CS766: Analysis of concurrent programs (first half) 2023

Lecture 20: Counterexample guided abstraction refinement (CEGAR)

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2023-01-18

Limitations of symbolic model checking

- Too precise
- Often does not scale!
- Approximations like BMC or concolic testing have sever limitations

Let us bring back abstraction!

Topic 20.1

Recall: abstract domain and abstract post

Recall: Abstract domain

Definition 20.1

Concrete *objects of analysis* *or domain* — $C = \mathfrak{p}(\mathbb{Q}^V)$

- *not all sets are concisely representable in computer*
- *too (infinitely) many of them*

Definition 20.2

Abstract domain — only simple to represent sets $D \subseteq C$

- D *should allow efficient algorithms for desired operations*
- *far fewer, but possibly infinitely many*
- *Sets in* $C \setminus D$ *are not precisely representable in* D

Definition 20.3

An abstraction function $\alpha : C \to D$ *maps each set* $c \in C$ *to* $\alpha(c)$.

Definition 20.4

A concretization function $\gamma : D \to C$ *maps each set* $d \in D$ *to* d.

Recall: Example: abstraction – intervals

Example 20.1

Let us assume $V = \{\texttt{x}\}$

Consider $D = \{\bot, \top\} \cup \{[a, b] | a, b \in \mathbb{Q}\}$.

Ordering among elements of D are defined as follows:
$\bot \sqsubseteq [a, b] \sqsubseteq \top$ *and* $[a_1, b_1] \sqsubseteq [a_2, b_2] \Leftrightarrow a_2 \leq a_1 \wedge b_1 \leq b_2$

Let $\alpha(c) \triangleq [inf(c), sup(c)]$ *and* $\gamma([a, b]) \triangleq [a, b]$

Exercise 20.1

Give the following value

- $\alpha(\{0, 3, 5\}) =$
- $\alpha((0, 3)) =$
- $\alpha([0, 3] \cup [5, 6]) =$
- $\alpha(\{1/x | x \geq 1\}) =$

Abstract operations

Let us suppose we have the following abstract domain

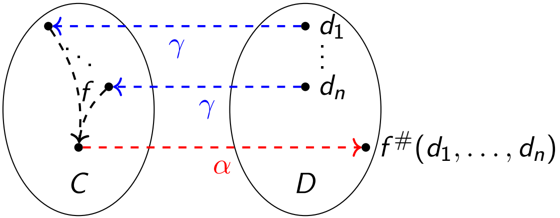
$$(C, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (D, \sqsubseteq).$$

Let us suppose we also have a function $f : C^n \to C$ in concrete domain C.

Definition 20.5

We define an abstract operation $f^{\#} : D^n \to D$ as follows

$$f^{\#}(d_1, \ldots, d_n) = \alpha \circ f(\gamma(d_1), \ldots, \gamma(d_n))$$



Example: abstract operation

We use f, α, and γ to implement $f^{\#}$. For example,

- We may implement $\sqcup$ as follows

$$x \sqcup y = \alpha(\gamma(x) \cup \gamma(y))$$

- We may implement $\sqcap$ as follows

$$x \sqcap y = \alpha(\gamma(x) \cap \gamma(y))$$

Example 20.2

Consider interval domain. Let us compute $[0,3] \sqcup [8,11]$.

- $[0,3] \sqcup [8,11] = \alpha(\gamma([0,3]) \cup \gamma([8,11])) = \alpha([0,3] \cup [8,11]) = [0,11]$

Commentary: The $\sqcup$ computation may look a simple thing made complex. However, the above captures the idea that the function calculation

Abstract strongest post

Recall from earlier lecture, we discussed abstract post. Now we have the formal definition.

$$sp^{\#}(d, \rho) = \alpha \circ sp(\gamma(d), \rho)$$

Example 20.3 (Reminder)

Recall the following abstraction function

$$wideOne(X) = \{n+1, n | n \in X\}$$

We defined the following abstract post

$$sp^{\#}(F, \rho) = \underbrace{wideOne}_{\alpha}(sp(\underbrace{F}_{\gamma \text{ is identity}}, \rho))$$

Topic 20.2

Abstract model checking

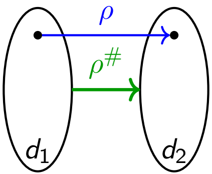
Abstract program

Definition 20.6

Let us consider a finite abstraction D and a program $P = (V, L, \ell_0, \ell_e, E)$. An abstract program $P^\# = \textsc{Abstract}(P, D)$ is $(V, L, \ell_0, \ell_e, E^\#)$ where $E^\#$ is defined as follows.

