Static Program Analysis

Lecture 17: Abstract Interpretation VII (Limits & Improvements of CEGAR)

Winter Semester 2016/17

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-1617/spa/
Schedule of Lectures

Jan 17/19: Interprocedural DFA
Jan 24/26: [no lectures/exercise classes]
Jan 31/Feb 2: Pointer/shape analysis
Feb 7: [no lecture]
Feb 9: Exam preparation
Seminar *Verification and Static Analysis of Software (SS 2017)*

**Topics**
- Pointer and shape analysis
- Advanced model checking techniques
- Analysis of probabilistic programs
- ...

**More information**
https://moves.rwth-aachen.de/teaching/ss-17/vsas/

**Registration**
between January 13 and 29 via
https://www.graphics.rwth-aachen.de/apse/
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Outline of Lecture 17

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

 Reminder: CEGAR

- Start with (coarse) initial abstraction \( A \)
- Property \( \varphi \) satisfied in \( A \)?
  - yes: Verification successful
  - no: Find run violating \( \varphi \)
    - spurious
    - real: Analyze counterexample
      - Error found
- Remove counterexample by refining \( A \)
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Reminder: CEGAR

1. Start with (coarse) initial abstraction $A$
2. Property $\varphi$ satisfied in $A$?
   - yes: Verification successful
   - no: Find run violating $\varphi$
   - spurious: Analyze counterexample
     - real: Error found

Problems:
- How to decide realness of counterexample?
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Reminder: CEGAR

- Start with (coarse) initial abstraction $A$
- Property $\varphi$ satisfied in $A$?
- Find run violating $\varphi$
- Analyze counterexample
- Error found
- Verification successful

Problems:
- How to decide realness of counterexample?
- How to extract new predicates from spurious counterexample?
Abstract Semantics for Predicate Abstraction

**Definition (Execution relation for predicate abstraction)**

If $c \in Cmd$ and $Q \in Abs(P)$, then $\langle c, Q \rangle$ is called an **abstract configuration**. The execution relation for predicate abstraction is defined by the following rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(skip)</td>
<td>$\langle \text{skip}, Q \rangle \Rightarrow \langle \downarrow, Q \rangle$</td>
</tr>
<tr>
<td>(asgn)</td>
<td>$\langle x := a, Q \rangle \Rightarrow \langle \downarrow, \bigcup {Q_\sigma[x \mapsto \text{val}_\sigma(a)] \mid \sigma \models Q } \rangle$</td>
</tr>
<tr>
<td>(seq1)</td>
<td>$\langle c_1, Q \rangle \Rightarrow \langle c'_1, Q' \rangle$ $c'_1 \neq \downarrow$ $\langle c_1, Q \rangle \Rightarrow \langle \downarrow, Q' \rangle$</td>
</tr>
<tr>
<td>(seq2)</td>
<td>$\langle c_1; c_2, Q \rangle \Rightarrow \langle c'_1; c_2, Q' \rangle$ $\langle c_1; c_2, Q \rangle \Rightarrow \langle c_2, Q' \rangle$</td>
</tr>
<tr>
<td>(if1)</td>
<td>$\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, Q \rangle \Rightarrow \langle c_1, Q \land b \rangle$</td>
</tr>
<tr>
<td>(if2)</td>
<td>$\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, Q \rangle \Rightarrow \langle c_2, Q \land \neg b \rangle$</td>
</tr>
<tr>
<td>(wh1)</td>
<td>$\langle \text{while } b \text{ do } c \text{ end}, Q \rangle \Rightarrow \langle c; \text{while } b \text{ do } c \text{ end}, Q \land b \rangle$</td>
</tr>
<tr>
<td>(wh2)</td>
<td>$\langle \text{while } b \text{ do } c \text{ end}, Q \rangle \Rightarrow \langle \downarrow, Q \land \neg b \rangle$</td>
</tr>
</tbody>
</table>
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Properties of Interest

