



# Static Program Analysis

Lecture 16: Abstract Interpretation VII (Limits & Improvements of CEGAR)

Summer Semester 2018

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ss-18/spa/>

---

## Schedule of Forthcoming Lectures

June 25/26: Interprocedural dataflow analysis

July 02/03: Pointer and shape analysis

July 09(/10): Wrap-Up

# Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

---

## Outline of Lecture 16

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

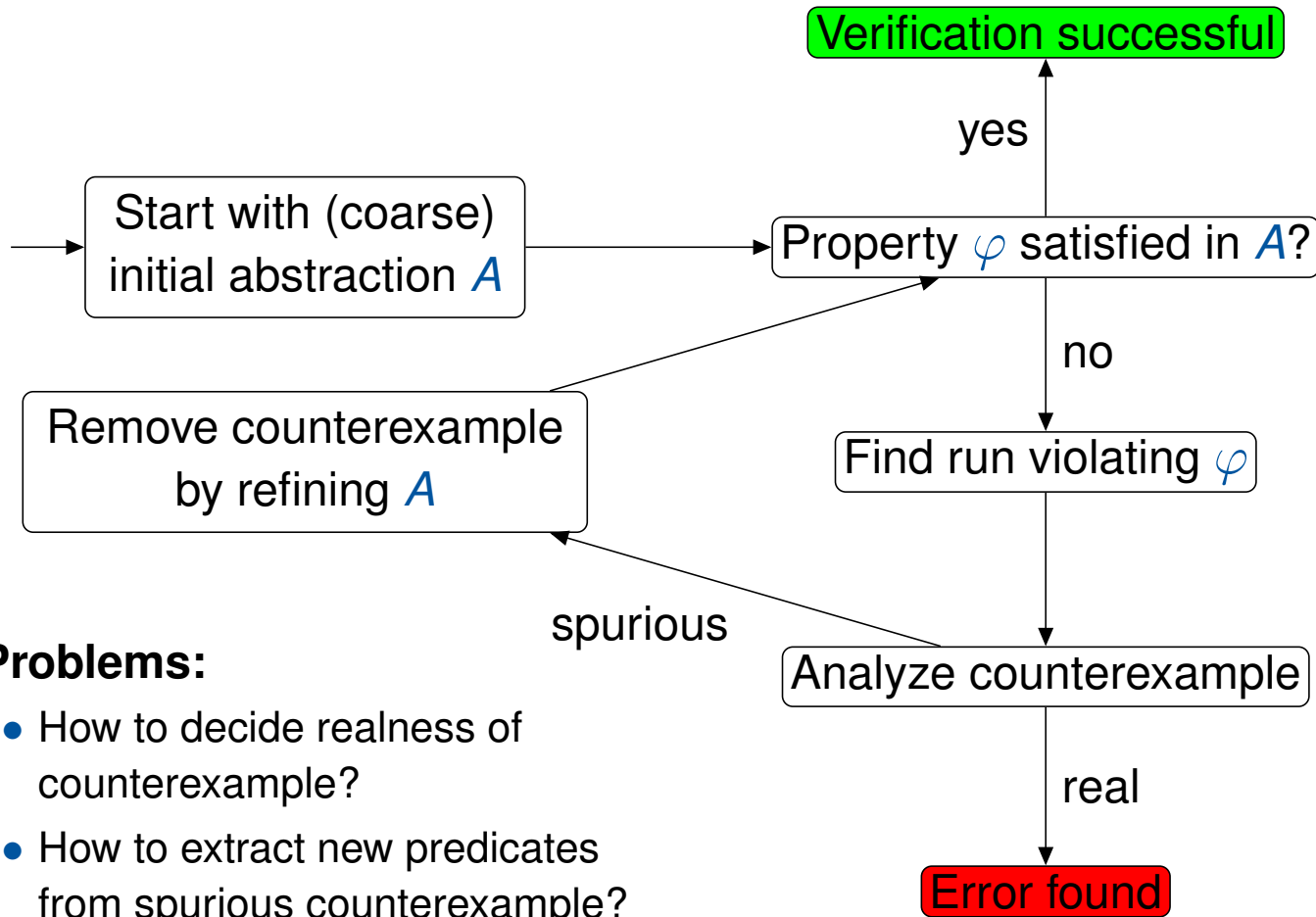
Where CEGAR Fails

Craig Interpolation

CEGAR in Practice

# Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

## Reminder: CEGAR



## Problems:

- How to decide realness of counterexample?
- How to extract new predicates from spurious counterexample?

