Static Program Analysis

Lecture 16: Abstract Interpretation VI (Counterexample-Guided Abstraction Refinement)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

http://moves.rwth-aachen.de/teaching/ws-1415/spa/

Winter Semester 2014/15

Oral Exam in Static Program Analysis

- Options:
  - Thu 12 March
  - Tue 24 March
  - Thu 26 March
  - Wed 08 April
- Registration via https://terminplaner2.dfn.de/foodle/Exam-Static-Program-Analysis-54991 (accessible through http://moves.rwth-aachen.de/teaching/ws-1415/spa/)

Outline

1. Recap: Predicate Abstraction
2. Additional Remarks
3. Counterexample-Guided Abstraction Refinement

Predicate Abstraction I

Definition (Predicate abstraction)

Let Var be a set of variables.

- A predicate is a Boolean expression $p \in BExp$ over Var.
- A state $\sigma \in \Sigma$ satisfies $p \in BExp$ ($\sigma \models p$) if $val_\sigma(p) = \text{true}$.
- p implies q ($p \models q$) if $\sigma \models q$ whenever $\sigma \models p$ (or: p is stronger than q, q is weaker than p).
- p and q are equivalent ($p \equiv q$) if $p \models q$ and $q \models p$.
- Let $P = \{p_1, \ldots, p_n\} \subseteq BExp$ be a finite set of predicates, and let $\neg P := \{\neg p_1, \ldots, \neg p_n\}$. An element of $P \cup \neg P$ is called a literal. The predicate abstraction lattice is defined by:

$$Abs(p_1, \ldots, p_n) := \left(\left\{\bigwedge Q \mid Q \subseteq P \cup \neg P\right\}, \models\right).$$

Abbreviations: $\text{true} := \bigwedge \emptyset, \text{false} := \bigwedge \{p_i, \neg p_i, \ldots\}$

Predicate Abstraction II

Lemma

$Abs(p_1, \ldots, p_n)$ is a complete lattice with

- $\bot = \text{false}$, $\top = \text{true}$
- $Q_1 \sqcap Q_2 = Q_1 \wedge Q_2$
- $Q_1 \sqcup Q_2 = \overline{Q_1 \vee Q_2}$ *where* $\overline{b} := \bigwedge\{q \in P \cup \neg P \mid b \models q\}$
  (i.e., strongest formula in $Abs(p_1, \ldots, p_n)$ that is implied by $Q_1 \vee Q_2$)

Example

Let $P := \{p_1, p_2, p_3\}$.

1. For $Q_1 := p_1 \wedge \neg p_2$ and $Q_2 := \neg p_2 \wedge p_3$, we obtain
$$Q_1 \sqcap Q_2 = Q_1 \wedge Q_2 \equiv p_1 \wedge \neg p_2 \wedge p_3$$
$$Q_1 \sqcup Q_2 = \overline{Q_1 \vee Q_2} \equiv \overline{\neg p_2 \wedge (p_1 \vee p_3)} \equiv \neg p_2$$
2. For $Q_1 := p_1 \wedge p_2$ and $Q_2 := p_1 \wedge \neg p_2$, we obtain
$$Q_1 \sqcap Q_2 = Q_1 \wedge Q_2 \equiv \text{false}$$
$$Q_1 \sqcup Q_2 = \overline{Q_1 \vee Q_2} \equiv \overline{p_1 \wedge (p_2 \vee \neg p_2)} \equiv p_1$$

Predicate Abstraction III

Definition (Galois connection for predicate abstraction)

The Galois connection for predicate abstraction is determined by

$$\alpha : 2^\Sigma \to Abs(p_1, \ldots, p_n) \quad \text{and} \quad \gamma : Abs(p_1, \ldots, p_n) \to 2^\Sigma$$

with

$$\alpha(S) := \bigsqcup \{Q_\sigma \mid \sigma \in S\} \quad \text{and} \quad \gamma(Q) := \{\sigma \in \Sigma \mid \sigma \models Q\}$$

where $Q_\sigma := \bigwedge(\{p_i \mid 1 \leq i \leq n, \sigma \models p_i\} \cup \{\neg p_i \mid 1 \leq i \leq n, \sigma \not\models p_i\})$.

