Principles of Abstract Interpretation
MIT press

Ch. **25**, Abstract reachability/invariance/safety verification semantics

Patrick Cousot

pcousot.github.io

PrAbsInt@gmail.com   github.com/PrAbsInt/

These slides are available at
http://github.com/PrAbsInt/slides/slides-25--verification-semantics-PrAbsInt.pdf

Chapter 25

Ch. **25**, Abstract reachability/invariance/safety verification semantics

- Given a well-defined abstract domain (Chapter **21**)

$$\mathbb{D}^{\mathtt{¤}} \triangleq \langle \mathbb{P}^{\mathtt{¤}}, \sqsubseteq^{\mathtt{¤}}, \bot^{\mathtt{¤}}, \sqcup^{\mathtt{¤}}, \mathsf{assign}_{\mathtt{¤}}[\![\mathsf{x}, \mathsf{A}]\!], \mathsf{test}^{\mathtt{¤}}[\![\mathsf{B}]\!], \overline{\mathsf{test}}^{\mathtt{¤}}[\![\mathsf{B}]\!]\rangle$$

a program specification

$$\mathcal{S}^{\mathtt{¤}}[\![\mathsf{P}]\!] \in \mathsf{labx}[\![\mathsf{P}]\!] \to \mathbb{P}^{\mathtt{¤}}$$

and an initial precondition $\mathcal{R}_0 \in \mathbb{P}^{\mathtt{¤}}$, the **verification problem** is to prove that

$$\widehat{\mathcal{S}}^{\mathtt{¤}}[\![\mathsf{P}]\!]\,\mathcal{R}_0 \mathrel{\dot{\sqsubseteq}^{\mathtt{¤}}} \mathcal{S}^{\mathtt{¤}}[\![\mathsf{P}]\!].$$

- The Turing/Naur/Floyd approach is

$$\begin{aligned}
& \widehat{\mathcal{S}}^{\mathtt{¤}}[\![\mathsf{P}]\!]\,\mathcal{R}_0 \mathrel{\dot{\sqsubseteq}^{\mathtt{¤}}} \mathcal{S}^{\mathtt{¤}}[\![\mathsf{P}]\!] \\
\Leftrightarrow\ & \mathsf{lfp}^{\dot{\sqsubseteq}^{\mathtt{¤}}}\, \mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0 \mathrel{\dot{\sqsubseteq}^{\mathtt{¤}}} \mathcal{S}^{\mathtt{¤}}[\![\mathsf{P}]\!] && \text{(Chapter } \mathbf{23} \text{ (Abstract equational semantics))} \\
\Leftrightarrow\ & \exists \mathcal{I} \in \mathsf{labs}[\![\mathsf{S}]\!] \to \mathbb{P}^{\mathtt{¤}} \,.\, \mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0(\mathcal{I}) \mathrel{\dot{\sqsubseteq}^{\mathtt{¤}}} \mathcal{I} \wedge \mathcal{I} \mathrel{\dot{\sqsubseteq}^{\mathtt{¤}}} \mathcal{S}^{\mathtt{¤}}[\![\mathsf{P}]\!] \\
& && \text{(Chapter } \mathbf{24} \text{ (Fixpoint induction))}
\end{aligned}$$

Reachability specification, invariant, inductive invariant

Reachability specification

Definition (25.1) A reachability specification

$$\mathcal{S}^{\varrho}[\![\mathtt{S}]\!] \in \mathsf{labx}[\![\mathtt{S}]\!] \to \wp(\mathbb{E}\mathbb{v}^{\varrho})$$

attaches states $\mathcal{S}^{\varrho}[\![\mathtt{S}]\!]\ell$ at each point ℓ of a program component $\mathtt{S}$.

en.wikipedia.org/wiki/Invariant_(mathematics)#Invariants_in_computer_science
en.wikipedia.org/wiki/Loop_invariant

Example of reachability specification

- What do you think of the following specification?

