s' \subseteq Q$$

$$\Longleftrightarrow \forall s' : \forall s \in I : \langle s, s' \rangle \in P \Rightarrow s' \in Q$$

$$\Longleftrightarrow \forall s : \forall s' : \langle s, s' \rangle \in P \Rightarrow (s \in I \Rightarrow s' \in Q)$$

$$\Longleftrightarrow \forall s : \forall s' \; \langle s, s' \rangle \in P \Rightarrow \langle s, s' \rangle \in \{\langle s, s' \rangle \mid s \in I \Rightarrow s' \in Q\}$$

$$\Longleftrightarrow P \subseteq \underbrace{\{\langle s, s' \rangle \mid s \in I \Rightarrow s' \in Q\}}_{\gamma(Q)}$$

$$\Longleftrightarrow P \subseteq \gamma(Q)$$

Intuition for Galois connections

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

- $P \subseteq \gamma(Q)$ i.e. P is over-approximated by Q with meaning $\gamma(Q)$

$\Rightarrow \alpha(P) \subseteq Q$ i.e. $\alpha(P)$ is a more precise approximation of P than Q

- $\alpha(P) \subseteq Q$ i.e. Q is an over-approximation of the best approximation $\alpha(P)$ of P

$\Rightarrow P \subseteq \gamma(Q)$ i.e so P is over-approximated by Q with meaning $\gamma(Q)$.

Properties of G.C.

– α is increasing

– $\alpha(y) \sqsubseteq \alpha(y)$ (reflexivity)

$\Rightarrow y \sqsubseteq \gamma \circ \alpha(y)$ (def. G.C)

– $x \sqsubseteq y$ (hypothesis)

$\Rightarrow x \sqsubseteq \gamma \circ \alpha(y)$ ($x \sqsubseteq y \sqsubseteq \gamma \circ \alpha(y)$ and transitivity)

$\Rightarrow \alpha(x) \sqsubseteq \alpha(y)$ (def. G.C)

Properties of G.C.

- Duality principle

$$
\begin{aligned}
& \langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \leqslant \rangle \\
\Longleftrightarrow\ & \alpha(x) \leqslant y \iff x \sqsubseteq \gamma(y) \\
\Longleftrightarrow\ & \gamma(y) \sqsupseteq x \iff y \geqslant \alpha(x) \\
\Longleftrightarrow\ & \gamma(x) \sqsupseteq y \iff x \geqslant \alpha(x) \\
\Longleftrightarrow\ & \langle M, \geqslant \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle L, \sqsupseteq \rangle
\end{aligned}
$$

i.e. if a theorem is true of $L, \sqsubseteq, M, \leqslant, \alpha, \gamma, \ldots$ then its dual for $L, \sqsupseteq, M, \geqslant, \gamma, \alpha, \ldots$ is also true

e.g. γ is increasing.

Properties of G.C.

- α preserves lubs.
  - Let $\sqcup X$ be the lub of X in L
  - does $\alpha(X) \triangleq \{\alpha(x) \mid x \in X\}$ has a lub $\vee \alpha(X)$ in M?
  - yes this is $\alpha(\sqcup X) = \vee\, \alpha(X)$.

proof

- $\forall x \in X : x \sqsubseteq \sqcup X$

  $\Rightarrow \forall x \in X : \alpha(x) \leqslant \alpha(\sqcup X)$ (α increasing)

  $\Rightarrow \alpha(\sqcup X)$ is an upper bound of $\alpha(X)$
- let m be any upper bound of $\alpha(X)$

  $\forall x \in X : \alpha(x) \leqslant m$ (def. upper bound)

  $\Rightarrow \forall x \in X \quad x \sqsubseteq \gamma(m)$ (def. G.C.)

  $\Rightarrow \quad \sqcup X \sqsubseteq \gamma(m)$ (def lub $\sqcup$)

  $\Rightarrow \alpha(\sqcup X) \sqsubseteq m$ (G.C)

  $\Rightarrow \alpha(\sqcup X)$ is the <u>least</u> upper bound of $\alpha(X)$

$\square$

- γ preserves glbs (by duality)

Properties of G.C.

– one adjoint uniquely determine the other

proof

$$\alpha(x) = \sqcap\{y : \alpha(x) \sqsubseteq y\}$$

$$= \sqcap\{y : y \sqsubseteq \gamma(y)\}$$

by duality

$$\gamma(x) = \sqcup\{y : \gamma(x) \geq y\}$$

$$= \sqcup\{y : y \leq \gamma(x)\}$$

$\square$

FIXPOINT ABSTRACTION

Fixpoint abstraction theorem

$$\left[\begin{array}{ll}
\left.\begin{array}{l}(L, \sqsubseteq, \bot, \sqcup) \\ (M, \leq, \vee)\end{array}\right\} & \text{complete lattices} \\
\langle L, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \leq \rangle & \text{Galois connection} \\
\left.\begin{array}{l}F \in L \to L \\ \overline{F} \in M \to L\end{array}\right\} & \text{preserve lubs} \\
\overline{F} \circ \alpha = \alpha \circ F & \text{(commutativity)} \\
\Longrightarrow \quad \alpha(\mathrm{lfp}\ F) = \mathrm{lfp}\ \overline{F} &
\end{array}\right.$$

Intuition : commutative abstractions preserve fixpoints

Numerous weaker versions.