- A certain program location is not reachable (dead code)
- Division by zero is excluded
- The value of \( x \) never becomes negative
- After program termination, the value of \( y \) is even

\[ \Rightarrow \text{All representable as (non-)reachability of “bad locations”} \]
\[ \Rightarrow \text{Counterexample = path to bad locations} \]

Definition (Counterexample)

- A **counterexample** is a sequence of \( k \geq 1 \) abstract transitions of the form

\[
\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle
\]

where
- \( c_0, \ldots, c_k \in \text{Cmd} \) (or \( c_k = \downarrow \))
- \( Q_1, \ldots, Q_k \in \text{Abs}(P) \) with \( Q_k \not\equiv \text{false} \)
- It is called **real** if there exist concrete states \( \sigma_0, \ldots, \sigma_k \in \Sigma \) such that

\[
\forall i \in \{1, \ldots, k\} : \sigma_i \models Q_i \quad \text{and} \quad \langle c_{i-1}, \sigma_{i-1} \rangle \rightarrow \langle c_i, \sigma_i \rangle
\]
- Otherwise it is called **spurious**.
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Elimination of Spurious Counterexamples

Lemma

\[ \langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle \] is a spurious counterexample, there exist Boolean expressions \( b_0, \ldots, b_k \) with \( b_0 \equiv \text{true} \), \( b_k \equiv \text{false} \), and

\[ \forall i \in \{1, \ldots, k\}, \sigma, \sigma' \in \Sigma : \sigma \models b_{i-1} \land \langle c_{i-1}, \sigma \rangle \rightarrow \langle c_i, \sigma' \rangle \Rightarrow \sigma' \models b_i \]

Proof (idea).

Inductive definition of \( b_i \) as strongest postconditions:

1. \( b_0 := \text{true} \)
2. for \( i = 1, \ldots, k \): definition of \( b_i \) depending on \( b_{i-1} \) and on (axiom) transition rule applied in
   \[ \langle c_{i-1}, \cdot \rangle \Rightarrow \langle c_i, \cdot \rangle : \]
   - (skip) \( b_i := b_{i-1} \)
   - (asgn) \( b_i := \exists x'. (b_{i-1}[x \mapsto x'] \land x = a[x \mapsto x']) \)
     (for \( x := a; x' = \text{previous value of } x \))
   - (if1) \( b_i := b_{i-1} \land b \)
   - (if2) \( b_i := b_{i-1} \land \neg b \)
   - (wh1) \( b_i := b_{i-1} \land b \)
   - (wh2) \( b_i := b_{i-1} \land \neg b \)

(yields \( b_k \equiv \text{false} \); by induction on \( k \))
Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Abstraction Refinement

- Using $b_1, \ldots, b_{k-1}$ as computed before, let $P' := P \cup \{p_1, \ldots, p_n\}$ where $p_1, \ldots, p_n$ are the atomic conjuncts occurring in $b_1, \ldots, b_{k-1}$
- Refine $Abs(P)$ to $Abs(P')$

Lemma

*Lemma*

*After refinement, the spurious counterexample*

$$\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle$$

*with $Q_k \not\equiv \text{false}$ does not exist anymore.*

*Proof.*

omitted
Where CEGAR Fails

Outline of Lecture 17

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice
Where CEGAR Fails

Example 17.1

- $c := [x := a]_0^0$
  $[y := b]_1^1$
  while $\neg(x = 0)$ do
    $[x := x - 1]_3^3$
    $[y := y - 1]_4^4$
  end;
  if $[a = b \land \neg(y = 0)]_5^5$ then
    $[\text{skip}]_6^6$
  else
    $[\text{skip}]_7^7$
  end
- **Interesting property:** label 6 unreachable
Where CEGAR Fails

Example 17.1

- \(c := [x := a]^0;\)
  \([y := b]^1;\)
  while \(\neg(x = 0)^2\) do
    \([x := x - 1]^3;\)
    \([y := y - 1]^4\)
  end;
  if \([a = b \land \neg(y = 0)]^5\) then
    \[skip]^6
  else
    \[skip]^7
  end

