algorithm AbstRefineLoop

- compute set of abstract states $\mathit{ReachStates}^{\#}$ 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 $\mathit{ReachStates}^{\#}$ 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
  $\mathit{Preds} := \emptyset$
  repeat
    $(\mathit{ReachStates}^{\#}, \mathit{Parent}) := \textsc{AbstReach}(\mathit{Preds})$
    if exists $\psi \in \mathit{ReachStates}^{\#}$ such that $\psi \wedge \varphi_{err} \not\models \mathit{false}$
then
      $\mathit{path} := \textsc{MakePath}(\psi, \mathit{Parent})$
      if $\textsc{FeasiblePath}(\mathit{path})$ **then**
        return "counterexample path: *path* "
      else
        $\mathit{Preds} := \textsc{RefinePath}(\mathit{path}) \cup \mathit{Preds}$
    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

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

- 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

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

- 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

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

progress

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

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

then

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

and thus

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

path computation

```
function MakePath
input
    ψ - reachable abstract state
    Parent - predecessor relation
begin
    path := empty sequence
    φ' := ψ
    while exist φ and ρ such that (φ, ρ, φ') ∈ Parent do
        path := ρ . path
        φ' := φ
    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
  $\varphi \;:=\; post(\varphi_{init}, \rho_1 \circ \ldots \circ \rho_n)$
  if $\varphi \wedge \varphi_{err} \not\models \mathit{false}$ **then**
    return *true*
  else
    return *false*
end

feasibility of a path

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

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

- $\textsc{FeasiblePath}$ sets φ to *false* and then returns *false*

counterexample-guided discovery of predicates

function $\textsc{RefinePath}$
input
$\rho_1 \ldots \rho_n$ - path
begin
$\varphi_0, \ldots, \varphi_n$:= compute such that
$(\varphi_{init} \models \varphi_0) \land$
$(post(\varphi_0, \rho_1) \models \varphi_1) \land \ldots \land (post(\varphi_{n-1}, \rho_n) \models \varphi_n) \land$
$(\varphi_n \land \varphi_{err} \models false)$
return $\{\varphi_0, \ldots, \varphi_n\}$
end

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

counterexample guided discovery of predicates

- input: sequence of program transitions $\rho_1 \dots \rho_n$
- output: sets of states $\varphi_0, \dots, \varphi_n$ 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 $\varphi_0, \dots, \varphi_n$ are added to *Preds*
  then the resulting α and $post^\#$ guarantee that

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

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

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
  Preds := $\emptyset$
  repeat
    (*ReachStates*$^{\#}$, *Parent*) := AbstReach(*Preds*)
    if exists $\psi \in$ *ReachStates*$^{\#}$ such that $\psi \wedge \varphi_{err} \not\models$ *false*
then
      path := MakePath(ψ, *Parent*)
      if FeasiblePath(*path*) **then**
        return "counterexample path: *path* "
      else
        Preds := RefinePath(*path*) $\cup$ *Preds*
    else
      return "program is correct"
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 $\mathit{ReachStates}^{\#}$ 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 $\alpha(\varphi_{init}) \sqsubseteq \alpha(\phi) = \phi$
- $(i \to i+1)$
  $\psi \sqsubseteq \phi$ 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 $post^{\#}(\psi) \sqsubseteq \alpha(\phi) = \phi$