Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2015-16
Lecture 13

12.1.16
How to define an abstract model:

Given $M$ and $\varphi$, choose

- $S_h$ - a set of abstract states

- $AP$ - a set of atomic propositions that label concrete and abstract states

- $h : S \rightarrow S_h$ - a mapping from $S$ on $S_h$ that satisfies:

  $$h(s) = h(t) \text{ only if } L(s) = L(t)$$

- $h$ is called appropriate w.r.t. $AP$
The abstract model
\[ M_h = (S_h, I_h, R_h, L_h) \]

• \( s_h \in I_h \iff \exists s \in I : h(s) = s_h \)

• \((s_h, t_h) \in R_h \iff \exists s, t \ [ h(s) = s_h \land h(t) = t_h \land (s, t) \in R ] \)

• \( L_h(s_h) = L(s) \) for some \( s \) where \( h(s) = s_h \)

This is an exact abstraction
An approximated abstraction (an approximation)

• $s_h \in I_h \iff \exists s \in I : h(s) = s_h$

• $(s_h, t_h) \in R_h \iff$
  
  $\exists s, t \ [ h(s) = s_h \land h(t) = t_h \land (s, t) \in R ]$

• $L_h$ is as before

Notation:
$M_r$ - reduced (exact)  $M_h$ - approximated
Depending on $h$ and the size of $M$, $M_h$ (i.e. $I_h$, $R_h$) can be built using:

- BDDs or
- SAT solver or
- Theorem prover

We later demonstrate such constructions for specific types of abstractions.
Predicate Abstraction

- Given a program over variables $V$
- **Predicate** $P_i$ is a first-order atomic formula over $V$
  Examples: $x+y < z^2$, $x=5$

- Choose: $AP = \{P_1, ..., P_k\}$ that includes
  - the atomic formulas in the property $\phi$ and
  - conditions in **if**, **while** statements of the program
Predicate Abstraction – Example

while (x ≤ 1) {
    
    ...... 
    
    if (y = 2) { .... } 
    
    ...... 

} 

ϕ = AFG(x > y)

AP = { x > y, x ≤ 1, y = 2 }
Predicate Abstraction

- Labeling of concrete states:

\[ L(s) = \{ P_i \mid s \models P_i \} \]
Example (concrete model)

Program over natural variables $x, y$

$S = \mathbb{N} \times \mathbb{N}$

$AP = \{ P_1, P_2, P_3 \}$ where

$P_1 = x \leq 1$, $P_2 = x > y$, $P_3 = y = 2$

$AP = \{ x \leq 1, x > y, y = 2 \}$

$L((0,0)) = L((1,1)) = L((0,1)) = \{ P_1 \}$

$L((0,2)) = L((1,2)) = \{ P_1, P_3 \}$

$L((2,3)) = \emptyset$
Abstract model - Definition

• Abstract states are defined over Boolean variables \{ B_1,\ldots,B_k \}:
  \[ S_h \subseteq \{ 0,1 \}^k \]

• \( h(s) = s_h \iff \forall 1 \leq j \leq k : [ s |= P_j \iff s_h |= B_j ] \)

• \( L_h(s_h) = \{ P_j \mid s_h |= B_j \} \)
Example (concrete model)

Program over natural variables \( x, y \)

\[ S = N \times N \]

\[ AP = \{ P_1, P_2, P_3 \} \text{ where} \]

\[ P_1 = x \leq 1 \, , \, P_2 = x > y \, , \, P_3 = y = 2 \]

\[ AP = \{ x \leq 1 \, , \, x > y \, , \, y = 2 \} \]

\[ L((0,0)) = L((1,1)) = L((0,1)) = \{ P_1 \} \]

\[ L((0,2)) = L((1,2)) = \{ P_1, P_3 \} \]

\[ L((2,3)) = \emptyset \]
Example – (abstract model)

\[ \mathcal{AP} = \{ P_1 = (x \leq 1), P_2 = (x > y), P_3 = (y = 2) \} \]

\[ S_h \subseteq \{ 0,1 \}^3 \]

\[ h((0,0)) = h((1,1)) = h(0,1)) = (1,0,0) \]
\[ h((0,2)) = h((1,2)) = (1,0,1) \]

No concrete state is mapped to (1,1,1)

\[ L_h((1,0,0)) = \{ P_1 \} \]
\[ L_h((1,0,1)) = \{ P_1, P_3 \} \]

The concrete state and its abstract state are labeled identically
Computing $R_h$ (same example)

$$(s_h, t_h) \in R_h \iff \exists s, t \ [ h(s) = s_h \land h(t) = t_h \land (s,t) \in R ]$$
Computing $R_h$ (same example)