- **Interesting property:** label 6 unreachable

- **Initial abstraction:** \(P = \emptyset\)
  \(\implies \text{Abs}(P) = \{\text{true, false}\}\)

- **Abstraction refinement:** on the board
### Where CEGAR Fails

**Example 17.1**

- \( c := [x := a]^{0}; \)
  \[ y := b \]^{1};
- while \( \neg(x = 0) \) do
  \[ x := x - 1 \]^{3};
  \[ y := y - 1 \]^{4}
end;
- if \( a = b \land \neg(y = 0) \) then
  \[ \text{skip} \]^{6}
else
  \[ \text{skip} \]^{7}
end

- **Interesting property:** label 6 unreachable
- **Initial abstraction:** \( P = \emptyset \)
  \( \iff \ \text{Abs}(P) = \{ \text{true}, \text{false} \} \)
- **Abstraction refinement:** on the board
- **Observation:** iteration yields predicates of the form
  \[ x = a - k \text{ and } y = b - k \]
for all \( k \in \mathbb{N} \)
- **Actually required:** loop invariant
  \[ a = b \implies x = y \]
but predicate \( x = y \) not generated in CEGAR loop
Craig Interpolation

Outline of Lecture 17

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice
Craig Interpolation

• **Problem:** predicates often unnecessarily complex and involving “irrelevant” variables
• **Idea:** consider only variables that are relevant for previous and future part of execution
Craig Interpolation

Problem: predicates often unnecessarily complex and involving “irrelevant” variables
Idea: consider only variables that are relevant for previous and future part of execution

William Craig (1918–2016)

Definition 17.2 (Craig interpolant)

Let $b_1, b_2 \in BExp$ where $b_1 \models b_2$. A Craig interpolant of $b_1$ and $b_2$ is a formula $b_3 \in BExp$ with $b_1 \models b_3$, $b_3 \models b_2$, and $\text{Var}_{b_3} \subseteq \text{Var}_{b_1} \cap \text{Var}_{b_2}$.
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle$
   (according to Definition 16.1)
**Craig Interpolation**

**Using Craig Interpolants I**

1. Begin with *spurious counterexample* \( \langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle \) (according to Definition 16.1)

2. Construct *strongest postconditions* \( s_0, \ldots, s_k \) with \( s_0 \equiv \text{true}, s_k \equiv \text{false} \) (according to Lemma 16.2)

3. Analogously it is possible to construct *weakest preconditions* \( w_0, \ldots, w_k \) with \( w_0 \equiv \text{true}, w_k \equiv \text{false} \) starting from \( w_k \):
   - \( w_k := \text{false} \)
   - for \( i = 0, \ldots, k - 1 \):
     - definition of \( w_i \) depending on \( w_{i+1} \) and on (axiom) transition rule applied in \( \langle c_i, \cdot \rangle \Rightarrow \langle c_{i+1}, \cdot \rangle 

   \[ \begin{align*}
   ■ & (\text{skip}) \quad w_i := w_{i+1} \\
   ■ & (\text{asgn}) \quad w_i := w_{i+1}[x \mapsto a] \\
   ■ & (\text{if1}) \quad w_i := (w_{i+1} \land b) \lor \neg b \equiv w_{i+1} \lor \neg b \\
   ■ & (\text{if2}) \quad w_i := w_{i+1} \lor b \\
   ■ & (\text{wh1}) \quad w_i := w_{i+1} \lor \neg b \\
   ■ & (\text{wh2}) \quad w_i := w_{i+1} \lor b
   \end{align*} \]

4. Possible to show: \( s_i \models w_i \) for each \( i \in \{0, \ldots, k\} \)

5. For each \( i \in \{0, \ldots, k\} \), choose Craig interpolant \( b_i \) of \( s_i \) and \( w_i \)

6. Refine abstraction by atomic conjuncts occurring in \( b_1, \ldots, b_{k-1} \)

**Remark:** Craig interpolants always exist for first-order formulae (but are not necessarily unique)
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample $\langle c_0, \text{true}\rangle \Rightarrow \langle c_1, Q_1\rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k\rangle$
   (according to Definition 16.1)

