Static Program Analysis

Lecture 16: Abstract Interpretation V (Application Example: 16-Bit Multiplication)

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

Abstract Semantics of WHILE I

Reminder: abstract domain is $Abs := 2^{Var \to A}$

Definition (Abstract execution relation for statements)

If $c \in Cmd$ and $abs \in Abs$, then $\langle c, abs \rangle$ is called an abstract configuration. The abstract execution relation is defined by the following rules:

$$\text{(skip)}\ \frac{}{\langle \texttt{skip}, abs \rangle \Rightarrow abs}$$

$$\text{(asgn)}\ \frac{}{\langle x \texttt{ := } a, abs \rangle \Rightarrow \{\rho[x \mapsto a'] \mid \rho \in abs, a' \in val^{\#}_{\rho}(a)\}}$$

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

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

Abstract Semantics of WHILE II

Definition (Abstract execution relation for statements; cont.)

$$(\mathrm{if1})\ \frac{\exists \rho \in abs : \mathsf{true} \in val^{\#}_{\rho}(b)}{\begin{array}{l}\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, abs\rangle \\ \Rightarrow \langle c_1, abs \setminus \{\rho \in abs \mid val^{\#}_{\rho}(b) = \{\mathsf{false}\}\}\rangle\end{array}}$$

$$(\mathrm{if2})\ \frac{\exists \rho \in abs : \mathsf{false} \in val^{\#}_{\rho}(b)}{\begin{array}{l}\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, abs\rangle \\ \Rightarrow \langle c_2, abs \setminus \{\rho \in abs \mid val^{\#}_{\rho}(b) = \{\mathsf{true}\}\}\rangle\end{array}}$$

$$(\mathrm{wh1})\ \frac{\exists \rho \in abs : \mathsf{true} \in val^{\#}_{\rho}(b)}{\begin{array}{l}\langle \texttt{while}\ b\ \texttt{do}\ c, abs\rangle \\ \Rightarrow \langle c\texttt{;while}\ b\ \texttt{do}\ c, abs \setminus \{\rho \in abs \mid val^{\#}_{\rho}(b) = \{\mathsf{false}\}\}\rangle\end{array}}$$

$$(\mathrm{wh2})\ \frac{\exists \rho \in abs : \mathsf{false} \in val^{\#}_{\rho}(b)}{\langle \texttt{while}\ b\ \texttt{do}\ c, abs\rangle \Rightarrow abs \setminus \{\rho \in abs \mid val^{\#}_{\rho}(b) = \{\mathsf{true}\}\}}$$

Abstract Semantics of WHILE III

Definition (Abstract transition function)

The abstract transition function is defined by the family of mappings

$$\mathrm{next}^{\#}_{c,c'} : \mathit{Abs} \to \mathit{Abs},$$

given by

$$\begin{aligned}\mathrm{next}^{\#}_{c,c'}(\mathit{abs}) &:= \bigsqcup\{\mathit{abs}' \in \mathit{Abs} \mid \langle c, \mathit{abs}\rangle \Rightarrow \langle c', \mathit{abs}'\rangle\}\\ \mathrm{next}^{\#}_{c,\downarrow}(\mathit{abs}) &:= \bigsqcup\{\mathit{abs}' \in \mathit{Abs} \mid \langle c, \mathit{abs}\rangle \Rightarrow \mathit{abs}'\}\end{aligned}$$

Theorem (Soundness of abstract semantics)

For each $c \in \mathit{Cmd}$ and $c' \in \mathit{Cmd} \cup \{\downarrow\}$, $\mathrm{next}^{\#}_{c,c'}$ is a safe approximation of $\mathrm{next}_{c,c'}$, i.e., for every $\mathit{abs} \in \mathit{Abs}$,

$$\alpha(\mathrm{next}_{c,c'}(\gamma(\mathit{abs}))) \sqsubseteq \mathrm{next}^{\#}_{c,c'}(\mathit{abs}).$$

Outline

A 16-Bit Multiplier

Example 16.1 (16-bit multiplier)

```
c = [out := 0]^1;
    [ovf := 0]^2;
    while [¬(f1=0) ∧ ovf=0]^3 do
      if [lsb(f1)=1]^4 then
        [(ovf,out) := (out:17)+f2]^5;
      else
        [skip]^6;
      [f1 := f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2) := (f2:17)<<1]^9;
      else
        [skip]^10;
```

- f1, f2: 16-bit input factors
- out: 16-bit result
- ovf: overflow bit
- lsb(z): least significant bit of z
- (z:k): extension of z to k bits by adding leading zeros
- (x,y):=z: simultaneous assignment with split of z
- <<1/>>1: left/right shift

