Static Program Analysis

Lecture 18: Abstract Interpretation VII (Counterexample-Guided Abstraction Refinement)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/spa11/

Summer Semester 2011

Outline

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: true $:= \bigwedge \emptyset$, false $:= \ldots \wedge p_i \wedge \ldots \wedge \neg p_i \wedge \ldots$

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

Abstract Semantics for Predicate Abstraction

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:

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

$$\text{(seq1)}\ \frac{\langle c_1, q \rangle \Rightarrow \langle c_1', q' \rangle}{\langle c_1 \texttt{;} c_2, q \rangle \Rightarrow \langle c_1' \texttt{;} c_2, q' \rangle} \qquad \text{(seq2)}\ \frac{\langle c_1, q \rangle \Rightarrow q'}{\langle c_1 \texttt{;} c_2, q \rangle \Rightarrow \langle c_2, q' \rangle}$$

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

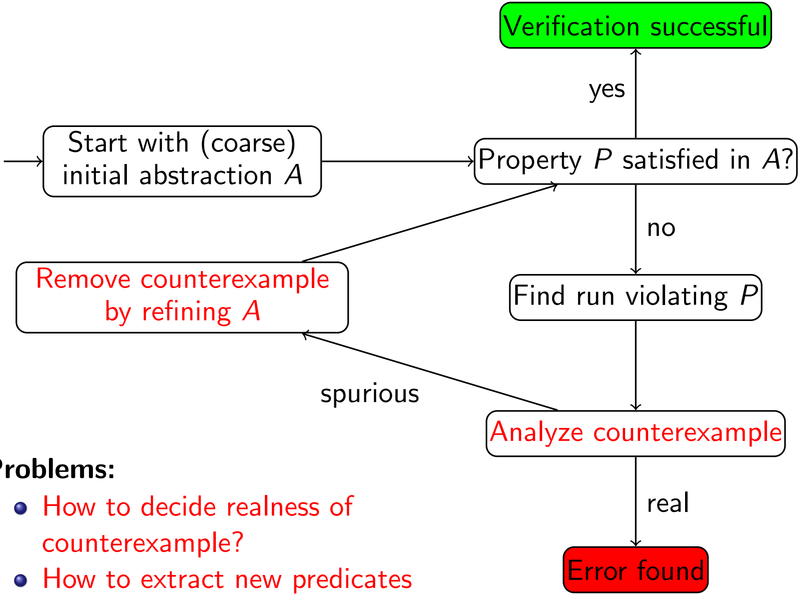
$$\text{(if2)}\ \frac{}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, q \rangle \Rightarrow \langle c_2, \overline{q \wedge \neg b} \rangle}$$

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

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

Outline

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 18.1 (Counterexample)

- A counterexample is a sequence of 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
  - $k \geq 1$
  - $c_0, \ldots, c_k \in Cmd$ [or $c_k = \downarrow$]
  - $q_1, \ldots, q_k \in Abs(p_1, \ldots, p_n)$ with $q_k \not\equiv \text{false}$
- It is called real if there exist concrete states $\sigma_0, \ldots, \sigma_k \in \Sigma$ such that
  $$\langle c_0, \sigma_0\rangle \rightarrow \langle c_1, \sigma_1\rangle \rightarrow \ldots \rightarrow \langle c_k, \sigma_k\rangle$$
- Otherwise it is called spurious.

Elimination of Spurious Counterexamples I

Lemma 18.2

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 predicates* $p_0, \ldots, p_k$ *such that* $p_0 \equiv \text{true}$, $p_k \equiv \text{false}$, *and*

$$\forall i \in \{1, \ldots, k\}, \sigma, \sigma' \in \Sigma : \sigma \models p_{i-1}, \langle c_{i-1}, \sigma\rangle \rightarrow \langle c_i, \sigma'\rangle \implies \sigma' \models p_i$$

Proof (idea).

Inductive definition of p_i as strongest postconditions:

1. $p_0 := \text{true}$
2. for $i = 1, \ldots, k$: definition of p_i depending on p_{i-1} and on (axiom) transition rule applied in $\langle c_{i-1}, .\rangle \Rightarrow \langle c_i, .\rangle$:

- (skip) $p_i := p_{i-1}$
- (asgn) $p_i := (p_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$ (x' = previous value of x; existentially quantified)
- (if1) $p_i := p_{i-1} \wedge b$
- (if2) $p_i := p_{i-1} \wedge \neg b$
- (wh1) $p_i := p_{i-1} \wedge b$
- (wh2) $p_i := p_{i-1} \wedge \neg b$