2. Construct strongest postconditions $s_0, \ldots, s_k$ with $s_0 \equiv \text{true}$, $s_k \equiv \text{false}$
   (according to Lemma 16.2)

3. Analogously it is possible to construct weakest preconditions $w_0, \ldots, w_k$ with $w_0 \equiv \text{true}$, $w_k \equiv \text{false}$ starting from $w_k$
   i. $w_k := \text{false}$
   ii. for $i = 0, \ldots, k - 1$: definition of $w_i$ depending on $w_{i+1}$ and on (axiom) transition rule applied in $\langle c_i, .\rangle \Rightarrow \langle c_{i+1}, .\rangle$:
      - (skip) $w_i := w_{i+1}$
      - (asgn) $w_i := w_{i+1}[x \mapsto a]$  
      - (if1) $w_i := (w_{i+1} \land b) \lor \neg b \equiv w_{i+1} \lor \neg b$
      - (if2) $w_i := w_{i+1} \lor b$
      - (wh1) $w_i := w_{i+1} \lor \neg b$
      - (wh2) $w_i := w_{i+1} \lor b$

4. Possible to show: $s_i \parallel w_i$ for each $i \in \{0, \ldots, k\}$

5. For each $i \in \{0, \ldots, k\}$, choose Craig interpolant $b_i$ of $s_i$ and $w_i$

6. Refine abstraction by atomic conjuncts occurring in $b_1, \ldots, b_k - 1$

Remark: Craig interpolants always exist for first-order formulae (but are not necessarily unique)
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample \( \langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle \)
   (according to Definition 16.1)

2. Construct strongest postconditions \( s_0, \ldots, s_k \) with \( s_0 \equiv \text{true}, \ s_k \equiv \text{false} \)
   (according to Lemma 16.2)

3. Analogously it is possible to construct weakest preconditions \( w_0, \ldots, w_k \) with \( w_0 \equiv \text{true}, \ w_k \equiv \text{false} \) starting from \( w_k \)
   i. \( w_k := \text{false} \)
   ii. for \( i = 0, \ldots, k - 1 \): definition of \( w_i \) depending on \( w_{i+1} \) and on (axiom) transition rule applied in
   \( \langle c_i, . \rangle \Rightarrow \langle c_{i+1}, . \rangle \):
   - (skip) \( w_i := w_{i+1} \)
   - (asgn) \( w_i := w_{i+1}[x \mapsto a] \)
   - (if1) \( w_i := (w_{i+1} \land b) \lor \neg b \equiv w_{i+1} \lor \neg b \)
   - (if2) \( w_i := w_{i+1} \lor b \)
   - (wh1) \( w_i := w_{i+1} \lor \neg b \)
   - (wh2) \( w_i := w_{i+1} \lor b \)