P	ℓ_i	$\mathcal{S}^{\vec{\varrho}}[\![\mathsf{P}]\!]\ell_i, \quad i = 1, 2$
ℓ_1 /* $x \geqslant 0$ */	ℓ_1	$\{\rho \in \mathbb{E}\mathrm{v} \mid \rho(\mathsf{x}) \geqslant 0)\}$
`x = x - 1 ;`		
ℓ_2 /* $x < 0$ */	ℓ_2	$\{\rho \in \mathbb{E}\mathrm{v} \mid \rho_0(\mathsf{x}) < 0\}$

$\square$

- It is not an invariant *i.e.* always true during execution
- Either the program is wrong (should be *e.g.* `x = -x-1 ;`)
- Or the specification is wrong (should be *e.g.* ℓ_2 /* $x \geqslant -1$ */)

Invariant specification

Definition (25.1, invariant reachability specification) A reachability/forward specification $\mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!]$ is invariant for an initial specification $\mathcal{R}_0 \in \wp(\mathbb{E}\mathbb{v}^{\varrho})$ if and only if

$$\forall \ell \in \mathsf{labx}[\![\mathtt{S}]\!] \;.\; \widehat{\mathcal{S}}^{\,\vec{\varrho}}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\ell \quad\subseteq\quad \mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!]\,\ell.$$

(For all program points $\ell \in \mathsf{labx}[\![\mathtt{P}]\!]$, $\mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!]\,\ell$ describes a superset of the reachable states at that program point.)

Example of reachability invariant

P	ℓ_i	$\mathcal{S}^{\vec{\varrho}}[\![\mathsf{P}]\!]\ell_i, \quad i = 1, \ldots, 76$
`/*` $x = x_0$ `*/`		$\mathcal{R}_0 \;=\; \{\langle\rho_0,\ \rho\rangle \mid \rho_0 = \rho \in \mathbb{E}\mathbb{v}\}$
`while` ℓ_1 `(x != 2) {`	ℓ_1	$\{\langle\rho_0,\ \rho\rangle \mid (\rho = \rho_0) \vee (\rho_0(\mathsf{x}) \notin \{0,1,2\} \wedge \rho(\mathsf{x}) = 2)\}$
`/*` ℓ_2 : $x = x_0 \neq 2$ `*/`		
`if` ℓ_2 `(x == 0)`	ℓ_2	$\{\langle\rho,\ \rho\rangle \mid \rho(\mathsf{x}) \neq 2\}$
ℓ_3 `/*` $x = x_0 = 0$ `*/`	ℓ_3	$\{\langle\rho,\ \rho\rangle \mid \rho(\mathsf{x}) = 0\}$
`break ;`		
`/*` ℓ_4 : $x = x_0 \neq 0$ `*/`	ℓ_4	$\{\langle\rho,\ \rho\rangle \mid \rho(\mathsf{x}) \notin \{0,2\}\}$
`if` ℓ_4 `(x == 1)`		
ℓ_5 `/*` $x = x_0 = 1$ `*/`	ℓ_5	$\{\langle\rho,\ \rho\rangle \mid \rho(\mathsf{x}) = 1\}$
`break ;`		
ℓ_6 `/*` $x = x_0 \notin \{0,1,2\}$ `*/`	ℓ_6	$\{\langle\rho,\ \rho\rangle \mid \rho(\mathsf{x}) \notin \{0,1,2\}\}$
`x = 2 ;`		
`}`		
ℓ_7 `/*` $(x_0 \in \{0,1\} \wedge x = x_0) \vee x = 2$ `*/`	ℓ_7	$\{\langle\rho_0,\ \rho\rangle \mid (\rho_0(\mathsf{x}) \in \{0,1\} \wedge \rho(\mathsf{x}) = \rho_0(\mathsf{x})) \vee \rho(\mathsf{x}) = 2\}$

Structural invariance proof

- The invariance proof $\widehat{\mathcal{S}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{R}_0 \mathrel{\dot{\subseteq}} \mathcal{S}^{\vec{\varrho}}[\![\mathsf{S}]\!]$ can be done by structural induction on the program component S.

- This idea was introduced by Hoare logic studied in Chapter **26**

Example of structural invariance proof

To prove

$$\ell_1 /\star\ \ x = 0\ \ \star/ \ \texttt{x = x - 1 ;}\ \ell_2 /\star\ \ x = -1\ \ \star/\ \texttt{x = x + 1 ;}\ \ell_3 /\star\ \ x = 0\ \ \star/$$

we can prove separately

- $\ell_1 /\star\ \ x = 0\ \ \star/$ `x = x - 1 ;` $\ell_2 /\star\ \ x = -1\ \ \star/$ and
- $\ell_2 /\star\ \ x = -1\ \ \star/$ `x = x + 1 ;` $\ell_3 /\star\ \ x = 0\ \ \star/$.

