Abstraction Refinement

Andreas Podelski

December 13, 2011

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

- abstraction function α

$$\varphi \models \alpha(\varphi)$$

- abstract post $post^{\#}$ for relation ρ

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

- abstract post for program relation $\mathcal{R}$

$$post^{\#}(\Phi, \mathcal{R}) \;=\; \{\alpha(post(\varphi, \rho)) \mid \varphi \in \Phi,\ \rho \in \mathcal{R}\}$$

NB. first argument of $post^{\#}$ is set of sets of states;
post distributes over disjunction, $post^{\#}$ does not

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

- compute $ReachStates^{\#}$ by fixpoint iteration

$$ReachStates^{\#} = \bigvee_{i \geq 0} (post^{\#})^i(\{\alpha(\varphi_{init})\}, \mathcal{R})$$

- $\varphi^{\#}_{reach}$ = disjunction of "abstract states" in $ReachStates^{\#}$

$$\varphi^{\#}_{reach} = \bigvee \{\psi \mid \psi \in ReachStates^{\#}\}$$

- consequence: $\varphi_{reach} \models \varphi^{\#}_{reach}$
- program safe **if** $\varphi^{\#}_{reach} \wedge \varphi_{err} \models false$

NB. abstract state ψ = set of states = $\alpha(\varphi)$ for some φ
in case of predicate abstraction:
abstract state ψ = conjunction of predicates

algorithm AbstReach for given abstraction function α

begin
  $post^{\#} := \lambda(\varphi, \rho) \,.\, \alpha(post(\varphi, \rho))$
  $ReachStates^{\#} := \{\alpha(\varphi_{init})\}$
  $Parent := \emptyset$
  $Worklist := ReachStates^{\#}$
  while $Worklist \neq \emptyset$ **do**
    $\varphi :=$ choose from $Worklist$
    $Worklist := Worklist \setminus \{\varphi\}$
    for each $\rho \in \mathcal{R}$ **do**
      $\varphi' := post^{\#}(\varphi, \rho)$
      if $\varphi' \notin ReachStates^{\#}$ **then**
        $ReachStates^{\#} := \{\varphi'\} \cup ReachStates^{\#}$
        $Parent := \{(\varphi, \rho, \varphi')\} \cup Parent$
        $Worklist := \{\varphi'\} \cup Worklist$
  return $(ReachStates^{\#}, Parent)$
end

predicate abstraction

- predicate = formula over the program variables V
- fix finite set of predicates $\mathit{Preds} = \{p_1, \ldots, p_n\}$
- abstraction function

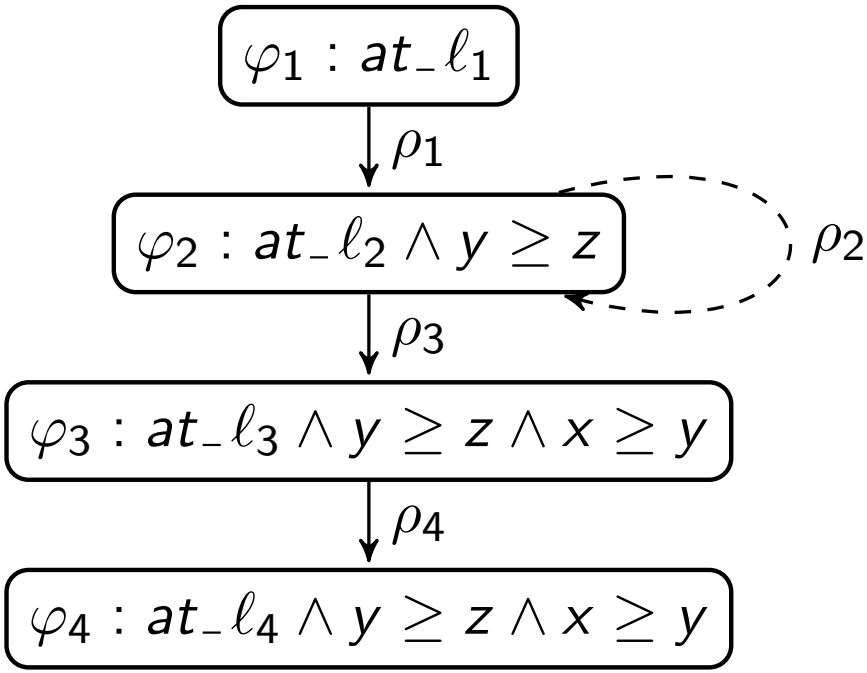
$$\alpha(\varphi) = \bigwedge \{p \in \mathit{Preds} \mid \varphi \models p\}$$