If $(\ell, \rho, \ell') \in E$ then $(\ell, \rho^\#, \ell') \in E^\#$, where

$$\rho^\# = \{\gamma(d) \times \gamma(d') | d' = sp^\#(d, \rho)\}.$$



We assume D and P allow $\rho^\#$ to be easily representable in a computer.

Properties of abstract programs

Theorem 20.1

$\forall d \in D \exists d' \in D.\ sp(\gamma(d), \rho^{\#}) = \gamma(d')$

In other words, the reachable states of the abstract programs are representable in D.

Theorem 20.2

If $P^{\#}$ is safe then P is safe.

Just analyze the abstract program.

Example : abstract edges

Example 20.4

Consider the following edge and sign abstraction $D = \{\top, -, 0, +, \bot\}$.

$$\rho_1 = (x' = 1)$$

Let us build abstract edge.

- $sp^{\#}(+, \rho_1) = +$
- $sp^{\#}(0, \rho_1) = +$
- $sp^{\#}(-, \rho_1) = +$
- $sp^{\#}(\top, \rho_1) = +$
- $sp^{\#}(\bot, \rho_1) = \bot$

No need to record pairs that start with $\bot$

$$\rho_1^{\#} = \{(-, +), (0, +), (+, +), (\top, +)\}$$

Exercise 20.2

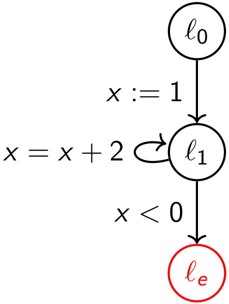
Give abstraction of $\rho_2 = (x' = x + 1)$

Example: abstract program

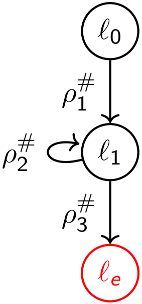
Example 20.5

Consider the following program and sign abstraction $D = \{\top, -, 0, +, \bot\}$.

Program:



Abstract program:



$$\rho_1^\# = \{(-,+),(0,+),(+,+),(\top,+)\}$$

$$\rho_2^\# = \{(-,\top),(0,+),(+,+),(\top,\top)\}$$

$$\rho_3^\# = \{(-,-),(\top,-)\}$$

We have only listed pairs that do not have $\bot$ as second component.

Abstract reachability graph

Since $D = (\sqsubseteq, \top, \bot)$ is finite, symbolic execution of $P^{\#} = \textsc{Abstract}(P, D)$ will produce finitely many symbolic states, which are called abstract states.

Definition 20.7

Abstract reachability graph(ARG) $(reach, R)$ *is the smallest directed graph such that*

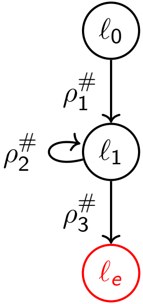
- $reach \subseteq L \times D$
- $(\ell_0, \top) \in reach$
- $((\ell, d), (\ell', d')) \in R$ *if* $\exists(\ell, \rho^{\#}, \ell') \in E^{\#}.\ \ d' = sp(d, \rho^{\#})$

Theorem 20.3

If $\forall d.\ d \neq \bot \wedge (l_e, d) \notin reach$ *then* $P^{\#}$ *is safe.*

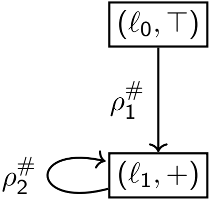
Example: abstract reachability graph

Abstract program:



$$\rho_1^\# = \{(-,+),(0,+),(+,+),(\top,+)\}$$
$$\rho_2^\# = \{(-,+),(-,0),(-,+),(0,+),(+,+),(\top,\top)\}$$
$$\rho_3^\# = \{(-,-),(\top,-)\}$$

Abstract reachability graph:



We are not showing abstract states with $\bot$.

Exercise 20.3

Draw the rest of ARG with $\bot$

Model checking

The word model checking originated from the area of modal logic, where finding a model that satisfies a formula is called model checking.

In our situation, we have a logical statement $P^{\#}$ is not safe

We search for a model of the statement, i.e., a path in the abstract reachability graph that reaches to error location.

If no model found, then $P^{\#}$ is safe.

Abstract reachability graph may be large.