Counter-example of structural invariance proof

For the following reachability specification

$$\ell_1 \;/\star\; x = 0 \;\star/\; \texttt{x = x - 1 ;} \;\ell_2\; /\star\; \mathtt{tt} \;\star/\; \texttt{x = x + 1 ;} \;\ell_3\; /\star\; x = 0 \;\star/$$

- we can prove $\ell_1 \;/\star\; x = 0 \;\star/\; \texttt{x = x - 1 ;} \;\ell_2\; /\star\; \mathtt{tt} \;\star/$
- but not $\ell_2 \;/\star\; \mathtt{tt} \;\star/\; \texttt{x = x + 1 ;} \;\ell_3\; /\star\; x = 0 \;\star/$

since the precondition $/\star\; \mathtt{tt} \;\star/$ is not strong enough to apply a purely local reasoning.

Inductive invariant

Definition (25.9, inductive invariant) An inductive reachability specification $\mathcal{I}^{\vec{\varrho}}[\![\mathsf{S}]\!] \in \mathsf{labs}[\![\mathsf{S}]\!] \to \wp(\mathbb{E}\mathbb{v}^{\varrho})$ is stronger than the reachability specification $\mathcal{S}^{\vec{\varrho}}[\![\mathsf{S}]\!]$ (*i.e.* $\mathcal{I}^{\vec{\varrho}}[\![\mathsf{S}]\!] \mathrel{\dot{\subseteq}} \mathcal{S}^{\vec{\varrho}}[\![\mathsf{S}]\!]$) and can be proved to be invariant by the following induction on S's computation steps:

- The invariant holds on program entry;
- If the invariant holds at any program point and a computation step is executed then the invariant holds at the next program point.

Example of inductive invariant

P	ℓ_i	non-inductive specification $\mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!](\ell_i)$	inductive invariant $\mathcal{I}^{\vec{\varrho}}[\![\mathtt{S}]\!](\ell_i)$
`/* tt */`		$\mathcal{R}_0 \quad = \quad \mathbb{E}\mathbb{v}$	$\mathcal{R}_0 \quad = \quad \mathbb{E}\mathbb{v}$
ℓ_1 `x = 1 ;`	ℓ_1	$\mathbb{E}\mathbb{v}$	$\mathbb{E}\mathbb{v}$
`while` ℓ_2 `(x > 0) /* tt */`	ℓ_2	$\mathbb{E}\mathbb{v}$	$\{\rho \in \mathbb{E}\mathbb{v} \mid \rho(\mathtt{x}) > 0\}$
ℓ_3 `x = x + 1 ;`	ℓ_3	$\mathbb{E}\mathbb{v}$	$\{\rho \in \mathbb{E}\mathbb{v} \mid \rho(\mathtt{x}) > 0\}$
ℓ_4 `/* ff */`	ℓ_4	$\varnothing$	$\varnothing$

The specification is not inductive for the iteration statement $\mathtt{S}$ since

$$\overline{\mathsf{test}}^{\vec{\varrho}}[\![\mathtt{x > 0}]\!](\mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!](\ell_2)) = \{\rho \in \mathbb{E}\mathbb{v} \mid \rho(\mathtt{x}) \leqslant 0\} \not\subseteq \varnothing = \mathcal{S}^{\vec{\varrho}}[\![\mathtt{S}]\!](\ell_4)$$

where $\ell_4 = \mathsf{after}[\![\mathtt{S}]\!]$.