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

algorithm AbstReach for given set of predicates *Preds*

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 $\varphi' \notin \mathit{ReachStates}^{\#}$ **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

Abstract Reachability Graph



$$\varphi_1 = \alpha(\varphi_{init})$$
$$\varphi_2 = post^{\#}(\varphi_1, \rho_1)$$
$$post^{\#}(\varphi_2, \rho_2) \models \varphi_2$$
$$\varphi_3 = post^{\#}(\varphi_2, \rho_3)$$
$$\varphi_4 = post^{\#}(\varphi_3, \rho_4)$$

- $Preds = \{false, at\_\ell_1, \ldots, at\_\ell_5, y \geq z, x \geq y\}$
- nodes $\varphi_1, \ldots, \varphi_4 \in ReachStates^{\#}$
- labeled edges $\in$ *Parent*
- dotted edge : entailment relation (here, $post^{\#}(\varphi_2, \rho_2) \models \varphi_2$)

abstraction function $\alpha(\varphi)$ is a closure operator

- monotonicity

$$\varphi_1 \models \varphi_2 \quad \text{implies} \quad \alpha(\varphi_1) \models \alpha(\varphi_2)$$

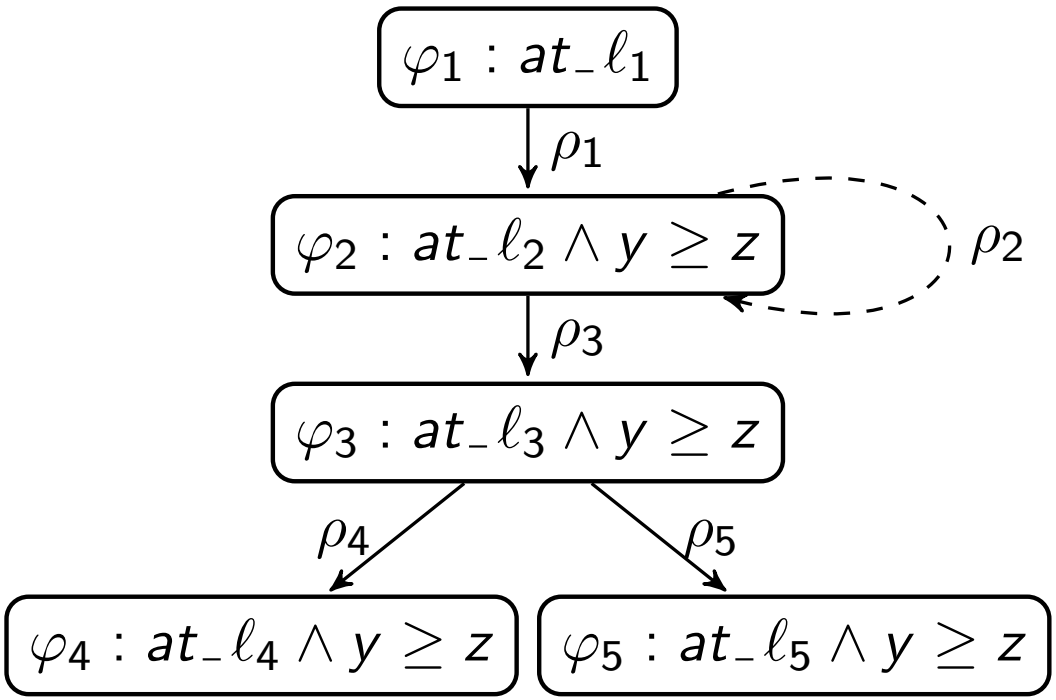
- idempotency

$$\alpha(\alpha(\varphi_1)) = \alpha(\varphi_1)$$

- extensiveness

$$\varphi_1 \models \alpha(\varphi_1)$$

Abstract reachability computation without predicate $x \geq y$



$$\varphi_1 = \alpha(\varphi_{init})$$
$$\varphi_2 = post^{\#}(\varphi_1, \rho_1)$$
$$post^{\#}(\varphi_2, \rho_2) \models \varphi_2$$
$$\varphi_3 = post^{\#}(\varphi_2, \rho_3)$$
$$\varphi_4 = post^{\#}(\varphi_3, \rho_4)$$
$$\varphi_5 = post^{\#}(\varphi_3, \rho_5)$$

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

next . . .

