**Algorithm AbstRefineLoop**

- compute set of abstract states $ReachStates^#$
  using an abstraction defined by set of predicates $Preds$
  (initially $Preds = \emptyset$)
- if set of error states disjoint from every abstract state: stop
- otherwise, take abstract state $\psi$ 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 $\psi$
- 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
  Preds := ∅
  repeat
    (ReachStates#, Parent) := AbstReach(Preds)
    if exists ψ ∈ ReachStates# such that ψ ∧ φ_{err} ∉ false
      then
        path := MakePath(ψ, Parent)
        if FeasiblePath(path) then
          return “counterexample path: path”
        else
          Preds := RefinePath(path) ∪ Preds
        end
      else
        return “program is correct”
    end
  end
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 \textit{Parent} relation
  \( \Rightarrow \) sequence of edge labels

\[ \rho_1, \rho_3, \rho_5 \]
analysis of counterexample path

- apply concrete post instead of abstract post

\[
\text{post}(\text{post}(\text{post}(\varphi_{\text{init}}, \rho_1), \rho_3), \rho_5) \\
= \text{post}(\text{post}(\text{at}_2 \land y \geq z, \rho_3), \rho_5) \\
= \text{post}(\text{at}_3 \land y \geq z \land x \geq y, \rho_5) \\
= \text{false}. 
\]

- executing the program transitions \(\rho_1, \rho_3,\) and \(\rho_5\) in sequence is not feasible
refinement of abstraction

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

$$post^\#(post^\#(post^\#(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5) \land \varphi_{err} \models false.$$
over-approximation along counterexample path

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

\[
\begin{align*}
\varphi_{init} &\models \psi_1 \\
post(\psi_1, \rho_1) &\models \psi_2 \\
post(\psi_2, \rho_3) &\models \psi_3 \\
post(\psi_3, \rho_5) &\models \psi_4 \\
\psi_4 \land \varphi_{err} &\models false
\end{align*}
\]

- 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 $\rho_1, \rho_3, \rho_5$

\[
post#(post#(post#(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5) \land \varphi_{err} \models false .
\]
progress

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

$$
\begin{align*}
\varphi_{init} & \models \psi_1 \\
post(\psi_1, \rho_1) & \models \psi_2 \\
post(\psi_2, \rho_3) & \models \psi_3 \\
post(\psi_3, \rho_5) & \models \psi_4 \\
\psi_4 \land \varphi_{err} & \models false
\end{align*}
$$

then

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

and thus

$$
post^\#(post^\#(post^\#(\alpha(\varphi_{init}), \rho_1), \rho_3), \rho_5) \land \varphi_{err} \models false
$$
function MakePath
input
\( \psi \) - reachable abstract state
\( Parent \) - predecessor relation
begin
1 \hspace{1em} path := empty sequence
2 \hspace{1em} \psi' := \psi
3 \hspace{1em} \textbf{while} exist \( \varphi \) and \( \rho \) such that \( (\varphi, \rho, \varphi') \in Parent \) \textbf{do}
4 \hspace{2em} path := \rho \cdot path
5 \hspace{2em} \psi' := \varphi
6 \hspace{2em} \textbf{return} path
end
path computation

- input: reachable abstract state $\psi + Parent$ relation
- view $Parent$ as a tree where $\psi$ occurs as a node
- output: sequence of program transitions that labels the tree edges on path from root to $\psi$
- sequence is constructed iteratively by a backward traversal starting from the input node
- variable $path$ keeps track of the construction
- in example, call $\text{MAKEPath}(\varphi_5, Parent)$
- $path$, initially empty, is extended with transitions $\rho_5, \rho_3, \rho_1$
- corresponding edges: $(\varphi_3, \rho_5, \varphi_5), (\varphi_2, \rho_3, \varphi_3), (\varphi_1, \rho_1, \varphi_1)$
- output: $path = \rho_1\rho_3\rho_5$
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 \( \varphi \land \varphi_{err} \not\models false \) 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 $\text{FeasiblePath}$ on example path $\rho_1\rho_3\rho_5$
- relational composition of transitions yields
  $$\rho_1 \circ \rho_3 \circ \rho_5 = false.$$  
- $\text{FeasiblePath}$ sets $\varphi$ to $false$ and then returns $false$
counterexample-guided discovery of predicates