Proof

- $\alpha(F^0(\bot)) = \alpha(\bot)$ which is the infimum of M (since $\bot \sqsubseteq \gamma(x)$ so $\alpha(\bot) \sqsubseteq x$, for all $x \in M$)

- $\alpha(F^n(\bot)) = \overline{F}^n(\alpha(\bot))$ induction hypothesis

$\Rightarrow \alpha(F^{n+1}(\bot))$

$= \alpha(F(F^n(\bot)))$

$= \overline{F}(\alpha(F^n(\bot)))$ commutation

$= \overline{F}(\overline{F}^n(\alpha(\bot)))$ induction hypothesis

$= \overline{F}^{n+1}(\alpha(\bot))$

- $\forall n: \alpha(F^n(\bot)) = \overline{F}^n(\alpha(\bot))$

$\Rightarrow \bigsqcup \alpha(F^n(\bot)) = \bigsqcup \overline{F}^n(\alpha(\bot))$ def. lub

$\Rightarrow \alpha(\bigsqcup F^n(\bot)) = \bigsqcup \overline{F}^n(\alpha(\bot))$ α preserves joins

$\Rightarrow \alpha(\mathrm{lfp}\, F) = \mathrm{lfp}\, \overline{F}$ Tarski II

$\square$

Example

$t^* = \mathrm{lfp}\, F$ where $F(x) = t^0 \cup x \mathbin{;} t$

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

$\alpha(F(x))$

$= \alpha(t^0 \cup x \mathbin{;} t)$ (def. F)

$= \alpha(t^0) \cup \alpha(x \mathbin{;} t)$ (α preserves joins $\cup$)

$= \alpha(\{ s' \mid \exists s \in I : s = s' \}) \cup \alpha(x \mathbin{;} t)$

$= I \cup \{ s' \mid \exists s \in I : \exists s'' : \langle s, s'' \rangle \in x \wedge \langle s'', s' \rangle \in t \}$

$= I \cup \{ s' \mid \exists s'' \in \{ s'' : \exists s \in I : \langle s, s'' \rangle \in x \} : \langle s'', s' \rangle \in t \}$

$= I \cup \{ s' \mid \exists s'' \in \alpha(x) : \langle s'', s' \rangle \in t \}$

Eureka!

$= \overline{F}(\alpha(x))$

where $\overline{F}(X) = I \cup \{ s' \mid \exists s'' \in X : \langle s'', s' \rangle \in t \}$

and so $\alpha(t^*) = \alpha(\mathrm{lfp}\, F) = \mathrm{lfp}\, \overline{F}$

i.e. calculational design of the verification condition $\overline{F}(X) \subseteq X$

DESIGN OF AN INVARIANCE PROOF METHOD

Parallel Programs

$[\![P_1 \parallel \ldots \parallel P_n]\!]$

States : $S = C_1 \times \ldots \times C_n \times M$

control points of the processes

state of the variables in the shared memory

Transition relation of processes

$S^i \in C_i \times M$ $\quad I^i \subseteq S^i$ initial states

$t^i \in \wp(S^i \times S^i)$

Transition relation of the parallel program

$t \in \wp(S \times S)$

$\vec{t}^i = \{\langle\langle c_1 \ldots c_i \ldots c_n\, m\rangle \langle c_1 \ldots c'_i \ldots c_n\, m'\rangle\rangle \mid t^i(\langle c_i, m\rangle, \langle c'_i, m'\rangle\}$

$t = \bigvee_{i=1}^{n} \vec{t}_i$

Principle of the design

$$R_{\langle S, I, t\rangle} \subseteq Q$$ invariance

$$\iff \alpha_I(t^*) \subseteq Q$$

$$\iff \alpha_I(\mathrm{lfp}\, F) \subseteq Q$$

$$\iff \mathrm{lfp}\, \overline{F} \subseteq Q$$ fixpoint abstraction

$$\iff \exists P : \overline{F}(P) \subseteq P \wedge P \subseteq Q$$ fixpoint induction

$$\iff \exists P : I \cup \{s \mid \exists s' \in P : \langle s', s\rangle \in t\} \subseteq P \wedge P \subseteq Q$$

$$\iff \exists P : I \subseteq P \;\wedge\; \forall s : \forall s' \in P : s' \xrightarrow{t} s \Rightarrow s \in P \;\wedge\; P \subseteq Q$$