- approach for analysing counterexample computed by AbstReach
- algorithms MakePath, FeasiblePath, and RefinePath
- overall algorithms AbstRefineLoop

algorithm AbstRefineLoop

- compute set of abstract states $ReachStates^{\#}$
  using an abstraction defined by set of predicates *Preds*
  (initially *Preds* = empty)
- if set of error states disjoint from every abstract state: stop
- otherwise, take abstract state ψ in $ReachStates^{\#}$ that overlaps with set of error states

  refinement is only possible if overlap is caused by imprecision
- construct *path*, sequence of program transitions leading to ψ
- analyze *path* using FeasiblePath
- if *path* feasible: stop
- otherwise (*path* is not feasible), compute a set of predicates that refines the abstraction function and repeat

algorithm AbstRefineLoop

```
    function AbstRefineLoop
    begin
1       Preds := ∅
2       repeat
3           (ReachStates#, Parent) := AbstReach(Preds)
4           if exists ψ ∈ ReachStates# such that ψ ∧ φ_err ⊭ false
5   then
6               path := MakePath(ψ, Parent)
7               if FeasiblePath(path) then
8                   return "counterexample path: path "
9               else
10                  Preds := RefinePath(path) ∪ Preds
11          else
                return "program is correct"
    end.
```

counterexample path

- abstract post computation

$$\varphi_5 = post^{\#}(post^{\#}(post^{\#}(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5)$$

- counterexample path in graph formed by *Parent* relation = sequence of edge labels

$$\rho_1,\ \rho_3,\ \rho_5$$

analysis of counterexample path

- apply concrete post instead of abstract post

$$
\begin{aligned}
& \mathit{post}(\mathit{post}(\mathit{post}(\varphi_{\mathit{init}}, \rho_1), \rho_3), \rho_5) \\
= \; & \mathit{post}(\mathit{post}(\mathit{at\_}\ell_2 \wedge y \geq z, \rho_3), \rho_5) \\
= \; & \mathit{post}(\mathit{at\_}\ell_3 \wedge y \geq z \wedge x \geq y, \rho_5) \\
= \; & \mathit{false} \ .
\end{aligned}
$$

- executing the program transitions ρ_1, ρ_3, and ρ_5 in sequence is not *feasible*

refinement of abstraction

- add more predicates to *Preds* such that the new abstraction function α and the new abstract post $post^{\#}$ exclude the counterexample path, meaning:

$$post^{\#}(post^{\#}(post^{\#}(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5) \wedge \varphi_{err} \models false \ .$$

over-approximation along counterexample path

- compute sets of states $\psi_1, \ldots, \psi_4$ such that

$$\begin{aligned} \varphi_{\mathit{init}} &\models \psi_1 \\ \mathit{post}(\psi_1, \rho_1) &\models \psi_2 \\ \mathit{post}(\psi_2, \rho_3) &\models \psi_3 \\ \mathit{post}(\psi_3, \rho_5) &\models \psi_4 \\ \psi_4 \wedge \varphi_{\mathit{err}} &\models \mathit{false} \end{aligned}$$

- add predicates to *Preds* such that $\psi_1, \ldots, \psi_4$ can be expressed (as conjunctions of predicates in new set *Preds*),
- *progress*: the new abstraction is sufficiently precise to exclude the counterexample path ρ_1, ρ_3, ρ_5

$$\mathit{post}^{\#}(\mathit{post}^{\#}(\mathit{post}^{\#}(\alpha(\varphi_{\mathit{init}}), \rho_1), \rho_3), \rho_5) \wedge \varphi_{\mathit{err}} \models \mathit{false} \ .$$

progress

if predicates can express $\psi_1, \ldots, \psi_4$ such that

$$\begin{aligned}
&\varphi_{init} \models \psi_1 \\
&\mathit{post}(\psi_1, \rho_1) \models \psi_2 \\
&\mathit{post}(\psi_2, \rho_3) \models \psi_3 \\
&\mathit{post}(\psi_3, \rho_5) \models \psi_4 \\
&\psi_4 \wedge \varphi_{err} \models \mathit{false}
\end{aligned}$$