\[
\text{function } \text{RefinePath} \\
\text{input} \\
\rho_1 \ldots \rho_n \text{ - path} \\
\text{begin} \\
1 \quad \varphi_0, \ldots, \varphi_n := \text{compute such that} \\
2 \quad (\varphi_{\text{init}} \models \varphi_0) \land \\
3 \quad (\text{post}(\varphi_0, \rho_1) \models \varphi_1) \land \ldots \land (\text{post}(\varphi_{n-1}, \rho_n) \models \varphi_n) \land \\
4 \quad (\varphi_n \land \varphi_{\text{err}} \models \text{false}) \\
5 \quad \text{return } \{\varphi_0, \ldots, \varphi_n\} \\
\text{end}
\]

- omitted: particular algorithm for finding $\varphi_0, \ldots, \varphi_n$
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
  - $\varphi_{init} \models \varphi_0$
  - $\text{post}(\varphi_{i-1}, \rho_i) \models \varphi_i$
  - $\varphi_n \land \varphi_{err} \models \text{false}$ for $i \in 1..n$
- if $\varphi_0, \ldots, \varphi_n$ are added to $P\text{reds}$ then the resulting $\alpha$ and $\text{post}^\#$ guarantee that
  \[
  \alpha(\varphi_{init}) \models \varphi_0 \\
  \text{post}^\#(\varphi_0, \rho_1) \models \varphi_1 \\
  \vdots \\
  \text{post}^\#(\varphi_{n-1}, \rho_n) \models \varphi_n \\
  \varphi_n \land \varphi_{err} \models \text{false}.
  \]
- in example, application of $\text{RefinePath}$ 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 \texttt{AbstRefineLoop} that verifies safety using predicate abstraction and counterexample guided refinement
predicate abstraction and refinement loop

function AbstRefineLoop
begin
  Preds := \emptyset
  repeat
    (ReachStates#, Parent) := AbstReach(Preds)
    if exists \( \psi \in \text{ReachStates}^# \) such that \( \psi \land \varphi_{err} \nvdash false \) then
      path := MakePath(\( \psi \), Parent)
      if FeasiblePath(path) then
        return “counterexample path: path”
      else
        Preds := RefinePath(path) \cup Preds
      else
        return “program is correct”
  end.
end.
**algorithm AbstRefineLoop**

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

- abstract domain \( D^{\#} \) with partial ordering \( \sqsubseteq \)
\( \varphi_{\text{reach}} \) stronger than every inductive property expressible in \( Preds \)

- abstract domain \( D# \) with partial ordering \( \sqsubseteq \)
- best abstraction: \( \alpha(\varphi) = \bigwedge\{\psi \in D# \mid \varphi \sqsubseteq \psi\} \)
\( \forall_{\text{reach}} \) stronger than every inductive property expressible in \( \text{Preds} \)

- abstract domain \( D^\# \) with partial ordering \( \sqsubseteq \)
- best abstraction: \( \alpha(\varphi) = \bigwedge \{ \psi \in D^\# \mid \varphi \sqsubseteq \psi \} \)
  \( \psi \in D^\# \) implies \( \alpha(\psi) = \psi \)
$\varphi^\#_{\text{reach}}$ stronger than every inductive property expressible in $\text{Preds}$

- 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: $\text{post}^\#(\psi) = \alpha(\text{post}(\psi))$
- $\varphi^\#_{\text{reach}} = \bigvee_{i \geq 0} (\text{post}^\#)^i(\alpha(\varphi_{\text{init}}))$
\( \varphi_{\text{reach}} \) stronger than every inductive property expressible in \( \text{Preds} \)

- 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: \( \text{post}^\#(\psi) = \alpha(\text{post}(\psi)) \)
- \( \varphi_{\text{reach}} = \bigvee_{i \geq 0} (\text{post}^\#)^i(\alpha(\varphi_{\text{init}})) \)
- \( \phi \) is inductive \( \equiv \varphi_{\text{init}} \cup \text{post}(\phi) \sqsubseteq \phi \)
\( \varphi_{reach} \) stronger than every inductive property expressible in \( \text{Preds} \)