In contrast, abstract interpretation does not construct large objects.

Abstract model checking

Algorithm 20.1: $\textsc{AbstMC}(P^\# = (V, L, \ell_0, \ell_e, E^\#),\ D = (\sqsubseteq, \top, \bot))$

Output: $\textsc{Correct}$ if $P^\#$ is safe, abstract counterexample otherwise

$worklist := \{(\ell_0, \top)\}$; $reach := \emptyset$; $covered, parent : reach \cup worklist \rightarrow reach \cup worklist := \emptyset$;

$path : reach \cup worklist \rightarrow$ (sequences of $E^\#$) $:= \{(\ell_0, \top) \mapsto \epsilon\}$;

while $worklist \neq \emptyset$ **do**

  choose $(\ell, d) \in worklist$; $worklist := worklist \setminus \{(\ell, d)\}$;

  if $d = \bot$ *or* $\exists s \in parent^*((\ell, d)).\ s \in covered$ **then** **continue**;

  if $\ell = \ell_e$ **then** **return** $\textsc{Counterexample}(path(\ell, d))$;

  $reach := reach \cup \{(\ell, d)\}$;

  if $\exists(\ell, d') \in reach - range(covered).\ d \sqsubseteq d'$ **then**

    $covered := covered[(\ell, d') \mapsto (\ell, d)]$ // covered by existing state

  else

    if $\exists(\ell, d') \in reach - range(covered).\ d' \sqsubseteq d$ **then**

      $covered := covered[(\ell, d) \mapsto (\ell, d')]$ // covering existing state

    foreach $(\ell, \rho^\#, \ell') \in E^\#$ **do**

      $d' := sp(d, \rho^\#)$; $worklist := worklist \cup \{(\ell', d')\}$; $parent := parent[(\ell', d') \mapsto (\ell, d)]$;

      $path := path[(\ell', d') \mapsto path(\ell, d).(\ell, \rho^\#, \ell')]$;

return $\textsc{Correct}$

$P^\#$ accessed only once

On the fly abstraction

In $\textsc{AbstMC}$, we only access $P^{\#}$ to compute post operator over d.

This suggests, $\textsc{AbstMC}$ can be implemented in the following two ways.

- Precompute $P^{\#}$ and run $\textsc{AbstMC}$ as presented.
- On the fly construction of $P^{\#}$. We construct transitions of $P^{\#}$ as we need them

Exercise 20.4

Discuss benefits of both the approaches

Finite abstractions

The following abstractions are widely used in modelcheckers

- Cartesian predicate abstraction
- Boolean predicate abstraction

Finite abstraction example : Cartesian predicate abstraction

Cartesian predicate abstraction is defined by a set of predicates $Preds = \{p_1, \ldots, p_n\}$
$C = \mathfrak{p}(\mathbb{Q}^{|V|})$
$D = \bot \cup \mathfrak{p}(Preds)$ // $\emptyset$ represents $\top$
$\bot \sqsubseteq S_1 \sqsubseteq S_2$ if $S_2 \subseteq S_1$
$\alpha(c) = \{p \in P | c \Rightarrow p\}$
$\gamma(S) = \bigwedge S$

Example 20.6

$V = \{\mathrm{x}, \mathrm{y}\}$
$P = \{\mathrm{x} \leq 1, -\mathrm{x} - \mathrm{y} \leq -1, \mathrm{y} \leq 5\}$
$\alpha(\{(0,0)\}) = \{\mathrm{x} \leq 1, \mathrm{y} \leq 5\}$
$\alpha((\mathrm{x}-1)^2 + (\mathrm{y}-3)^2 = 1) = \{-\mathrm{x} - \mathrm{y} \leq -1, \mathrm{y} \leq 5\}$

Representing predicate domain

We represent abstract state as bit vectors.

Example 20.7

Consider $V = \{\mathtt{x}, \mathtt{y}\}$ *and* $P = \{\mathtt{x} \leq 1, -\mathtt{x} - \mathtt{y} \leq -1, \mathtt{y} \leq 5\}$