# 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
- ⇒ All representable as (non-)reachability of “bad locations”
- ⇒ Counterexample = path to bad locations

## Definition (Counterexample)

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

$$\langle c_0, Q_0 \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$$

where

- $c_0, \dots, c_k \in \text{Cmd}$  (or  $c_k = \downarrow$ )
- $Q_0, \dots, Q_k \in \text{Abs}(P)$  with  $Q_0 \equiv \text{true}$  and  $Q_k \not\equiv \text{false}$
- It is called **real** if there exist concrete states  $\sigma_0, \dots, \sigma_k \in \Sigma$  such that
$$\forall i \in \{1, \dots, k\} : \sigma_i \models Q_i \quad \text{and} \quad \langle c_{i-1}, \sigma_{i-1} \rangle \rightarrow \langle c_i, \sigma_i \rangle$$
- Otherwise it is called **spurious**.

# Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

## Elimination of Spurious Counterexamples

### Lemma

If  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$  is a spurious counterexample, there exist Boolean expressions  $b_0, \dots, b_k$  with  $b_0 \equiv \text{true}$ ,  $b_k \equiv \text{false}$ , and

$$\forall i \in \{1, \dots, k\}, \sigma, \sigma' \in \Sigma : \sigma \models b_{i-1} \wedge \langle c_{i-1}, \sigma \rangle \rightarrow \langle c_i, \sigma' \rangle \implies \sigma' \models b_i$$

### Proof (idea).

Inductive definition of  $b_i$  as **strongest postconditions**:

1.  $b_0 := \text{true}$
2. for  $i = 1, \dots, k$ : definition of  $b_i$  depending on  $b_{i-1}$  and on (axiom) transition rule applied in  $\langle c_{i-1}, \cdot \rangle \Rightarrow \langle c_i, \cdot \rangle$ :

Atomic commands:

- (skip)  $b_i := b_{i-1}$
- (asgn)  $b_i := \exists x'. (b_{i-1} [x \mapsto x'] \wedge x = a[x \mapsto x'])$   
(for “ $x := a$ ”;  $x'$  = previous value of  $x$ )

Control structures (with guard  $b$ ):

- (if1)  $b_i := b_{i-1} \wedge b$
- (if2)  $b_i := b_{i-1} \wedge \neg b$
- (wh1)  $b_i := b_{i-1} \wedge b$
- (wh2)  $b_i := b_{i-1} \wedge \neg b$

(yields  $b_k \equiv \text{false}$ ; by induction on  $k$ )

□

# Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

## Abstraction Refinement

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

## Lemma

*After refinement, the spurious counterexample*

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

*with  $Q_k \neq \text{false}$  does not exist any more.*

## Proof.

omitted □

# Where CEGAR Fails

---

## Outline of Lecture 16

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice



## Where CEGAR Fails

---

### Where CEGAR Fails

#### Example 16.1

- $c := [x := a]^0;$   
   $[y := b]^1;$   
  while  $[\neg(x = 0)]^2$  do  
     $[x := x - 1]^3;$   
     $[y := y - 1]^4$   
  end;  
  if  $[a = b \wedge \neg(y = 0)]^5$  then  
     $[\text{skip}]^6$   
  else  
     $[\text{skip}]^7$   
  end
- **Interesting property:** label 6 unreachable

## Where CEGAR Fails

---

### Where CEGAR Fails

#### Example 16.1

- $c := [x := a]^0;$   
   $[y := b]^1;$   
  while  $[\neg(x = 0)]^2$  do  
     $[x := x - 1]^3;$   
     $[y := y - 1]^4$   
  end;  
  if  $[a = b \wedge \neg(y = 0)]^5$  then  
     $[\text{skip}]^6$   
  else  
     $[\text{skip}]^7$   
  end
- **Interesting property:** label 6 unreachable

- **Initial abstraction:**  $P = \emptyset$   
  ( $\implies \text{Abs}(P) = \{\text{true}, \text{false}\}$ )
- **Abstraction refinement:** on the board