then

$$\begin{aligned}
&\alpha(\varphi_{init}) \models \psi_1 \\
&\mathit{post}^{\#}(\psi_1, \rho_1) \models \psi_2 \\
&\mathit{post}^{\#}(\psi_2, \rho_3) \models \psi_3 \\
&\mathit{post}^{\#}(\psi_3, \rho_5) \models \psi_4 \\
&\psi_4 \wedge \varphi_{err} \models \mathit{false}
\end{aligned}$$

and thus

$$\mathit{post}^{\#}(\mathit{post}^{\#}(\mathit{post}^{\#}(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5) \wedge \varphi_{err} \models \mathit{false} \ .$$

path computation

function MakePath
input
ψ - reachable abstract state
Parent - predecessor relation
begin
1 *path* := empty sequence
2 φ' := ψ
3 **while** exist φ and ρ such that $(\varphi, \rho, \varphi') \in$ *Parent* **do**
4 *path* := ρ . *path*
5 φ' := φ
6 **return** *path*
end

path computation

- input: rechable abstract state ψ + *Parent* relation
- view *Parent* as a tree where ψ occurs as a node
- output: sequence of program transitions that labels the tree edges on path from root to ψ
- sequence is constructed iteratively by a backward traversal starting from the input node
- variable *path* keeps track of the construction
- in example, call $\textsc{MakePath}(\varphi_5, \mathit{Parent})$
- *path*, initially empty, is extended with transitions ρ_5, ρ_3, ρ_1
- corresponding edges: $(\varphi_3, \rho_5, \varphi_5)$, $(\varphi_2, \rho_3, \varphi_3)$, $(\varphi_1, \rho_1, \varphi_1)$
- output: $\mathit{path} = \rho_1\rho_3\rho_5$

feasibility of a path

function FeasiblePath
input
$\rho_1 \ldots \rho_n$ - path
begin
1 $\varphi := post(\varphi_{init}, \rho_1 \circ \ldots \circ \rho_n)$
2 **if** $\varphi \wedge \varphi_{err} \not\models false$ **then**
3 **return** *true*
4 **else**
5 **return** *false*
end

feasibility of a path

- input: sequence of program transitions $\rho_1 \ldots \rho_n$
- checks if there is a computation that produced by this sequence
- check uses the post-condition function and the relational composition of transition
- apply FeasiblePath on example path $\rho_1 \rho_3 \rho_5$
- relational composition of transitions yields

$$\rho_1 \circ \rho_3 \circ \rho_5 = \mathit{false} \ .$$

- FeasiblePath sets φ to *false* and then returns *false*

counterexample-guided discovery of predicates

function RefinePath
input
$\rho_1 \dots \rho_n$ - path
begin
1 $\varphi_0, \dots, \varphi_n$:= compute such that
2 $(\varphi_{init} \models \varphi_0) \wedge$
3 $(post(\varphi_0, \rho_1) \models \varphi_1) \wedge \dots \wedge (post(\varphi_{n-1}, \rho_n) \models \varphi_n) \wedge$
4 $(\varphi_n \wedge \varphi_{err} \models false)$
5 **return** $\{\varphi_0, \dots, \varphi_n\}$
end

- omitted: particular algorithm for finding $\varphi_0, \dots, \varphi_n$

counterexample guided discovery of predicates

- input: sequence of program transitions $\rho_1 \ldots \rho_n$
- output: sets of states $\varphi_0, \ldots, \varphi_n$ such that
  - $\varphi_{init} \models \varphi_0$
  - $post(\varphi_{i-1}, \rho_i) \models \varphi_i$
  - $\varphi_n \wedge \varphi_{err} \models false$ for $i \in 1..n$
- if $\varphi_0, \ldots, \varphi_n$ are added to *Preds*
  then the resulting α and $post^{\#}$ guarantee that

$$\begin{aligned} &\alpha(\varphi_{init}) \models \varphi_0 \\ &post^{\#}(\varphi_0, \rho_1) \models \varphi_1 \\ &\ldots \\ &post^{\#}(\varphi_{n-1}, \rho_n) \models \varphi_n \\ &\varphi_n \wedge \varphi_{err} \models false\ . \end{aligned}$$

- in example, application of RefinePath on $\rho_1\rho_3\rho_5$ yields sequence of sets of states $\psi_1, \ldots, \psi_4$

next . . .