4. Possible to show: \( s_i \models w_i \) for each \( i \in \{0, \ldots, k\} \)
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample \(\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle\) (according to Definition 16.1)
2. Construct strongest postconditions \(s_0, \ldots, s_k\) with \(s_0 \equiv \text{true}, s_k \equiv \text{false}\) (according to Lemma 16.2)
3. Analogously it is possible to construct weakest preconditions \(w_0, \ldots, w_k\) with \(w_0 \equiv \text{true}, w_k \equiv \text{false}\) starting from \(w_k\)
   i. \(w_k := \text{false}\)
   ii. for \(i = 0, \ldots, k - 1\): definition of \(w_i\) depending on \(w_{i+1}\) and on (axiom) transition rule applied in \(\langle c_i, \_ \rangle \Rightarrow \langle c_{i+1}, \_ \rangle\):
      - (skip) \(w_i := w_{i+1}\)
      - (asgn) \(w_i := w_{i+1}[x \mapsto a]\)
      - (if1) \(w_i := (w_{i+1} \land b) \lor \lnot b \equiv w_{i+1} \lor \lnot b\)
      - (if2) \(w_i := w_{i+1} \lor b\)
      - (wh1) \(w_i := w_{i+1} \lor \lnot b\)
      - (wh2) \(w_i := w_{i+1} \lor b\)
4. Possible to show: \(s_i \models w_i\) for each \(i \in \{0, \ldots, k\}\)
5. For each \(i \in \{0, \ldots, k\}\), choose Craig interpolant \(b_i\) of \(s_i\) and \(w_i\)
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample \( \langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle \) 
   (according to Definition 16.1)
2. Construct strongest postconditions \( s_0, \ldots, s_k \) with \( s_0 \equiv \text{true}, s_k \equiv \text{false} \) 
   (according to Lemma 16.2)
3. Analogously it is possible to construct weakest preconditions \( w_0, \ldots, w_k \) with \( w_0 \equiv \text{true}, w_k \equiv \text{false} \) starting from \( w_k \)
   i. \( w_k := \text{false} \)
   ii. for \( i = 0, \ldots, k-1 \): definition of \( w_i \) depending on \( w_{i+1} \) and on (axiom) transition rule applied in \( \langle c_i, . \rangle \Rightarrow \langle c_{i+1}, . \rangle \):
      ■ (skip) \( w_i := w_{i+1} \)
      ■ (asgn) \( w_i := w_{i+1}[x \mapsto a] \)
      ■ (if1) \( w_i := (w_{i+1} \land b) \lor \neg b \equiv w_{i+1} \lor \neg b \)
      ■ (if2) \( w_i := w_{i+1} \lor b \)
      ■ (wh1) \( w_i := w_{i+1} \lor \neg b \)
      ■ (wh2) \( w_i := w_{i+1} \lor b \)
4. Possible to show: \( s_i \models w_i \) for each \( i \in \{0, \ldots, k\} \)
5. For each \( i \in \{0, \ldots, k\} \), choose Craig interpolant \( b_i \) of \( s_i \) and \( w_i \)
6. Refine abstraction by atomic conjuncts occurring in \( b_1, \ldots, b_{k-1} \)
Craig Interpolation

Using Craig Interpolants I

1. Begin with spurious counterexample \( \langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \ldots \Rightarrow \langle c_k, Q_k \rangle \)
   (according to Definition 16.1)

2. Construct strongest postconditions \( s_0, \ldots, s_k \) with \( s_0 \equiv \text{true}, s_k \equiv \text{false} \)
   (according to Lemma 16.2)

3. Analogously it is possible to construct weakest preconditions \( w_0, \ldots, w_k \) with \( w_0 \equiv \text{true}, w_k \equiv \text{false} \) starting from \( w_k \)
   i. \( w_k := \text{false} \)
   ii. for \( i = 0, \ldots, k - 1 \): definition of \( w_i \) depending on \( w_{i+1} \) and on (axiom) transition rule applied in
      \( \langle c_i, \cdot \rangle \Rightarrow \langle c_{i+1}, \cdot \rangle \):
      - (skip) \( w_i := w_{i+1} \)
      - (asgn) \( w_i := w_{i+1}[x \mapsto a] \)
      - (if1) \( w_i := (w_{i+1} \land b) \lor \neg b \equiv w_{i+1} \lor \neg b \)
      - (if2) \( w_i := w_{i+1} \lor b \)
      - (wh1) \( w_i := w_{i+1} \lor \neg b \)
      - (wh2) \( w_i := w_{i+1} \lor b \)

4. Possible to show: \( s_i \models w_i \) for each \( i \in \{0, \ldots, k\} \)

5. For each \( i \in \{0, \ldots, k\} \), choose Craig interpolant \( b_i \) of \( s_i \) and \( w_i \)

6. Refine abstraction by atomic conjuncts occurring in \( b_1, \ldots, b_{k-1} \)

Remark: Craig interpolants always exist for first-order formulae (but are not necessarily unique)
Example 17.3 (cf. Example 16.3)