Turing/Floyd/Naur invariance proof method

To prove that a reachability specification $\mathcal{S}^{\vec{\varrho}}[\![\mathsf{s}]\!]$ is invariant, it is necessary and sufficient to find an invariant specification $\mathcal{I}^{\vec{\varrho}}[\![\mathsf{s}]\!]$ (there always exists one) such that

(1) the invariant specification $\mathcal{I}^{\vec{\varrho}}[\![\mathsf{s}]\!]$ is inductive (which implies that $\widehat{\boldsymbol{\mathcal{S}}}^{\,\vec{\varrho}}[\![\mathsf{s}]\!]\,\mathcal{R}_0 \mathrel{\dot{\subseteq}} \mathcal{I}^{\vec{\varrho}}[\![\mathsf{s}]\!]$);

(2) the invariant specification is stronger than the reachability specification $\mathcal{I}^{\vec{\varrho}}[\![\mathsf{s}]\!] \mathrel{\dot{\subseteq}} \mathcal{S}^{\vec{\varrho}}[\![\mathsf{s}]\!]$

proving, by transitivity that, $\widehat{\boldsymbol{\mathcal{S}}}^{\,\vec{\varrho}}[\![\mathsf{s}]\!]\,\mathcal{R}_0 \mathrel{\dot{\subseteq}} \mathcal{S}^{\vec{\varrho}}[\![\mathsf{s}]\!]$.

Abstract specification, abstract invariant, abstract inductive invariant

Generalization to abstract specification, abstract invariant, abstract inductive invariant, and abstract structural proof method

- We generalize from $\wp(\mathbb{E}\mathbb{v}^{\varrho})$ to an abstract domain $\overline{\mathbb{P}}^{\,\texttt{¤}}$
- Informally, $\mathcal{S}^{\texttt{¤}}[\![\mathsf{s}]\!]$ is *invariant* for s (*i.e.* $\widehat{\mathbb{S}}^{\,\texttt{¤}}[\![\mathsf{s}]\!]\,\mathcal{R}_0 \mathrel{\dot{\sqsubseteq}^{\texttt{¤}}} \mathcal{S}^{\texttt{¤}}[\![\mathsf{s}]\!]$) if and only if
  (1) there exists an *inductive invariant* $\mathcal{I} \in \mathsf{labs}[\![\mathsf{s}]\!] \to \overline{\mathbb{P}}^{\,\texttt{¤}}$
  (2) which is *stronger* than the specification (*i.e.* $\mathcal{I} \mathrel{\dot{\sqsubseteq}^{\texttt{¤}}} \mathcal{S}^{\texttt{¤}}[\![\mathsf{s}]\!]$) and is *inductive* that is, by definition, satisfies the *verification conditions*

$$\widehat{\mathcal{V}}^{\,\texttt{¤}}[\![\mathsf{s}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad \triangleq \quad (\mathsf{E}[\![\mathsf{s}]\!]\,\mathcal{R}_0(\mathcal{I}) \mathrel{\dot{\sqsubseteq}^{\texttt{¤}}} \mathcal{I}) \tag{25.22}$$

  that is (25.12) to (25.21) below.

- Verification conditions for a program $\mathtt{P ::= Sl}\ \ell'$

$$\widehat{\mathscr{V}}^{\,¤}[\![\mathtt{P}]\!] \quad = \quad \widehat{\mathscr{V}}^{\,¤}[\![\mathtt{Sl}]\!] \tag{25.12}$$

- Verification conditions for a skip statement $\mathtt{S ::= ;}$

$$\begin{aligned}
\widehat{\mathscr{V}}^{\,¤}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \quad \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\mathsf{at}[\![\mathtt{S}]\!]} && (25.13.a)\\
\wedge & \quad \mathcal{I}_{\mathsf{at}[\![\mathtt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\mathsf{after}[\![\mathtt{S}]\!]} && (25.13.b)
\end{aligned}$$

- Verification conditions for an assignment statement $\mathtt{S ::= x = E ;}$

$$\begin{aligned}
\widehat{\mathscr{V}}^{\,¤}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \quad \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\mathsf{at}[\![\mathtt{S}]\!]} && (25.14.a)\\
\wedge & \quad \mathsf{assign}_{¤}[\![\mathtt{x}, \mathtt{E}]\!]\,\mathcal{I}_{\mathsf{at}[\![\mathtt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\mathsf{after}[\![\mathtt{S}]\!]} && (25.14.b)
\end{aligned}$$