Example

- Let $Var := \{\mathrm{x}, \mathrm{y}\}$
- Let $P := \{p_1, p_2, p_3\}$ where $p_1 := (\mathrm{x<=y})$, $p_2 := (\mathrm{x=y})$, $p_3 := (\mathrm{x>y})$
- If $S = \{\sigma_1, \sigma_2\} \subseteq \Sigma$ with $\sigma_1 = [\mathrm{x} \mapsto 1, \mathrm{y} \mapsto 2]$, $\sigma_2 = [\mathrm{x} \mapsto 2, \mathrm{y} \mapsto 2]$, then
$$\begin{aligned} \alpha(S) &= Q_{\sigma_1} \sqcup Q_{\sigma_2} \\ &= \underline{(p_1 \wedge \neg p_2 \wedge \neg p_3) \sqcup (p_1 \wedge p_2 \wedge \neg p_3)} \\ &= (p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge \neg p_3) \\ &\equiv p_1 \wedge \neg p_3 \end{aligned}$$
- If $Q = p_1 \wedge \neg p_2 \in Abs(p_1, \ldots, p_n)$, then $\gamma(Q) = \{\sigma \in \Sigma \mid \sigma(\mathrm{x}) < \sigma(\mathrm{y})\}$

Definition (Execution relation for predicate abstraction)

If $c \in Cmd$ and $Q \in Abs(p_1, \ldots, p_n)$, then $\langle c, Q\rangle$ is called an abstract configuration. The execution relation for predicate abstraction is defined by the following rules:

$$\text{(skip)}\ \frac{}{\langle \texttt{skip}, Q\rangle \Rightarrow \langle \downarrow, Q\rangle} \qquad \text{(asgn)}\ \frac{}{\langle x \texttt{ := } a, Q\rangle \Rightarrow \langle \downarrow, \bigsqcup\{Q_{\sigma[x\mapsto val_\sigma(a)]} \mid \sigma \models Q\}\rangle}$$

$$\text{(seq1)}\ \frac{\langle c_1, Q\rangle \Rightarrow \langle c_1', Q'\rangle \quad c_1' \neq \downarrow}{\langle c_1 ; c_2, Q\rangle \Rightarrow \langle c_1' ; c_2, Q'\rangle} \qquad \text{(seq2)}\ \frac{\langle c_1, Q\rangle \Rightarrow \langle \downarrow, Q'\rangle}{\langle c_1 ; c_2, Q\rangle \Rightarrow \langle c_2, Q'\rangle}$$

$$\text{(if1)}\ \frac{}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, Q\rangle \Rightarrow \langle c_1, \overline{Q \wedge b}\rangle}$$

$$\text{(if2)}\ \frac{}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, Q\rangle \Rightarrow \langle c_2, \overline{Q \wedge \neg b}\rangle}$$

$$\text{(wh1)}\ \frac{}{\langle \texttt{while } b \texttt{ do } c, Q\rangle \Rightarrow \langle c ; \texttt{while } b \texttt{ do } c, \overline{Q \wedge b}\rangle}$$

$$\text{(wh2)}\ \frac{}{\langle \texttt{while } b \texttt{ do } c, Q\rangle \Rightarrow \langle \downarrow, \overline{Q \wedge \neg b}\rangle}$$

Outline

1. Recap: Predicate Abstraction
2. Additional Remarks
3. Counterexample-Guided Abstraction Refinement

Additional Remarks

In Rules (if1, (if2), (wh1), (wh2), the fact that $b = p_i$ for some $i \in \{1, \ldots, n\}$ implies $Q \wedge [\neg]b \in Abs(p_1, \ldots, p_n)$, but not $\overline{Q \wedge [\neg]b} = Q \wedge [\neg]b$

Example 16.1 (cf. Example 15.7)

- $p_1 :=$ (x > y), $p_2 :=$ (x >= y)
- $Q :=$ true, $b := p_1$

$\Rightarrow \overline{Q \wedge b} = p_1 \wedge p_2 \neq Q \wedge b = p_1$

For similar reasons, generally $Q_1 \sqcup Q_2 \; (= \overline{Q_1 \vee Q_2}) \neq Q_1 \cap Q_2$

Example 16.2

- $p_1 :=$ (x > y), $p_2 :=$ (x >= y), $p_3 :=$ (x = y)
- $Q_1 := p_1 \wedge p_2 \wedge \neg p_3$ ($\equiv$ x > y), $Q_2 := p_3$ ($\equiv$ x = y)