## Where CEGAR Fails

### Where CEGAR Fails

#### Example 16.1

- $c := [x := a]^0;$   
   $[y := b]^1;$   
  while  $[\neg(x = 0)]^2$  do  
     $[x := x - 1]^3;$   
     $[y := y - 1]^4$   
  end;  
  if  $[a = b \wedge \neg(y = 0)]^5$  then  
     $[\text{skip}]^6$   
  else  
     $[\text{skip}]^7$   
  end
- **Interesting property:** label 6 unreachable

- **Initial abstraction:**  $P = \emptyset$   
  ( $\implies \text{Abs}(P) = \{\text{true}, \text{false}\}$ )
- **Abstraction refinement:** on the board
- **Observation:** iteration yields predicates of the form

$$x = a - k \quad \text{and} \quad y = b - k$$

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 16

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

CEGAR in Practice

# Craig Interpolation

---

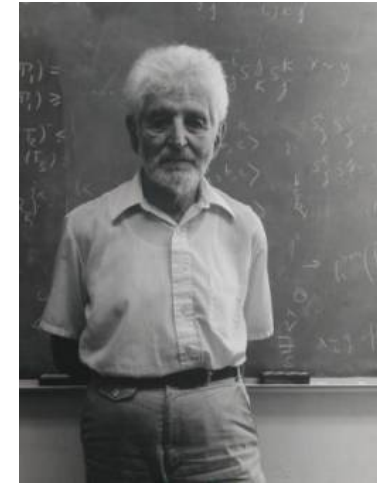
## Craig Interpolation

- **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

## 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 16.2 (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  $Var_{b_3} \subseteq Var_{b_1} \cap Var_{b_2}$ .

# Craig Interpolation

---

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)



## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)
3. Analogously it is possible to construct **weakest preconditions**  $w_0, \dots, w_k$  with  $w_0 \equiv \text{true}$ ,  $w_k \equiv \text{false}$  starting from  $w_k$ 
  - i.  $w_k := \text{false}$
  - ii. for  $i = k - 1, \dots, 0$ : 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)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$
    - (if2)  $w_i := w_{i+1} \vee b$
    - (wh1)  $w_i := w_{i+1} \vee \neg b$
    - (wh2)  $w_i := w_{i+1} \vee b$

# Craig Interpolation

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)
3. Analogously it is possible to construct **weakest preconditions**  $w_0, \dots, w_k$  with  $w_0 \equiv \text{true}$ ,  $w_k \equiv \text{false}$  starting from  $w_k$ 
  - i.  $w_k := \text{false}$
  - ii. for  $i = k - 1, \dots, 0$ : 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)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$
    - (if2)  $w_i := w_{i+1} \vee b$
    - (wh1)  $w_i := w_{i+1} \vee \neg b$
    - (wh2)  $w_i := w_{i+1} \vee b$
4. Possible to show:  $s_i \models w_i$  for each  $i \in \{0, \dots, k\}$

# Craig Interpolation

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)
3. Analogously it is possible to construct **weakest preconditions**  $w_0, \dots, w_k$  with  $w_0 \equiv \text{true}$ ,  $w_k \equiv \text{false}$  starting from  $w_k$ 
  - i.  $w_k := \text{false}$
  - ii. for  $i = k - 1, \dots, 0$ : 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)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$
    - (if2)  $w_i := w_{i+1} \vee b$
    - (wh1)  $w_i := w_{i+1} \vee \neg b$
    - (wh2)  $w_i := w_{i+1} \vee b$
4. Possible to show:  $s_i \models w_i$  for each  $i \in \{0, \dots, k\}$
5. For each  $i \in \{0, \dots, k\}$ , choose **Craig interpolant**  $b_i$  of  $s_i$  and  $w_i$

# Craig Interpolation

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)
3. Analogously it is possible to construct **weakest preconditions**  $w_0, \dots, w_k$  with  $w_0 \equiv \text{true}$ ,  $w_k \equiv \text{false}$  starting from  $w_k$ 
  - i.  $w_k := \text{false}$
  - ii. for  $i = k - 1, \dots, 0$ : 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)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$
    - (if2)  $w_i := w_{i+1} \vee b$
    - (wh1)  $w_i := w_{i+1} \vee \neg b$
    - (wh2)  $w_i := w_{i+1} \vee b$
4. Possible to show:  $s_i \models w_i$  for each  $i \in \{0, \dots, k\}$
5. For each  $i \in \{0, \dots, k\}$ , choose **Craig interpolant**  $b_i$  of  $s_i$  and  $w_i$
6. **Refine abstraction** by atomic conjuncts occurring in  $b_1, \dots, b_{k-1}$