- 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: \( \text{post}^\#(\psi) = \alpha(\text{post}(\psi)) \)
- \( \varphi_{reach} = \bigvee_{i \geq 0} (\text{post}^\#)^i(\alpha(\varphi_{init})) \)
- \( \phi \) is inductive \( \equiv \) \( \varphi_{init} \cup \text{post}(\phi) \sqsubseteq \phi \)
- 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: $post^\#(\psi) = \alpha(post(\psi))$
- $\varphi_{reach} = \bigvee_{i \geq 0} (post^\#)^i(\alpha(\varphi_{init}))$
- $\phi$ 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 $\varphi_{reach}^\#$
\( \varphi_{\text{reach}} \) stronger than every inductive property expressible in \( \text{Preds} \)

- 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: \( post^\#(\psi) = \alpha(post(\psi)) \)
- \( \varphi^\#_{\text{reach}} = \bigvee_{i \geq 0} (post^\#)^i(\alpha(\varphi_{\text{init}})) \)
- \( \phi \) is inductive \( \equiv \varphi_{\text{init}} \cup post(\phi) \sqsubseteq \phi \)
- if \( \phi \in D^\# \) inductive then \( \varphi^\#_{\text{reach}} \sqsubseteq \phi \)
- proof by induction on index of disjunct in \( \varphi^\#_{\text{reach}} \)
- \( (i = 0) \)
  \( \varphi_{\text{init}} \sqsubseteq \phi \) implies \( \alpha(\varphi_{\text{init}}) \sqsubseteq \alpha(\phi) = \phi \)
\( \varphi_{\text{reach}} \) stronger than every inductive property expressible in \( \text{Preds} \)

- 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: \( \text{post}^\#(\psi) = \alpha(\text{post}(\psi)) \)
- \( \varphi_{\text{reach}} = \bigvee_{i \geq 0} (\text{post}^\#)^i(\alpha(\varphi_{\text{init}})) \)
- \( \phi \) is inductive \( \equiv \varphi_{\text{init}} \cup \text{post}(\phi) \sqsubseteq \phi \)
- if \( \phi \in D^\# \) inductive then \( \varphi_{\text{reach}}^\# \sqsubseteq \phi \)
- proof by induction on index of disjunct in \( \varphi_{\text{reach}}^\# \)
  - \( (i = 0) \)
    \( \varphi_{\text{init}} \sqsubseteq \phi \) implies \( \alpha(\varphi_{\text{init}}) \sqsubseteq \alpha(\phi) = \phi \)
  - \( (i \rightarrow i + 1) \)
    \( \psi \sqsubseteq \phi \) implies \( \text{post}(\psi) \sqsubseteq \text{post}(\phi) \sqsubseteq \phi \)
\( \varphi_{\text{reach}} \) stronger than every inductive property expressible in \( \text{Preds} \)

- 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: \( \text{post}^\#(\psi) = \alpha(\text{post}(\psi)) \)
- \( \varphi^\#_{\text{reach}} = \bigvee_{i \geq 0} (\text{post}^\#)^i(\alpha(\varphi_{\text{init}})) \)
- \( \phi \) is inductive \( \equiv \varphi_{\text{init}} \cup \text{post}(\phi) \sqsubseteq \phi \)
- if \( \phi \in D^\# \) inductive then \( \varphi^\#_{\text{reach}} \sqsubseteq \phi \)
- proof by induction on index of disjunct in \( \varphi^\#_{\text{reach}} \)
  - \( (i = 0) \)
    \( \varphi_{\text{init}} \sqsubseteq \phi \) implies \( \alpha(\varphi_{\text{init}}) \sqsubseteq \alpha(\phi) = \phi \)
  - \( (i \rightarrow i + 1) \)
    \( \psi \sqsubseteq \phi \) implies \( \text{post}(\psi) \sqsubseteq \text{post}(\phi) \sqsubseteq \phi \)
    which implies \( \text{post}^\#(\psi) \sqsubseteq \alpha(\phi) = \phi \)