inductive invariants for example program

- weakest inductive invariant: *true* (set of all states)
  contains error states
- strongest inductive invariant (does not contain error states)

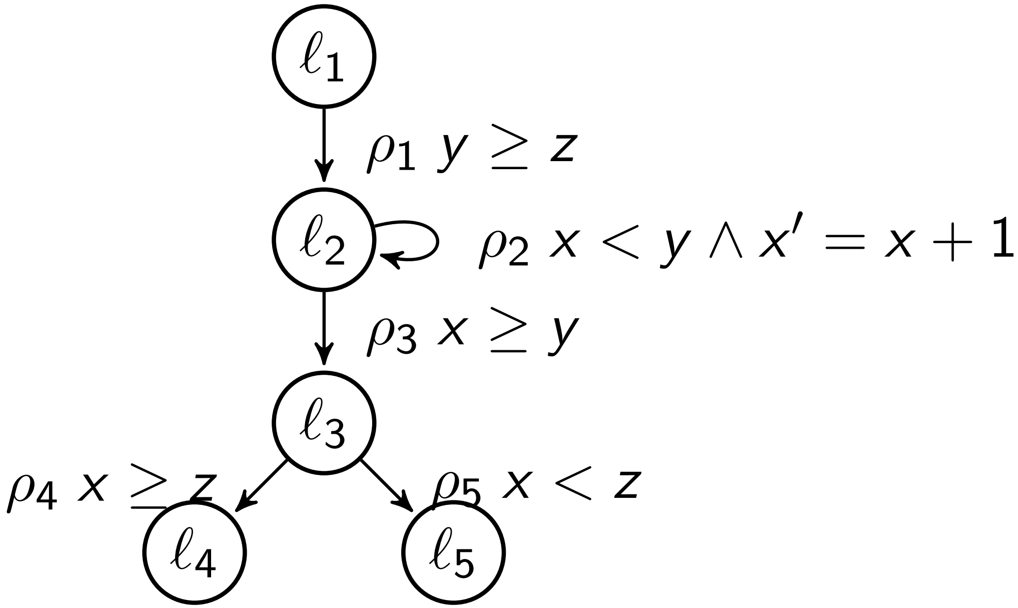
$$
\begin{aligned}
& pc = \ell_1 \lor \\
& (pc = \ell_2 \land y \geq z) \lor \\
& (pc = \ell_3 \land y \geq z \land x \geq y) \lor \\
& (pc = \ell_4 \land y \geq z \land x \geq y)
\end{aligned}
$$

- a slightly weaker inductive invariant also proves the safety of our examples:

$$
\begin{aligned}
& pc = \ell_1 \lor \\
& (pc = \ell_2 \land y \geq z) \lor \\
& (pc = \ell_3 \land y \geq z \land x \geq y) \lor \\
& pc = \ell_4
\end{aligned}
$$

- can we drop another conjunct in one of the disjuncts?

```
1:   assume(y >= z);
2:   while (x < y) {
       x++;
     }
3:   assert(x >= z);
4:  exit
5:  error
```



inductive invariant (strict superset of reachable states):

$$
\begin{aligned}
\varphi_{reach} = (&pc = \ell_1 \vee \\
&pc = \ell_2 \wedge y \geq z \vee \\
&pc = \ell_3 \wedge y \geq z \wedge x \geq y \vee \\
&pc = \ell_4)
\end{aligned}
$$

fixpoint iteration

- computation of reachable program states $=$
  iterative application of post on initial program states until
  a fixpoint is reached
  i.e., no new program states are obtained by applying post
- in general, iteration process does not *converge*
  i.e., does not reach fixpoint in finite number of iterations

example: fixpoint iteration *diverges*

$$\rho_2 \;\equiv\; (move(\ell_2, \ell_2) \wedge x + 1 \leq y \wedge x' = x + 1 \wedge skip(y, z))$$

$$\begin{aligned}
post(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - 1 \leq z \wedge x \leq y) \\
post^2(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - 2 \leq z \wedge x \leq y) \\
post^3(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - 3 \leq z \wedge x \leq y) \\
&\dots \\
post^n(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - n \leq z \wedge x \leq y)
\end{aligned}$$

example: fixpoint not reached after n steps, $n \geq 1$

- set of states reachable after applying post twice not included in the union of previous two sets:

$$
\begin{aligned}
(at\_\ell_2 \wedge x - 2 \leq z \wedge x \leq y) \quad & \not\models \\
& at\_\ell_2 \wedge x \leq z \vee \\
& at\_\ell_2 \wedge x - 1 \leq z \wedge x \leq y
\end{aligned}
$$