Find an <u>inductive invariant</u> P

The inductive invariant is true for all initial states in I

Assuming the invariant true ($s' \in P$) prove that it remains true ($s \in P$) after a program step ($s' \xrightarrow{t} s$) i.e the invariant is <u>inductive</u>

The inductive invariant

Principle of the design

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

(1) Patrick Cousot & Radhia Cousot. Induction principles for proving invariance properties of programs. In D. Néel, editor, *Tools & Notions for Program Construction: an Advanced Course*, pages 75—119. Cambridge University Press, Cambridge, UK, August 1982.

Example : Turing / Naur / Floyd

$S = C \times M$ $\quad$ $c \in C$ control state

$\quad\quad\quad\quad\quad\quad$ $m \in M$ memory state

$$\alpha(P) = \prod_{c \in C} \{m \mid \langle c, m \rangle \in P\}$$

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)

$$\alpha_{AM}(P) = \prod_{c_1 \in C_1,\, c_2 \in C_2,\, \ldots,\, c_n \in C_n} \{ m \mid \langle c_1 \ldots c_n\, m \rangle \in P \}.$$

Ascroft - Manna verification conditions

- obtained by the commutation condition of the fixpoint abstract theorem.

- $\forall c_1 \in C_1 : \forall c_2 \in C_2 ; \ldots : \forall c_n \in C_n : \forall m \in M :$
  $\forall i \in [1, n] :$

$$\langle c_1 \ldots c_{i-1}\, c_i\, c_{i+1} \ldots c_n\ m \rangle \in P_{c_1 \ldots c_i \ldots c_n}$$

$$\wedge\ \langle c_i, m \rangle \xrightarrow{t} \langle c_i', m' \rangle$$

$$\Longrightarrow \langle c_1 \ldots c_{i-1}\, c_i'\, c_{i+1} \ldots c_n\ m' \rangle \in P_{c_1 \ldots c_i' \ldots c_n}$$

- Too many invariants $|C_1| \times |C_2| \times \ldots \times |C_n|$

The Lamport method.

- Apply the further isomorphic abstraction:

$$\alpha_L(P) = \prod_{i=1}^{n} \prod_{c_i \in C_i} \{ \langle c_1 \ldots c_{i-1}\; c_{i+1} \ldots c_n\; m \rangle \mid \langle c_1 \ldots c_{i-1}\; c_i\; c_{i+1} \ldots c_n\; m \rangle \in P \}$$

To each process P_i

to each program point of that process

attach an invariant
- on the control points of the <u>other</u> processes
- on the shared memory state m

WOT

Lamport's verification conditions

- obtained by the commutation condition of the fixpoint abstraction for α_L

- $\forall i \in [1, n]:$

  $\forall c_i \in C_i:$

$$\left.\begin{array}{l} \langle c_1 \ldots c_{i-1}\ c_{i+1} \ldots c_n\ m \rangle \in P_{c_i} \\ \wedge\ t_i(\langle c_i, m\rangle, \langle c_i', m'\rangle) \\ \Longrightarrow \langle c_1\ c_{i-1}\ c_{i+1} \ldots c_n\ m \rangle \in P_{c_i'} \end{array}\right\} \text{ sequential proof}$$

$\wedge\ \forall j \in [1, n] \setminus \{i\}$

$$\left.\begin{array}{l} \langle c_1 \ldots c_{i-1}\ c_{i+1} \ldots c_j \ldots c_n\ m \rangle \in P_{c_i} \\ \wedge\ t_j(\langle c_j, m\rangle, \langle c_j', m'\rangle \\ \Longrightarrow \langle c_1 \ldots c_{i-1}\ c_{i+1} \ldots c_j', m' \rangle \in P_{c_i} \end{array}\right\} \text{ proof of absence of interference}$$

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

$$\langle c_1 \ldots c_{i-1}\ c_i\ c_{i+1} \ldots c_{j-1}\ c_j \ldots c_n\ m \rangle \in P_{c_j}$$

Example

$\{x=0\}$ i.e $\{m \mid m(x)=0\}$!

$$
\left[\begin{array}{l}
1: \; c_2=3 \land x=0 \\
\quad \lor\, c_2=4 \land x=1 \\
\quad x := x+1 \\
2: \; c_2=3 \land x=1 \\
\quad \lor\, c_2=4 \land x=2
\end{array}\right]
\;\Big\|\;
\left[\begin{array}{l}
3: \; c_1=1 \land x=0 \\
\quad \lor\, c_1=2 \land x=1 \\
\quad x := x+1 \\
4: \; c_1=1 \land x=1 \\
\quad \lor\, c_1=2 \land x=2
\end{array}\right]
$$

$\{x=2\}$

Initialisation : $\{x=0\} \land c_1=1 \land c_2=3 \Longrightarrow P_1$
$\Longrightarrow P_3$

sequential proof

Absence of interference proof.