Program with one statement: $x := x+1$

$( (b_1,b_2,b_3) , (b'_1,b'_2,b'_3) ) \in R_h \iff$

$\exists xyx'y' [ \begin{array}{l}
P_1(x,y) \iff b_1 \land \\
P_2(x,y) \iff b_2 \land \\
P_3(x,y) \iff b_3 \land \\
x' = x + 1 \land y' = y \land \\
P_1(x',y') \iff b'_1 \land \\
P_2(x',y') \iff b'_2 \land \\
P_3(x',y') \iff b'_3
\end{array} ] \land h(s) = s_h \land h(t) = t_h$
Logic preservation Theorem

- **Theorem** If $\varphi$ is an ACTL/ACTL* specification over AP, then

$$M_h \models \varphi \Rightarrow M \models \varphi$$

- However, the reverse may not be valid.
Traffic Light Example

Property:
\[ \varphi = \text{AG AF} \rightarrow (\text{state}=\text{red}) \]

Abstraction function \( h \) maps green, yellow to go.

\[ M \models \varphi \iff M_h \models \varphi \]
Traffic Light Example (Cont)

If the abstract model invalidates a specification, the actual model may still satisfy the specification.

- Property:
  \( \varphi = AG \ AF \ (state=\text{red}) \)

- \( M \models \varphi \) but \( M_h \not\models \varphi \)

- Spurious Counterexample:
  \( \langle \text{red}, \text{go}, \text{go}, \ldots \rangle \)
CounterExample-Guided Abstraction-Refinement (CEGAR)
The CEGAR Methodology

1. \( M \) and \( \varphi \)
2. Generate initial abstraction
3. Model check
   - \( M_h \)
   - \( M_h \models \varphi \)
   - \( M_h \not\models \varphi \)
4. Refinement: generate new abstraction
5. Generate counterexample \( T_h \)
6. Check spurious counterexample
   - \( T_h \) is spurious
   - \( T_h \) is not spurious
7. Stop
Refinement versus approximation

Remark:

**Refinement** builds new $S_h$ and $h$, then recomputes $R_h$ and $I_h$ (exact or approximated)

**Approximation** uses the same $S_h$ and $h$ and just adds more transitions and initial states
Generating the Initial Abstraction

- If we use predicate abstraction then predicates are extracted from the program’s control flow and the checked property.

- If we use localization reduction then the un-abstracted variables are those appearing in the predicates above.
Predicate Abstraction - Example

while (true) {
    if (reset == 1) { x=y=0; }
    else if (x<y) { x=x+1; }
    else if (x==y && !(y==2)) { y=y+1; }
    else if (x==y) { x=y=0; }
}

$$\varphi = AF(x==y)$$

$$AP=\{reset==1, \ x<y, \ x==y, \ y==2\}$$
Model Check The Abstract Model

Given the abstract model $M_h$

- If $M_h \not\models \varphi$, then the model checker generates a counterexample trace ($T_h$)
- Most current model checkers generate paths or loops
- Question: is $T_h$ spurious?
Counterexamples

- For $AGp$ it is a path to a state satisfying $\neg p$
- For $AFp$ it is a infinite path represented by a path+loop, where all states satisfy $\neg p$

On the other hand

- For $EFp$ we need to return the whole computation tree (the whole model)
- For $AX(AGp \lor AGq)$ we need to return a computation tree demonstrating $EX(EF\neg p \land EF\neg q)$
Path Counterexample

Assume that we have four abstract states
\[
\{1,2,3\} \leftrightarrow \alpha \quad \{4,5,6\} \leftrightarrow \beta \\
\{7,8,9\} \leftrightarrow \gamma \quad \{10,11,12\} \leftrightarrow \delta
\]

Abstract counterexample \( T_h = \langle \alpha, \beta, \gamma, \delta \rangle \)

\( T_h \) is not spurious, therefore, \( M \models \varphi \)
Spurious Path Counterexample

The concrete states mapped to the failure state are partitioned into 3 sets:

<table>
<thead>
<tr>
<th>states</th>
<th>dead-end</th>
<th>bad</th>
<th>irrelevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>reachable</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>out edges</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

\(T_h\) is spurious
Refining The Abstraction

- **Goal**: refine $h$ so that the dead-end states and bad states do not belong to the same abstract state.

- For this example, two possible solutions.
Refining the abstraction

• Refinement separates dead-end states from bad states, thus, eliminating the spurious transition from $S_{i-1}$ to $S_i$
Completeness of CEGAR

If $M$ is finite

- Our methodology refines the abstraction until either the property is proved or a real counterexample is found

- **Theorem** Given a finite model $M$ and an ACTL* specification $\phi$ whose counterexample is either path or loop, our algorithm will find a model $M_a$ such that

$$M_a \models \phi \iff M \models \phi$$
Conclusion

We presented a framework for **Counterexample Guided Abstraction Refinement (CEGAR)** that

- Automatically constructs an initial abstraction, based on the checked property and the system
- If the abstract system contains a spurious counterexample then the abstraction is automatically refined in order to eliminate the counterexample