- Verification conditions for a conditional statement $\texttt{S ::= if (B) S}_t$

$$
\begin{aligned}
\widehat{\mathscr{V}}^{\,\texttt{¤}}[\![\texttt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \mathcal{R}_0 \sqsubseteq^{\texttt{¤}} \mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} & (25.15.a)\\
\wedge \quad & \widehat{\mathscr{V}}^{\,\texttt{¤}}[\![\texttt{S}_t]\!]\,(\mathsf{test}^{\texttt{¤}}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]})\,\mathcal{I} & (25.15.b)\\
\wedge \quad & \overline{\mathsf{test}}^{\texttt{¤}}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} \sqsubseteq^{\texttt{¤}} \mathcal{I}_{\mathsf{after}[\![\texttt{S}]\!]} & (25.15.c)
\end{aligned}
$$

- Verification conditions for a conditional statement $\texttt{S ::= if (B) S}_t\ \texttt{else S}_f$

$$
\begin{aligned}
\widehat{\mathscr{V}}^{\,\texttt{¤}}[\![\texttt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \mathcal{R}_0 \sqsubseteq^{\texttt{¤}} \mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} & (25.16.a)\\
\wedge \quad & \widehat{\mathscr{V}}^{\,\texttt{¤}}[\![\texttt{S}_t]\!]\,(\mathsf{test}^{\texttt{¤}}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]})\,\mathcal{I} & (25.16.b)\\
\wedge \quad & \widehat{\mathscr{V}}^{\,\texttt{¤}}[\![\texttt{S}_t]\!]\,(\overline{\mathsf{test}}^{\texttt{¤}}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]})\,\mathcal{I} & (25.16.c)
\end{aligned}
$$

- Verification conditions for an empty statement list $\texttt{Sl} ::= \epsilon$

$$\widehat{\mathscr{V}}^{\,¤}[\![\texttt{Sl}]\!]\,\mathcal{R}_0\,\mathcal{I} \;=\; \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\mathsf{at}[\![\texttt{Sl}]\!]} \tag{25.17}$$

- Verification conditions for a statement list $\texttt{Sl} ::= \texttt{Sl}'\ \texttt{S}$

$$\begin{aligned}\widehat{\mathscr{V}}^{\,¤}[\![\texttt{Sl}]\!]\,\mathcal{R}_0\,\mathcal{I} &= \widehat{\mathscr{V}}^{\,¤}[\![\texttt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} && \text{when } \texttt{Sl}' ::= \epsilon\\ &= \widehat{\mathscr{V}}^{\,¤}[\![\texttt{Sl}']\!]\,\mathcal{R}_0\,\mathcal{I} \wedge \widehat{\mathscr{V}}^{\,¤}[\![\texttt{S}]\!]\,\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]}\,\mathcal{I} && \text{otherwise}\end{aligned} \tag{25.18}$$

- Verification conditions for a break statement $\texttt{S} ::= \ell\ \texttt{break ;}$

$$\widehat{\mathscr{V}}^{\,¤}[\![\texttt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \;=\; \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} \wedge \mathcal{I}_{\mathsf{after}[\![\texttt{S}]\!]} \sqsubseteq^{¤} \bot^{¤} \tag{25.20}$$

- Verification conditions for an iteration statement $\texttt{S} ::= \texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b$

$$
\begin{aligned}
\hat{\mathcal{V}}^{\,¤}[\![\texttt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} && (25.19.a)\\
\wedge \quad & \hat{\mathcal{V}}^{\,¤}[\![\texttt{S}_b]\!](\mathsf{test}^{¤}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]})\,\mathcal{I} && (25.19.b)\\
\wedge \quad & \overline{\mathsf{test}}^{\,¤}[\![\texttt{B}]\!]\mathcal{I}_{\mathsf{at}[\![\texttt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\mathsf{after}[\![\texttt{S}]\!]} && (25.19.c)\\
\wedge \quad & \forall \ell \in \mathsf{breaks\text{-}of}[\![\texttt{S}_b]\!]\,.\,\mathcal{I}_{\ell} \sqsubseteq^{¤} \mathcal{I}_{\mathsf{after}[\![\texttt{S}]\!]} && (25.19.d)
\end{aligned}
$$

- Verification conditions for a compound statement $\texttt{S} ::= \texttt{\{ Sl \}}$

$$
\hat{\mathcal{V}}^{\,¤}[\![\texttt{S}]\!] \quad = \quad \hat{\mathcal{V}}^{\,¤}[\![\texttt{Sl}]\!] \qquad (25.21)
$$