Elimination of Spurious Counterexamples II

Example 18.3

- Let $c_0 :=$ `[x := z]`^{0}`;[z := z + 1]`^{1}`;[y := z]`^{2}`;`
  `if [x = y]`^{3} `then [skip]`^{4} `else [skip]`^{5}
- Interesting property: after termination, `x` $\neq$ `y`, i.e., label 4 unreachable
- Initial abstraction: $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) counterexample:
  $\langle 0, \text{true}\rangle \Rightarrow \langle 1, \text{true}\rangle \Rightarrow \langle 2, \text{true}\rangle \Rightarrow \langle 3, \text{true}\rangle \Rightarrow \langle 4, \text{true}\rangle$
- Forward construction of predicates:
  - $p_0 := \text{true}$
  - (asgn) $p_i := (p_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies p_1 := (p_0[\texttt{x} \mapsto \texttt{x'}] \wedge \texttt{x=z}[\texttt{x} \mapsto \texttt{x'}] \equiv (\texttt{x=z})$
  - (asgn) $p_i := (p_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies p_2 := (p_1[\texttt{z} \mapsto \texttt{z'}] \wedge \texttt{z=z+1}[\texttt{z} \mapsto \texttt{z'}] \equiv (\texttt{x=z'} \wedge \texttt{z=z'+1})$
  - (asgn) $p_i := (p_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
    $\implies p_3 := (p_2[\texttt{y} \mapsto \texttt{y'}] \wedge \texttt{y=z}[\texttt{y} \mapsto \texttt{y'}] \equiv (\texttt{x=z'} \wedge \texttt{z=z'+1} \wedge \texttt{y=z})$
  - (if1) $p_i := p_{i-1} \wedge b$
    $\implies p_4 := p_3 \wedge (\texttt{x=y}) \equiv (\texttt{x=z'} \wedge \texttt{z=z'+1} \wedge \texttt{y=z} \wedge \texttt{x=y}) \equiv \text{false}$

Abstraction Refinement I

Abstraction refinement step:

- Using $p_1, \ldots, p_{k-1}$ as computed before, let $P' := P \cup \{p_1, \ldots, p_{k-1}\}$
- Refine $Abs(P)$ to $Abs(P')$

Lemma 18.4

After refinement, the spurious counterexample $\langle c_0, \mathrm{true}\rangle \Rightarrow \langle c_1, q_1\rangle \Rightarrow \ldots \Rightarrow \langle c_k, q_k\rangle$ *with* $q_k \not\models \mathrm{false}$ *does not exist anymore.*

Proof.

omitted □

Example 18.5 (cf. Example 18.3)

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

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

Another Example: Multiplication

Example 18.6

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

Where CEGAR Fails

Example 18.7

- Let $c_0 :=$ [x := a]^0;
```
[y := b]^1;
while [¬(x = 0)]^2 do
  [x := x - 1]^3;
  [y := y - 1]^4;
if [a = b ∧ ¬(y = 0)]^5 then
  [skip]^6;
else
  [skip]^7;
```
- Interesting property: label 6 unreachable
- Initial abstraction: $P = \emptyset$ ($\implies$ $Abs(P) = \{\text{true}, \text{false}\}$)
- Abstraction refinement: on the board
- Observation: iteration yields predicates of the form x = a-k and y = b-k for all $k \in \mathbb{N}$
- Actually required: loop invariant a = b $\implies$ x = y, but x = y not generated in CEGAR loop

Outline

1. Repetition: Predicate Abstraction
2. Counterexample-Guided Abstraction Refinement
3. Final Remarks

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
- **Formally:** if $p \models r$ and $r \models q$ with $Var_r \subseteq Var_p \cup Var_q$, then r is called a Craig interpolant of p and q
- **Example 18.3:**

$$
\begin{aligned}
\langle \texttt{x:=z;}\ldots, \text{true}\rangle &\Rightarrow \langle \texttt{z:=z+1;}\ldots, \texttt{x=z}\rangle \\
&\Rightarrow \langle \texttt{y:=z;}\ldots, \texttt{x=z-1}\rangle \\
&\Rightarrow \langle \texttt{if x=y}\ldots, \texttt{x=y-1}\rangle \\
&\Rightarrow \langle \texttt{skip}, \text{false}\rangle
\end{aligned}
$$

A CEGAR Implementation: BLAST

- Berkeley Lazy Abstraction Software Verification Tool
- Software model checker for C programs
- Verifies that software satisfies behavioral requirements of associated interfaces
- Uses CEGAR with Craig interpolation
- Sucessfully applied to C programs with $> 130,000$ LOC
  - T.A. Henzinger, R. Jhala, R. Majumdar, K.L. McMillan: *Abstractions from Proofs*, Proc. POPL 2004, 232–244
- WWW: `http://mtc.epfl.ch/software-tools/blast/`