$\Rightarrow Q_1 \sqcup Q_2 = \overline{Q_1 \vee Q_2} = p_2 \neq Q_1 \cap Q_2 = \text{true}$

Computation of Postconditions

Problem: $\overline{b} = \bigwedge\{q \in P \cup \neg P \mid b \models q\}$ (i.e., the strongest formula in $Abs(p_1, \ldots, p_n)$ that is implied by b) is generally not computable (due to undecidability of implication in certain logics)

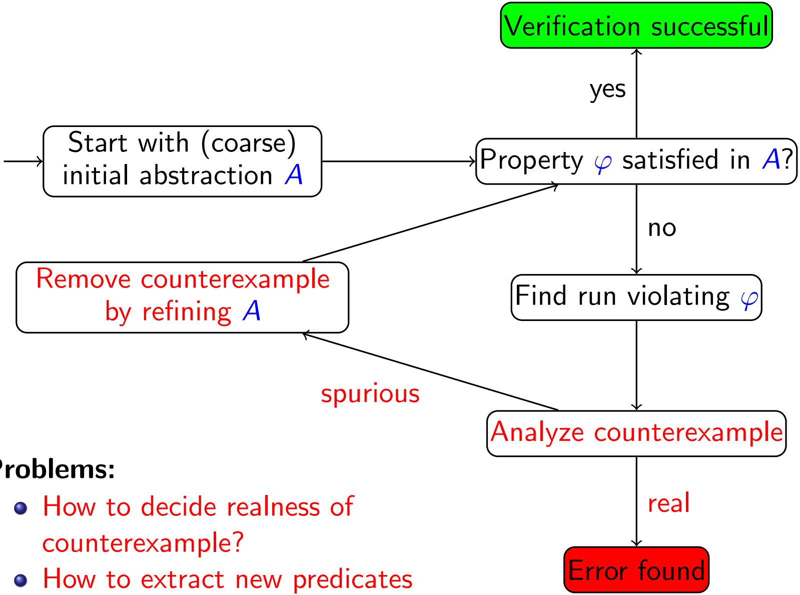
Solutions:

- Over-approximation: fall back to non-strongest postconditions
  - in practice, (automatic) theorem proving
  - for every $i \in \{1, \ldots, n\}$, try to prove $b \models p_i$ and $b \models \neg p_i$
  - approximate $\overline{b}$ by conjunction of all provable literals
- Restriction of programs:
  - $\models$ decidable for certain logics
  - example: Presburger arithmetic (first-order theory of $\mathbb{N}$ with $+$)
  - thus $\overline{b}$ computable for WHILE programs without multiplication
- Restriction to finite domains:
  - for example, binary numbers of fixed size
  - thus everything (domain, Galois connection, ...) exactly computable
  - problem: exponential blowup $\Longrightarrow$ solution: Binary Decision Diagrams

Outline

1. Recap: Predicate Abstraction
2. Additional Remarks
3. Counterexample-Guided Abstraction Refinement

Reminder: CEGAR



Problems:

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

Counterexamples

Typical 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

Definition 16.3 (Counterexample)

- A counterexample is a sequence of abstract transitions of the form

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

  where
  - $k \geq 1$
  - $c_0, \ldots, c_k \in Cmd$ (or $c_k = \downarrow$)
  - $Q_1, \ldots, Q_k \in Abs(p_1, \ldots, p_n)$ with $Q_k \not\equiv \mathsf{false}$
- It is called real if there exist concrete states $\sigma_0, \ldots, \sigma_k \in \Sigma$ such that

$$\forall i \in \{1, \ldots, k\} : \sigma_i \models Q_i \text{ and } \langle c_{i-1}, \sigma_{i-1}\rangle \rightarrow \langle c_i, \sigma_i\rangle$$

- Otherwise it is called spurious.

Elimination of Spurious Counterexamples I

Lemma 16.4

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

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

Proof (idea).