- set of states reachable after n-fold application of *post* still contains previously unreached states:

$$
\begin{aligned}
\forall n \geq 1 : (at\_\ell_2 \wedge x - n \leq z \wedge x \leq y) \quad & \not\models \\
& at\_\ell_2 \wedge x \leq z \vee \\
& \bigvee_{1 \leq i < n} (at\_\ell_2 \wedge x - i \leq z \wedge x \leq y)
\end{aligned}
$$

abstraction of φ_{reach} by $\varphi^{\#}_{reach}$

- instead of computing φ_{reach},
  compute over-approximation $\varphi^{\#}_{reach}$ such that $\varphi^{\#}_{reach} \supseteq \varphi_{reach}$
- check whether $\varphi^{\#}_{reach}$ contains any error states
- if $\varphi^{\#}_{reach} \wedge \varphi_{err} \models \mathit{false}$ holds then $\varphi_{reach} \wedge \varphi_{err} \models \mathit{false}$,
  and hence the program is safe
- compute $\varphi^{\#}_{reach}$ by applying iteration
- instead of iteratively applying *post*, use
  over-approximation $\mathit{post}^{\#}$ such that always

$$\mathit{post}(\varphi, \rho) \models \mathit{post}^{\#}(\varphi, \rho)$$

- decompose computation of $\mathit{post}^{\#}$ into two steps:
  first, apply *post* and
  then, over-approximate result using a function α such that

$$\forall \varphi : \varphi \models \alpha(\varphi) \ .$$

abstraction of *post* by $post^{\#}$

- given an abstraction function α, define $post^{\#}$:

$$post^{\#}(\varphi, \rho) \;=\; \alpha(post(\varphi, \rho))$$

- compute $\varphi^{\#}_{reach}$:

$$\begin{aligned}\varphi^{\#}_{reach} &= \alpha(\varphi_{init}) \lor \\ &\quad\; post^{\#}(\alpha(\varphi_{init}), \rho_{\mathcal{R}}) \lor \\ &\quad\; post^{\#}(post^{\#}(\alpha(\varphi_{init}), \rho_{\mathcal{R}}), \rho_{\mathcal{R}}) \lor \ldots \\ &= \bigvee\nolimits_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}), \rho_{\mathcal{R}})\end{aligned}$$

- consequence: $\varphi_{reach} \models \varphi^{\#}_{reach}$

predicate abstraction

- construct abstraction using a given set of building blocks, so-called predicates
- predicate $=$ formula over the program variables V
- fix finite set of predicates $Preds = \{p_1, \ldots, p_n\}$
- over-approximation of φ by conjunction of predicates in $Preds$

$$\alpha(\varphi) = \bigwedge\{p \in Preds \mid \varphi \models p\}$$

- computation requires n entailment checks
  ($n =$ number of predicates)

example: compute $\alpha(at\_\ell_2 \wedge y \geq z \wedge x + 1 \leq y)$

- $Preds = \{at\_\ell_1, \ldots, at\_\ell_5, y \geq z, x \geq y\}$

1. check logical consequence between argument to the abstraction function and each of the predicates:

	$y \geq z$	$x \geq y$	at_ℓ_1	at_ℓ_2	at_ℓ_3	at_ℓ_4	at_ℓ_5
$at\_\ell_2 \wedge$ $y \geq z \wedge$ $x + 1 \leq y$	$\models$	$\not\models$	$\not\models$	$\models$	$\not\models$	$\not\models$	$\not\models$

2. result of abstraction = conjunction over entailed predicates

$$\alpha\left(\begin{array}{l} at\_\ell_2 \wedge \\ y \geq z \ \wedge \ x + 1 \leq y \end{array}\right) \ = \ at\_\ell_2 \wedge y \geq z$$

trivial abstraction $\alpha(\varphi) = \mathit{true}$

- result of applying predicate abstraction is *true* if

trivial abstraction $\alpha(\varphi) = \mathit{true}$

- result of applying predicate abstraction is *true* if none of the predicates is entailed by φ (“predicates are too specific”)

trivial abstraction $\alpha(\varphi) = \mathit{true}$

- result of applying predicate abstraction is *true* if none of the predicates is entailed by φ ("predicates are too specific")
  ... always the case if $\mathit{Preds} = \emptyset$