Let $[101]$ *represent* $\mathtt{x} \leq 1 \land \mathtt{y} \leq 5$

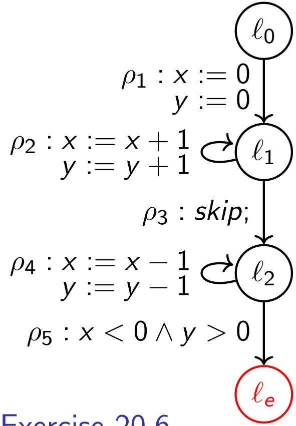
Exercise 20.5

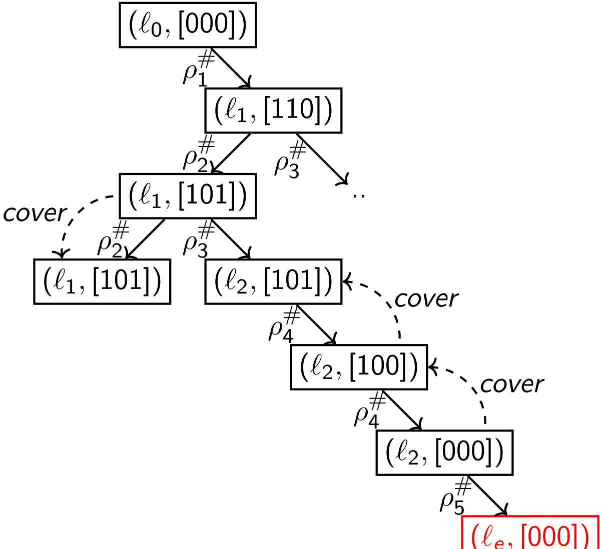
- $[100]$ *represents ...*
- $[000]$ *represent ...*
- *Is* $[100] \sqsubseteq [000]$*?*
- *Is* $[100] \sqsubseteq [001]$*?*
- *Is* $[101] \sqsubseteq [001]$*?*
- *Can we represent false in predicate domain without using special symbol* $\bot$*?*

Example: ARG with Cartesian predicate abstraction

$Preds = \{x \geq 0, y \leq 0, x \geq 1\}$.

Program:





Exercise 20.6

Complete the ARG

Spurious counterexample

$\textsc{AbstMC}(P^{\#}, D)$ may fail to prove $P^{\#}$ correct and return a path $e_1^{\#} \dots e_m^{\#}$, which is called abstract counterexample.

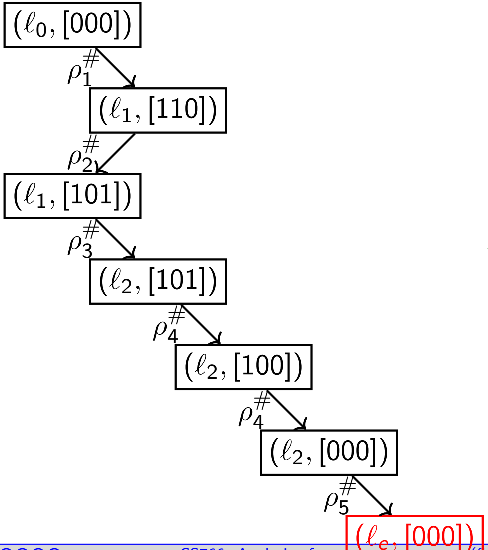
Let $e_1 \dots e_m$ be the corresponding path in P. Now we have two possibilities.

- $e_1 \dots e_m$ is feasible. Then, we have found a bug
- $e_1 \dots e_m$ is not feasible. Then, we call $e_1 \dots e_m$ as spurious counterexample.

We need to fix our abstraction such that we do not get the spurious counter example.

Example : spurious counterexample

Example 20.8



Since we cannot execute $\rho_1\rho_2\rho_3\rho_4\rho_4\rho_5$, the path is a spurious counterexample.

We check the feasibility of the path using satisfiability of path constraints.

Refinement relation

Definition 20.8

Consider abstractions

$$(C, \subseteq) \xleftrightarrows[\alpha_1]{\gamma_1} (D_1, \sqsubseteq_1) \quad \textit{and} \quad (C, \subseteq) \xleftrightarrows[\alpha_1]{\gamma_2} (D_2, \sqsubseteq_2).$$