Inductive definition of b_i as strongest postconditions:

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

- (skip) $b_i := b_{i-1}$
- (asgn) $b_i := \exists x'.(b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
  (x' = previous value of x)
- (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 $p_k \equiv \text{false}$; by induction on k) □

Elimination of Spurious Counterexamples II

Example 16.5

- Let $c_0 := [\mathtt{x\ :=\ z}]^0;[\mathtt{z\ :=\ z\ +\ 1}]^1;[\mathtt{y\ :=\ z}]^2;$
  $\mathtt{if}\ [\mathtt{x\ =\ y}]^3\ \mathtt{then}\ [\mathtt{skip}]^4\ \mathtt{else}\ [\mathtt{skip}]^5$
- Interesting property: after termination, $\mathtt{x} \neq \mathtt{y}$, i.e., label 4 unreachable
- Initial abstraction: $P = \emptyset$ ($\implies$ $Abs(P) = \{\mathsf{true}, \mathsf{false}\}$)
- (Spurious) counterexample:

$$\langle 0, \mathsf{true}\rangle \Rightarrow \langle 1, \mathsf{true}\rangle \Rightarrow \langle 2, \mathsf{true}\rangle \Rightarrow \langle 3, \mathsf{true}\rangle \Rightarrow \langle 4, \mathsf{true}\rangle$$

- Forward construction of Boolean expressions:
  - $b_0 := \mathsf{true}$
  - (asgn) $b_i := \exists x'.(b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies b_1 := \exists x'.(b_0[\mathtt{x} \mapsto x'] \wedge \mathtt{x} = \mathtt{z}[\mathtt{x} \mapsto x']) \equiv (\mathtt{x} = \mathtt{z})$
  - (asgn) $b_i := \exists x'.(b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies b_2 := \exists z'.(b_1[\mathtt{z} \mapsto z'] \wedge \mathtt{z} = \mathtt{z} + 1[\mathtt{z} \mapsto z'])$
    $= \exists z'.(\mathtt{x} = z' \wedge \mathtt{z} = z' + 1) \equiv (\mathtt{x} + 1 = \mathtt{z})$
  - (asgn) $b_i := \exists x'.(b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies b_3 := \exists y'.(b_2[\mathtt{y} \mapsto y'] \wedge \mathtt{y} = \mathtt{z}[\mathtt{y} \mapsto y']) \equiv (\mathtt{x} + 1 = \mathtt{z} \wedge \mathtt{y} = \mathtt{z})$
  - (if1) $b_i := b_{i-1} \wedge b$
    $\implies b_4 := b_3 \wedge \mathtt{x} = \mathtt{y} \equiv (\mathtt{x} + 1 = \mathtt{z} \wedge \mathtt{y} = \mathtt{z} \wedge \mathtt{x} = \mathtt{y}) \equiv \mathsf{false}$

Abstraction Refinement

Abstraction refinement step:

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

Lemma 16.6

After refinement, the spurious counterexample

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

with $Q_k \neq \mathsf{false}$ does not exist anymore.

Proof.

omitted □

A Simple Example

Example 16.7 (cf. Example 16.5)

- Let $c_0 := [\texttt{x := z}]^0; [\texttt{z := z + 1}]^1; [\texttt{y := z}]^2;$ $\texttt{if } [\texttt{x = y}]^3 \texttt{ then } [\texttt{skip}]^4 \texttt{ else } [\texttt{skip}]^5$
- $P = \emptyset$, $P' = \{\underbrace{\mathtt{x} = \mathtt{z}}_{p_1}, \underbrace{\mathtt{x} + 1 = \mathtt{z}}_{p_2}, \underbrace{\mathtt{y} = \mathtt{z}}_{p_3}\}$
- Refined abstract transitions:

$$\begin{aligned} \langle 0, \text{true}\rangle &\Rightarrow \langle 1, p_1 \wedge \neg p_2\rangle \\ &\Rightarrow \langle 2, \neg p_1 \wedge p_2\rangle \\ &\Rightarrow \langle 3, \neg p_1 \wedge p_2 \wedge p_3\rangle \\ &\Rightarrow \langle 4, \underbrace{\overline{\neg p_1 \wedge p_2 \wedge p_3 \wedge \mathtt{x{=}y}}}_{\equiv \text{false}}\rangle \end{aligned}$$

Another Example: Multiplication

Example 16.8

- Let $c_0 :=$

```
[z := 0]^0;
while [x > 0]^1 do
  [z := z + y]^2;
  [x := x - 1]^3;
if [z mod y = 0]^4 then
  [skip]^5;
else
  [skip]^6;
```

- Initial assumption: $y > 0$
- Interesting property: label 6 unreachable
- Initial abstraction: $P = \emptyset$ ($\implies$ $Abs(P) = \{true, false\}$)
- Abstraction refinement: on the board