Example

while ℓ_1 (x!=2) { if ℓ_2 (x==0) ℓ_3 break; if ℓ_4 (x==1) ℓ_5 break; ℓ_6 x=2; } ℓ_7

```
R0 => l1                  (23.1) (23.6) (23.8.a)
test[(x != 2)]l1 => l2    (23.1) (23.6) (23.8.b) (23.6) (23.4.a)
test[(x == 0)]l2 => l3    (23.1) (23.6) (23.8.b) (23.6) (23.4.b) (23.9)
ntest[(x == 0)]l2 => l4   (23.1) (23.6) (23.8.b) (23.6) (23.4.c)
test[(x == 1)]l4 => l5    (23.1) (23.6) (23.8.b) (23.4.b) (23.9)
ntest[(x == 1)]l4 => l6   (23.1) (23.6) (23.8.b) (23.4.c)
assign[x=2]l6 => l1       (23.1) (23.6) (23.8.b) (23.3.b)
ntest[(x != 2)]l1 => l7   (23.1) (23.6) (23.8.c)
l3 => l7                  (23.1) (23.6) (23.8.d)
l5 => l7                  (23.1) (23.6) (23.8.d)
```


Structural inductive abstract invariance proof method

Theorem (25.11, Sound and complete abstract invariance proof method) Let $\mathbb{D}^{¤}$ be an abstract domain in Definition 21.1, $\mathcal{S}^{¤}[\![\mathsf{s}]\!] \in \mathsf{labs}[\![\mathsf{s}]\!] \rightarrow \overline{\mathbb{P}}^{¤}$ be an abstract specification and $\mathcal{R}_0 \in \overline{\mathbb{P}}^{¤}$ be an abstract precondition of program component s. The *invariance proof method* is

$$\widehat{\mathcal{S}}^{¤}[\![\mathsf{s}]\!]\,\mathcal{R}_0 \mathrel{\dot{\sqsubseteq}^{¤}} \mathcal{S}^{¤}[\![\mathsf{s}]\!] \quad\Leftrightarrow\quad \exists \mathcal{I} \in \mathsf{labs}[\![\mathsf{s}]\!] \rightarrow \overline{\mathbb{P}}^{¤}\,.\,\widehat{\mathcal{V}}^{¤}[\![\mathsf{s}]\!]\,\mathcal{R}_0\,\mathcal{I} \wedge \mathcal{I} \mathrel{\dot{\sqsubseteq}^{¤}} \mathcal{S}^{¤}[\![\mathsf{s}]\!]$$

($\Leftarrow$ is soundness and $\Rightarrow$ is completeness)

Note: a static analysis consists in (1) inferring $\mathcal{I}$ and (2) checking the verification conditions $\widehat{\mathcal{V}}^{¤}[\![\mathsf{s}]\!]\,\mathcal{I}$.

Proof of Theorem 25.11

- By Theorem 23.20, the solution of the system of equations $\widehat{\mathscr{E}}^{\,¤}[\![\mathsf{P}]\!]\,\mathcal{R}_0$ of the form $\mathcal{X} = \mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0\,(\mathcal{X})$ is $\mathsf{lfp}^{\dot\sqsubseteq^{¤}}\,\mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0 = \widehat{\mathscr{S}}^{\,¤}[\![\mathsf{P}]\!]\,\mathcal{R}_0$ where $\mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0$ is pointwise $\sqsubseteq^{¤}$-continuous hence increasing.
- Therefore, the proof that $\widehat{\mathscr{S}}^{\,¤}[\![\mathsf{S}]\!]\,\mathcal{R}_0 \mathrel{\dot\sqsubseteq^{¤}} \mathcal{S}^{¤}[\![\mathsf{S}]\!]$ is equivalent to $\mathsf{lfp}^{\dot\sqsubseteq^{¤}}\,\mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0 \mathrel{\dot\sqsubseteq^{¤}} \mathcal{S}^{¤}[\![\mathsf{S}]\!]$.
- Applying the fixpoint induction Theorem 24.1, this is equivalent to

$$\exists \mathcal{I} \in \mathsf{labs}[\![\mathsf{S}]\!] \to \overline{\mathbb{P}}^{¤}\,.\,\mathsf{E}[\![\mathsf{P}]\!]\,\mathcal{R}_0(\mathcal{I}) \mathrel{\dot\sqsubseteq^{¤}} \mathcal{I} \wedge \mathcal{I} \mathrel{\dot\sqsubseteq^{¤}} \mathcal{S}^{¤}[\![\mathsf{S}]\!]$$