D_2 refines D_1 if

$$\forall c \in C.\ \gamma_1(\alpha_1(c)) \subseteq \gamma_2(\alpha_2(c))$$

Exercise 20.7

$\gamma_1 \circ \alpha_2$ *is order embedding.*

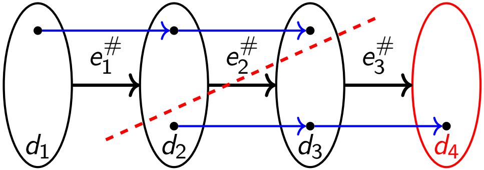
Abstraction refinement

Theorem 20.4

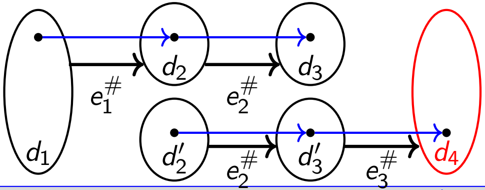
If $\text{ABSTRACT}(P, D_1)$ *exhibits a spurious counterexample then there is an abstraction* D_2 *such that* D_2 *refines* D_1 *and* $\text{ABSTRACT}(P, D_2)$ *does not exhibit the same counter example.*

Proof sketch.

Spurious counterexample:



Refined abstraction:



We say the refinement to D_2 from D_1 ensures progress, i.e., counterexamples are not repeated if ARG is build again with D_2

Refinment Strategy for predicate abstraction

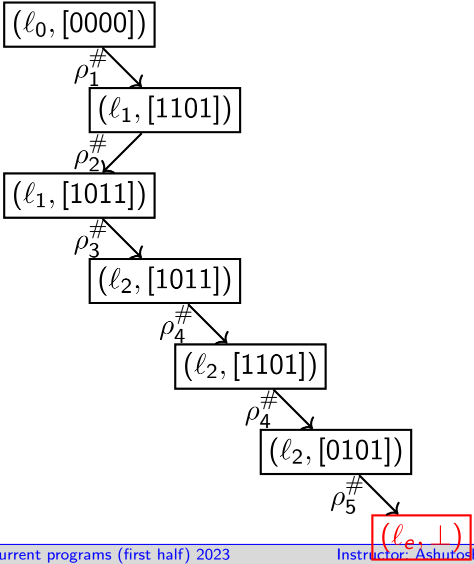
General refinement strategy

Split abstract states such that the spurious counterexample is disconnected.

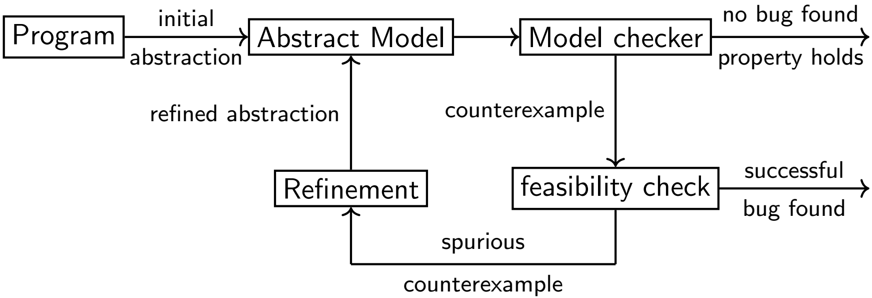
In predicate abstraction, we only need to add more predicates. The new abstraction will certainly be refinement.

Example: refinement

Adding $y \leq -1$ removes the spurious counterexample. $Preds = \{x \geq 0, y \leq 0, x \geq 1, y \leq 1\}$



CEGAR: CounterExample Guided Abstraction Refinement



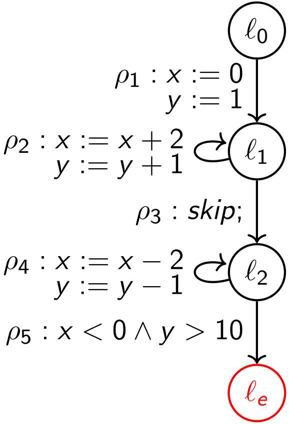
Topic 20.3

Problems

Abstract reachability graph

Exercise 20.8

Choose a set of predicates that will prove the following program correct and show the ARG of the program using the predicates.



CPAchecker

Exercise 20.9

Download CPAchecker: https://cpachecker.sosy-lab.org/
Apply the tool on the following example and report the generated ARG.

```
int x=0; y=0; z=0; w=0;
while( * )) {
  if( * ) {
     x = x+1;
     y = y+100;
  }else if ( * ) {
    if (x >= 4) {
      x = x+1;
      y = y+1;
    }
  }else if (y > 10*w && z >= 100*x) {
     y = -y;
  }
  w = w+1;
  z = z+10;
}
if (x >= 4 && y <= 2)
```

LTL to Bübhi

Exercise 20.10

Convert the following LTL formula into a Büchi automatom

$$\Box\Diamond a \land \Diamond\Box b$$

End of Lecture 20