Finalisation : $c_1=1 \land P2 \land c_2=4 \land P_4 \Longrightarrow x=2$.

The Owicki & Gries abstraction

$$\alpha_{OG}(P) = \prod_{i=1}^{n} \prod_{c_i \in C_i} \{ m \mid \exists\, c_1 \ldots c_{i-1}\, c_{i+1} \ldots c_n : \langle c_1 \ldots c_{i-1}\, c_i\, c_{i+1} \ldots c_n\, m \rangle \in P \}$$

to each process P_i

to each program point 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 lfp of the verification condition
- Here is an example of strongest invariant:

$\{x=0\}$

$$\left[\begin{array}{l} 1: \{x \geq 0\} \\ \quad x := x+1 \\ 2: (x \geq 1) \end{array} \;\middle\|\; \begin{array}{l} 3: (x \geq 0) \\ \quad x := x+1 \\ 4: (x \geq 1) \end{array}\right]$$

$\{x \geq 1\}$

$\Rightarrow$ 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

$$\left[\!\left[\begin{array}{lll} 1 & : & c_1 = 1 \\ & & x := x+1 \\ 2 & : & c_1 = 2 \end{array}\;\middle\|\;\begin{array}{lll} 3 & : & c_2 = 3 \\ & & x := x+1 \\ 4 & : & c_2 = 4 \end{array}\right]\!\right]$$

- Owicki & 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 Lamport's method

See details in:

R. Cousot. Reasoning about program invariance proof methods. Res. rep. CRIN-80-P050, Centre de Recherche en Informatique de Nancy (CRIN), Institut National Polytechnique de Lorraine, Nancy, France, July 1980. `http://www.di.ens.fr/~cousot/publications.www/CRIN-80-P050-jul-1980.PDF`.

WHAT ABOUT JONES'

RELY / GUARANTEE ?

Reachable states

$$R = \mathrm{lfp}\ F$$

$$F(X) = I \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$$

$$= I \cup \{s' \mid \exists s \in X : \bigvee_{i=1}^{n} s \xrightarrow{\vec{t}_i} s'\}$$

$$= \bigcup_{i=1}^{n} \Big(I \cup \{s' \mid \exists s \in X : s \xrightarrow{\vec{t}_i} s'\} \cup$$

$$\bigcup_{\substack{j=1\\ j \neq i}}^{n} \{s' \mid \exists s \in X : s \xrightarrow{\vec{t}_j} s'\}$$

$$= R(G)\,X$$

where $G(X) = \bigcup_{\substack{j=1\\ j \neq i}}^{n} \{s' \mid \exists s \in X : s \xrightarrow{\vec{t}_j} s'\}$ ← guarantee

$R(G)\,X = \bigcup_{\substack{j=1\\ j \neq i}}^{n} \{s' \mid \exists s \in X : s \xrightarrow{\vec{t}_j} s'\}$ ← rely (assuming guarantee)

Reachable states

Theorem

$$\mathrm{lfp}\, F = \mathrm{lfp}\ \lambda X \,.\, R(G(X))\, X$$

proof

The least fixpoint of

$$X = F(X)$$

is the same as the least fixpoint of the system of equation

$$X = R(Y)\, X$$
$$Y = G(X)$$

by the theorem of asynchronous iterations with memory (Cousot & Cousot, 1977)

□

JONES RELY / GUARANTEE

- Apply the fixpoint induction principle to
  $$\text{lfp}\ \lambda\langle X, Y\rangle \cdot \langle R(Y)X\ ,\ G(X)\rangle$$
- up to the Lamport abstraction α_L, assigning to each control point an assertion on
  - the shared variable
  - the control point of the other process

(or Owicki & Gries with auxiliary variables ;)

- Cliff B. Jones:
  Tentative Steps Toward a Development Method for Interfering Programs. ACM Trans. Program. Lang. Syst. 5(4): 596-619 (1983)

- Joey W. Coleman, Cliff B. Jones:
  A Structural Proof of the Soundness of Rely/guarantee Rules. J. Log. Comput. 17(4): 807-841 (2007)

APPLICATIONS

Astrée A

- **Astrée** : a static analyser of C for synchronous control-command embedded software

- **Astrée 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

CONCLUSION

Conclusion

- Too many computer scientists are tinker[wo]men (bricoleu[rs/ses])
- 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 (Cousot & Cousot, POPL 2012)

THE END