- It remains to state the boolean verification conditions

$$\widehat{\mathscr{V}}^{\,¤}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad \triangleq \quad \mathsf{E}[\![\mathsf{S}]\!]\,\mathcal{R}_0(\mathcal{I}) \mathrel{\dot\sqsubseteq^{¤}} \mathcal{I} \tag{25.22}$$

  which are an abstraction $\widehat{\mathscr{V}}^{\,¤}[\![\mathsf{S}]\!] = \vec{\alpha}\,(\mathsf{E}[\![\mathsf{P}]\!])$ of the equations $\mathsf{E}[\![\mathsf{P}]\!]$ where $\vec{\alpha}\,(F) \triangleq \mathcal{R}_0 \mapsto \mathcal{I} \mapsto F(\mathcal{I}) \mathrel{\dot\sqsubseteq^{¤}} \mathcal{I}$.
- We calculate $\widehat{\mathscr{V}}^{\,¤}[\![\mathsf{S}]\!]$ by applying $\vec{\alpha}$ to $\mathsf{E}[\![\mathsf{P}]\!]$ and simplifying.
- We proceed by structural induction on S. See the details in the book. □

Alternative verification conditions

In (25.19.d), $\ell \in \text{breaks-of}[\![\mathtt{S}_b]\!]$ if and only if there is a break statement $\mathtt{S} ::= \ell\ \mathtt{break}\ \mathtt{;}$ such that $\ell = \text{at}[\![\mathtt{S}]\!]$ and $\text{break-to}[\![\mathtt{S}]\!] = \text{after}[\![\mathtt{S}]\!]$ so that the verification condition (25.19.d) that is $\forall \ell \in \text{breaks-of}[\![\mathtt{S}_b]\!] \mathrel{.} \mathcal{I}_\ell \sqsubseteq^{¤} \mathcal{I}_{\text{after}[\![\mathtt{S}]\!]}$ can be distributed as $\mathcal{I}_{\text{at}[\![\mathtt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\text{break-to}[\![\mathtt{S}]\!]}$ to all such break statements. We get equivalently

- Verification conditions for an iteration statement $\mathtt{S} ::= \mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b$

$$
\begin{aligned}
\widehat{\mathscr{V}}^{¤}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad & \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\text{at}[\![\mathtt{S}]\!]} && (25.19'.\text{a})\\
\wedge\quad & \widehat{\mathscr{V}}^{¤}[\![\mathtt{S}_b]\!](\text{test}^{¤}[\![\mathtt{B}]\!]\mathcal{I}_{\text{at}[\![\mathtt{S}]\!]})\,\mathcal{I} && (25.19'.\text{b})\\
\wedge\quad & \overline{\text{test}}^{¤}[\![\mathtt{B}]\!]\mathcal{I}_{\text{at}[\![\mathtt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\text{after}[\![\mathtt{S}]\!]} && (25.19'.\text{c})
\end{aligned}
$$

- Verification conditions for a break statement $\mathtt{S} ::= \ell\ \mathtt{break}\ \mathtt{;}$

$$
\widehat{\mathscr{V}}^{¤}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\mathcal{I} \quad = \quad \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}_{\text{at}[\![\mathtt{S}]\!]} \wedge \mathcal{I}_{\text{at}[\![\mathtt{S}]\!]} \sqsubseteq^{¤} \mathcal{I}_{\text{break-to}[\![\mathtt{S}]\!]} \qquad (25.20')
$$