- algorithm for counterexample-guided abstraction refinement
- put together all building blocks into an algorithm AbstRefineLoop that verifies safety using predicate abstraction and counterexample guided refinement

predicate abstraction and refinement loop

```
    function AbstRefineLoop
    begin
1       Preds := ∅
2       repeat
3           (ReachStates#, Parent) := AbstReach(Preds)
4           if exists ψ ∈ ReachStates# such that ψ ∧ φ_err ⊭ false
5   then
6               path := MakePath(ψ, Parent)
7               if FeasiblePath(path) then
8                   return "counterexample path: path "
9               else
10                  Preds := RefinePath(path) ∪ Preds
11          else
                return "program is correct"
    end.
```

algorithm AbstRefineLoop

- input: program, output: proof or counterexample
- compute $\varphi^{\#}_{\mathit{reach}}$ using an abstraction defined wrt. set of predicates *Preds* (initially empty)
- over-approximation $\varphi^{\#}_{\mathit{reach}}$: set of formulas $\mathit{ReachStates}^{\#}$ where each formula represents a set of states
- if set of error states disjoint from over-approximation: stop
- otherwise, consider a formula ψ in $\mathit{ReachStates}^{\#}$ that witnesses overlap with error states
- refinement is only possible if overlap is caused by imprecision
- construct *path*, sequence of program transitions leading to ψ
- analyze *path* using FeasiblePath
- if *path* feasible: stop
- otherwise (*path* is not feasible), compute a set of predicates that refines the abstraction function

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$

  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$

  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$

  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$

  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$
- if $\phi \in D^{\#}$ inductive then $\varphi^{\#}_{reach} \sqsubseteq \phi$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$
  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$
- if $\phi \in D^{\#}$ inductive then $\varphi^{\#}_{reach} \sqsubseteq \phi$
- proof by induction on index of disjunct in $\varphi^{\#}_{reach}$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$
  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$
- if $\phi \in D^{\#}$ inductive then $\varphi^{\#}_{reach} \sqsubseteq \phi$
- proof by induction on index of disjunct in $\varphi^{\#}_{reach}$
- $(i = 0)$
  $\varphi_{init} \sqsubseteq \phi$ implies $\alpha(\varphi_{init}) \sqsubseteq \alpha(\phi) = \phi$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$
  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$
- if $\phi \in D^{\#}$ inductive then $\varphi^{\#}_{reach} \sqsubseteq \phi$
- proof by induction on index of disjunct in $\varphi^{\#}_{reach}$
- $(i = 0)$
  $\varphi_{init} \sqsubseteq \phi$ implies $\alpha(\varphi_{init}) \sqsubseteq \alpha(\phi) = \phi$
- $(i \rightarrow i + 1)$
  $\psi \sqsubseteq \phi$ implies $post(\psi) \sqsubseteq post(\phi) \sqsubseteq \phi$

$\varphi^{\#}_{reach}$ stronger than every inductive property expressible in *Preds*

- abstract domain $D^{\#}$ with partial ordering $\sqsubseteq$
- best abstraction: $\alpha(\varphi) = \bigwedge\{\psi \in D^{\#} \mid \varphi \sqsubseteq \psi\}$
  $\psi \in D^{\#}$ implies $\alpha(\psi) = \psi$
- best abstract post: $post^{\#}(\psi) = \alpha(post(\psi))$
- $\varphi^{\#}_{reach} = \bigvee_{i \geq 0} (post^{\#})^{i}(\alpha(\varphi_{init}))$
- ϕ is inductive $\equiv$ $\varphi_{init} \cup post(\phi) \sqsubseteq \phi$
- if $\phi \in D^{\#}$ inductive then $\varphi^{\#}_{reach} \sqsubseteq \phi$
- proof by induction on index of disjunct in $\varphi^{\#}_{reach}$
- $(i = 0)$
  $\varphi_{init} \sqsubseteq \phi$ implies $\alpha(\varphi_{init}) \sqsubseteq \alpha(\phi) = \phi$
- $(i \rightarrow i + 1)$
  $\psi \sqsubseteq \phi$ implies $post(\psi) \sqsubseteq post(\phi) \sqsubseteq \phi$
  which implies $post^{\#}(\psi) \sqsubseteq \alpha(\phi) = \phi$