example: predicate abstraction to compute $\varphi^{\#}_{reach}$

- $Preds = \{false, at\_\ell_1, \dots, at\_\ell_5, y \geq z, x \geq y\}$
- over-approximation of the set of initial states φ_{init}:

$$\varphi_1 = \alpha(at\_\ell_1) = at\_\ell_1$$

- apply $post^{\#}$ on φ_1 wrt. each program transition:

$$\varphi_2 = post^{\#}(\varphi_1, \rho_1) = \alpha(\underbrace{at\_\ell_2 \wedge y \geq z}_{post(\varphi_1, \rho_1)}) = at\_\ell_2 \wedge y \geq z$$

$$post^{\#}(\varphi_1, \rho_2) = \cdots = post^{\#}(\varphi_1, \rho_5) = \bigwedge\{false, \dots\} = false$$

apply $post^{\#}$ to $\varphi_2 \;=\; (at\_\ell_2 \wedge y \geq z)$

- application of ρ_1, ρ_4, and ρ_5 on φ_2 results in *false* (since ρ_1, ρ_4, and ρ_5 are applicable only if either at_ℓ_1 or at_ℓ_3 hold)
- for ρ_2 we obtain

  $$post^{\#}(\varphi_2, \rho_2) = \alpha(at\_\ell_2 \wedge y \geq z \wedge x \leq y) = at\_\ell_2 \wedge y \geq z$$

  result is φ_2 and, therefore, is discarded
- for ρ_3 we obtain

  $$\begin{aligned} post^{\#}(\varphi_2, \rho_3) &= \alpha(at\_\ell_3 \wedge y \geq z \wedge x \geq y) \\ &= at\_\ell_3 \wedge y \geq z \wedge x \geq y \\ &= \varphi_3 \end{aligned}$$

apply $post^{\#}$ to $\varphi_3 \;=\; (at\_\ell_3 \wedge y \geq z \wedge x \geq y)$

- ρ_1, ρ_2, and ρ_3: inconsistency with program counter valuation in φ_3
- for ρ_4 we obtain:

$$\begin{aligned} post^{\#}(\varphi_3, \rho_4) &= \alpha(at\_\ell_4 \wedge y \geq z \wedge x \geq y \wedge x \geq z) \\ &= at\_\ell_4 \wedge y \geq z \wedge x \geq y \\ &= \varphi_4 \end{aligned}$$

- for ρ_5 (assertion violation) we obtain:

$$\begin{aligned} post^{\#}(\varphi_3, \rho_5) &= \alpha(at\_\ell_5 \wedge y \geq z \wedge x \geq y \wedge x + 1 \leq z) \\ &= \mathit{false} \end{aligned}$$

- any further application of program transitions does not compute any additional reachable states
- thus, $\varphi^{\#}_{reach} = \varphi_1 \vee \ldots \vee \varphi_4$
- since $\varphi^{\#}_{reach} \wedge at\_\ell_5 \models \mathit{false}$, the program is proven safe

algorithm AbstReach

begin

$\alpha \; := \; \lambda\varphi \,.\, \bigwedge\{p \in \mathit{Preds} \mid \varphi \models p\}$

$\mathit{post}^{\#} \; := \; \lambda(\varphi, \rho) \,.\, \alpha(\mathit{post}(\varphi, \rho))$

$\mathit{ReachStates}^{\#} \; := \; \{\alpha(\varphi_{\mathit{init}})\}$

$\mathit{Parent} \; := \; \emptyset$

$\mathit{Worklist} \; := \; \mathit{ReachStates}^{\#}$

while $\mathit{Worklist} \neq \emptyset$ **do**

$\varphi \; := \;$ choose from $\mathit{Worklist}$

$\mathit{Worklist} \; := \; \mathit{Worklist} \setminus \{\varphi\}$

for each $\rho \in \mathcal{R}$ **do**

$\varphi' \; := \; \mathit{post}^{\#}(\varphi, \rho)$

if $\varphi' \not\models \bigvee \mathit{ReachStates}^{\#}$ **then**

$\mathit{ReachStates}^{\#} \; := \; \{\varphi'\} \cup \mathit{ReachStates}^{\#}$

$\mathit{Parent} \; := \; \{(\varphi, \rho, \varphi')\} \cup \mathit{Parent}$

$\mathit{Worklist} \; := \; \{\varphi'\} \cup \mathit{Worklist}$

return $(\mathit{ReachStates}^{\#}, \mathit{Parent})$

end