Verifying that an invariant is inductive

- Invariance proof methods are a special case with $\overline{\mathbb{P}}^{¤} = \mathbb{P}^{¤}$ where $\vec{\varrho} \in \{\vec{r}, \vec{R}\}$, $\mathbb{E}\mathbb{v}^{\vec{r}} = \mathbb{E}\mathbb{v}$ for assertional invariance, and $\mathbb{E}\mathbb{v}^{\vec{R}} = \mathbb{E}\mathbb{v} \times \mathbb{E}\mathbb{v}$ for relational invariance.
- An invariant $\mathcal{I} \in \mathbb{L} \to \mathbb{P}^{¤}$ is defined independently of a precondition $\mathcal{R}_0$ which is equivalent to defining $\mathcal{R}_0 \triangleq \mathcal{I}(\mathsf{at}[\![\mathsf{S}]\!])$ and assuming that $\mathcal{I}(\mathsf{at}[\![\mathsf{S}]\!])$ is an hypothesis.
- An invariant can be checked to be inductive by

$$\begin{array}{rcl} \widehat{\mathcal{F}}^{\vec{\varrho}}[\![\mathsf{S}]\!] & \in & (\mathbb{L} \to \mathbb{P}^{\vec{\varrho}}) \to \mathbb{B} \\ \widehat{\mathcal{F}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{I} & \triangleq & \widehat{\mathcal{V}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{I}(\mathsf{at}[\![\mathsf{S}]\!])\,\mathcal{I} \end{array} \tag{25.30}$$

- Lemma 25.31: $\widehat{\mathcal{V}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\mathcal{I} = \mathcal{R}_0 \sqsubseteq^{¤} \mathcal{I}(\mathsf{at}[\![\mathsf{S}]\!]) \wedge \widehat{\mathcal{F}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{I}$.
- So, the verification conditions (25.12) to (25.21) do apply
- For example (25.14.a) and (25.14.b) become

$$\widehat{\mathcal{F}}^{\vec{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{I} \quad = \quad \mathsf{assign}_{\vec{\varrho}}[\![\mathsf{x}, \mathsf{E}]\!]\,\mathcal{I}_{\mathsf{at}[\![\mathsf{S}]\!]} \subseteq \mathcal{I}_{\mathsf{after}[\![\mathsf{S}]\!]}$$

Conclusion

- The success of Turing/Floyd/Naur invariance verification method of Chapter **25** is that it is the most abstract, sound, and complete invariance proof method.
  - Being the most abstract, it cannot be further simplified.
  - Being sound, it is infallible.
  - Being complete, it is always applicable and never fails.

Conclusion

- Is Turing/Floyd/Naur invariance verification method automatizable?
- Yes (using theorem provers/SMT solvers, with the limits of undecidability)
- Does this work in practice?
- Yes, in the small, *e.g.* Frama-C has many examples of textbook programs (*e.g.* Sum of values in an array)
- Yes, in the large, if you have a dedicated team of gifted researchers, *e.g.* CompCert proved in Coq
- No, in the very large (millions of lines) since the invariants are hard to specify, proofs are much bigger than the programs and must be maintained when programs are modified, and prover's failures are unpredictable and very hard to circumvent, which has a huge cost.

Conclusion

- Is Turing/Floyd/Naur verification method generalizable beyond invariance?
- Yes, to any specification for the abstract interpreter, as shown in this Chapter **25**, and summarized in the introductory slide, *e.g*.
  - for reachability/invariance [Floyd, 1967; Naur, 1966; Turing, 1949] with numerous variants [P. Cousot and R. Cousot, 1982]
  - for safety properties [P. Cousot and R. Cousot, 2012]

Bibliography

Cousot, Patrick and Radhia Cousot (June 1982). "Induction principles for proving invariance properties of programs". In: D. Néel, ed. *Tools & Notions for Program Construction: an Advanced Course*. Cambridge University Press, Cambridge, UK, pp. 75–119.

— (2012). "An abstract interpretation framework for termination". In: *POPL*. ACM, pp. 245–258.

Floyd, Robert W. (1967). "Assigning meaning to programs". In: J.T. Schwartz, ed. *Proc. Symp. in Applied Math.* Vol. 19. Amer. Math. Soc., pp. 19–32.

Naur, Peter (1966). "Proofs of algorithms by general snapshots". *BIT* 6, pp. 310–316.

Turing, Alan (1949). "Checking a large routine". In: *Report of a Conference on High Speed Automatic Calculating Machines, University of Cambridge Mathematical Laboratory, Cambridge, England*, pp. 67–69. URL: `http://www.turingarchive.org/browse.php/b/8`.

Home work

Read Ch. **25** "Abstract reachability/invariance/safety verification semantics" of

Principles of Abstract Interpretation
Patrick Cousot
MIT Press

The End, Thank you