# Craig Interpolation

---

## Using Craig Interpolants I

1. Begin with **spurious counterexample**  $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$   
(according to Definition 15.1)
2. Construct **strongest postconditions**  $s_0, \dots, s_k$  with  $s_0 \equiv \text{true}$ ,  $s_k \equiv \text{false}$   
(according to Lemma 15.2)
3. Analogously it is possible to construct **weakest preconditions**  $w_0, \dots, w_k$  with  $w_0 \equiv \text{true}$ ,  $w_k \equiv \text{false}$  starting from  $w_k$ 
  - i.  $w_k := \text{false}$
  - ii. for  $i = k - 1, \dots, 0$ : 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)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$
    - (if2)  $w_i := w_{i+1} \vee b$
    - (wh1)  $w_i := w_{i+1} \vee \neg b$
    - (wh2)  $w_i := w_{i+1} \vee b$
4. Possible to show:  $s_i \models w_i$  for each  $i \in \{0, \dots, k\}$
5. For each  $i \in \{0, \dots, k\}$ , choose **Craig interpolant**  $b_i$  of  $s_i$  and  $w_i$
6. **Refine abstraction** by atomic conjuncts occurring in  $b_1, \dots, b_{k-1}$

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

---

# Craig Interpolation

---

## Using Craig Interpolants II

### Example 16.3 (cf. Example 15.3)

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

# Craig Interpolation

---

## Using Craig Interpolants II

### Example 16.3 (cf. Example 15.3)

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

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

## Using Craig Interpolants II

### Example 16.3 (cf. Example 15.3)

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 15.3):  $s_0 = \text{true}$

$$s_1 = (x = z)$$

$$s_2 = (z = x + 1)$$

$$s_3 = (z = x + 1 \wedge y = z)$$

$$s_4 = \text{false}$$



# Craig Interpolation

## Using Craig Interpolants II

### Example 16.3 (cf. Example 15.3)

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 15.3):  $s_0 = \text{true}$

$$s_1 = (x = z)$$

$$s_2 = (z = x + 1)$$

$$s_3 = (z = x + 1 \wedge y = z)$$

$$s_4 = \text{false}$$

3. **Weakest preconditions**  $w_i$ : on the board

– (skip)  $w_i := w_{i+1}$

– (asgn)  $w_i := w_{i+1}[x \mapsto a]$

– (if1)  $w_i := (w_{i+1} \wedge b) \vee \neg b \equiv w_{i+1} \vee \neg b$

– (if2)  $w_i := w_{i+1} \vee b$

– (wh1)  $w_i := w_{i+1} \vee \neg b$

– (wh2)  $w_i := w_{i+1} \vee b$

4. **Craig interpolants**  $b_i$  with  $s_i \models b_i$ ,  $b_i \models w_i$ ,  $\text{Var}_{b_i} \subseteq \text{Var}_{s_i} \cap \text{Var}_{w_i}$ : on the board

# CEGAR in Practice

---

## Outline of Lecture 16

Recap: Counterexample-Guided Abstraction Refinement (CEGAR)

Where CEGAR Fails

Craig Interpolation

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.”*  
    *(T. Ball, B. Cook, V. Levin, S.K. Rajamani: SLAM and Static Driver Verifier: Technology Transfer of Formal Methods inside Microsoft, IFM 2004)*
- 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  $\times$  Predicates  $\rightarrow$  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)
  - T. Ball, V. Levin, S.K. Rajamani: *A Decade of Software Model Checking with SLAM*. Comm. ACM 54(7), 2011, 68–76
- WWW: <http://research.microsoft.com/en-us/projects/slam/>

## 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
- ⇒ “abstract reachability tree”
- Successfully applied to C programs with  $> 130,000$  LOC
  - D. Beyer, M.E. Keremoglu: *CPAchecker: A Tool for Configurable Software Verification*. Proc. CAV, 2011, 184–190
- WWW: <http://cpachecker.sosy-lab.org/>

## Practical Experiences

(cf. V. D'Silva, D. Kroening, G. Weissenbacher: *A Survey of Automated Techniques for Formal Software Verification*, IEEE Trans. on CAD of Integrated Circuits and Systems 27(7), 2008, 1165–1178)

- 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**  
( $\Rightarrow$  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