Let \( c_0 := [x := z]^0;[z := z + 1]^1;[y := z]^2; 
\text{if } [x = y]^3 \text{ then } [\text{skip}]^4 \text{ else } [\text{skip}]^5 \text{ end} \)
Craig Interpolation

Using Craig Interpolants II

Example 17.3 (cf. Example 16.3)

Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2$;
if $[x = y]^3$ then $[\text{skip}]^4$ else $[\text{skip}]^5$ end

1. Spurious counterexample:

$$\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$$
Example 17.3 (cf. Example 16.3)

Let $c_0 := [x := z]^0;[z := z + 1]^1;[y := z]^2$; if $[x = y]^3$ then $[\text{skip}]^4$ else $[\text{skip}]^5$ end

1. Spurious counterexample:
   \[
   \langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle
   \]

2. Strongest postconditions (cf. Example 16.3):
   \[
   s_0 = \text{true}
   \]
   \[
   s_1 = (x = z)
   \]
   \[
   s_2 = (x + 1 = z)
   \]
   \[
   s_3 = (x + 1 = z \land y = z)
   \]
   \[
   s_4 = \text{false}
   \]
Craig Interpolation

Using Craig Interpolants II

Example 17.3 (cf. Example 16.3)

Let $c_0 := [x := z]^0;[z := z + 1]^1;[y := z]^2;\quad$ if $[x = y]^3$ then $[\text{skip}]^4$ else $[\text{skip}]^5$ end

1. Spurious counterexample:
   \[
   \langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle
   \]

2. Strongest postconditions (cf. Example 16.3): $s_0 = \text{true}$
   
   $s_1 = (x = z)$
   
   $s_2 = (x + 1 = z)$
   
   $s_3 = (x + 1 = z \land y = z)$
   
   $s_4 = \text{false}$

3. Weakest preconditions $w_i$: on the board

4. Craig interpolants $b_i$: on the board
Outline of Lecture 17

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice
CEGAR in Practice

SLAM Tool

- was: Software, Languages, Analysis, and Modeling
  “SLAM originally was an acronym but we found it too cumbersome to explain. We now prefer to think of ‘slamming’ the bugs in a program.”
- First implementation of CEGAR for C programs
- Checks behavioural requirements of software interfaces
  – e.g., “a thread may not acquire a lock it has already acquired, or release a lock it does not hold”
- Supports recursive procedures, pointers, and memory allocation
- Sub-tools:
  – C2bp: C program × Predicates → Boolean program (Boolean variables = predicates)
  – BEPOP: symbolic (BDD-based) model checker for (recursive) Boolean programs
  – newton: abstraction refinement
- Developed into commercial product (Static Driver Verifier – SDV; part of Windows Driver Foundation development kit)
CEGAR in Practice

CPAchecker Tool

- CPA: “Configurable Program Analysis”
- Java re-implementation of Berkeley Lazy Abstraction Software Verification Tool (BLAST)
- Software model checker for C programs
- Uses CEGAR with Craig interpolation and lazy abstraction
  - abstraction is constructed on-the-fly
  - model locally refined on demand
  - enables use of different predicates at different program points
  $\Rightarrow$ “abstract reachability tree”
- Successfully applied to C programs with $> 130,000$ LOC
- WWW: http://cpachecker.sosy-lab.org/
CEGAR in Practice

Practical Experiences


- Predicate abstraction & CEGAR suitable for checking control-flow-related safety properties
  - predicates good for representation of control flow
  - safety (“Nothing bad is going to happen.”) goes well with over-approximation
  - liveness (“Eventually something good will happen.”) requires under-approximation

- Does not work well with complex heap-based data structures or arrays
  (⇒ Pointer/Shape Analysis)

- (Real) counterexamples often more useful than correctness proof

- Abstraction refinement cycle may not terminate

- Main application field: safety properties of device drivers and systems code up to 50 kLOCs