Procedure: in each iteration,

1. if LSB of f1 is set (4), add f2 to out (5)
2. shift f1 right (7)
3. shift f2 left (9)

Expected result: if $\langle c, \sigma \rangle \rightarrow^+ \sigma'$, then

- $\sigma'(\texttt{out}) = \sigma(\texttt{f1}) \cdot \sigma(\texttt{f2})$ or
- $\sigma'(\texttt{ovf}) = 1$

(termination is trivial)

Example run: on the board

The Abstraction

(see E.M. Clarke, O. Grumberg, D.A. Peled: *Model Checking*, MIT Press, 1999, pp. 205)

- `f1`: no abstraction (as `f1` controls multiplication)
- `f2`: congruence modulo m
  (for specific values of m – see Theorem 16.4)
  - extraction function: $\beta : \mathbb{Z} \to \{0, \ldots, m-1\} : z \mapsto z \bmod m$
    (see Exercise 6.1)
  - congruence: $z_1 \equiv z_2 \pmod{m}$ iff $z_1 \bmod m = z_2 \bmod m$
- `out`: congruence modulo m
- `ovf`: no abstraction (single bit)

Lemma 16.2 (Properties of modulo congruence)

For every $z_1, z_2 \in \mathbb{Z}$ *and* $m \geq 1$,

$$\begin{aligned}(z_1 + z_2) \bmod m &\equiv ((z_1 \bmod m) + (z_2 \bmod m)) \bmod m\\ (z_1 - z_2) \bmod m &\equiv ((z_1 \bmod m) - (z_2 \bmod m)) \bmod m\\ (z_1 \cdot z_2) \bmod m &\equiv ((z_1 \bmod m) \cdot (z_2 \bmod m)) \bmod m\end{aligned}$$

Thus: modulo value of expression determined by modulo values of subexpressions

Abstract Interpretation of Multiplier

Example 16.3 (Abstraction of 16-bit multiplier (cf. Example 16.1))

Abstract execution for

- `f1` $= 101_2$ $(= 5)$
- `f2` $= 1001010_2$ $(= 74)$
- $m = 5$, $74 \bmod 5 = 4$
- `out`, `ovf` initially undefined

$\Longrightarrow$ initial abstract value:

$$\begin{aligned} abs = \{[&\texttt{f1} \mapsto 101_2, \texttt{f2} \mapsto 4, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid \\ &r \in \{0, \ldots, 4\}, b \in \mathbb{B}\} \end{aligned}$$

First transitions: on the board

Ensuring Completeness I

Theorem 16.4 (Chinese Remainder Theorem)

Let $m_1, \ldots, m_k \geq 1$ be pairwise relatively prime (i.e., $\gcd(m_i, m_j) = 1$ for $1 \leq i < j \leq k$). Let $m := m_1 \cdot \ldots \cdot m_k$, and let $z_1, \ldots, z_k \in \mathbb{Z}$. Then there is a unique $z \in \mathbb{Z}$ such that

$$0 \leq z < m \quad \textit{and} \quad z \equiv z_i \pmod{m_i} \textit{ for all } i \in \{1, \ldots, k\}.$$

Application: for fixed initial (abstract) value of `f1` and `f2`,

- z = concrete final value of `out`
- z_i = abstract final value of `out` $\pmod{m_i}$
- $k := 5,\ m_1 := 5,\ m_2 := 7,\ m_3 := 9,\ m_4 := 11,\ m_5 := 32$ (thus $m = 5 \cdot 7 \cdot 9 \cdot 11 \cdot 32 = 110880 > 2^{16}$)
- Theorem 16.4 yields unique $z < m$ with $z \equiv z_i \pmod{m_i}$
- $m > 2^{16} \implies z$ is correct result of multiplication (see next slide)
- thus termination implies correct result or overflow

Efficiency:

- Exhaustive testing: $2^{16} \cdot 2^{16} = 2^{32} = 4.29 \cdot 10^9$ runs
- Abstract interpretation: $2^{16} \cdot (5 + 7 + 9 + 11 + 32) = 4.19 \cdot 10^6$ runs

Ensuring Completeness II

Proof.

To show: $\forall y_1, y_2 \in \mathbb{B}^{16}, \sigma, \sigma' \in \Sigma :$
$\sigma(\texttt{f1}) = y_1, \sigma(\texttt{f2}) = y_2, \langle c, \sigma\rangle \rightarrow^+ \sigma', \sigma'(\texttt{ovf}) = 0$
$\implies \sigma'(\texttt{out}) = y_1 \cdot y_2$

Known: $\forall i \in \{1, \ldots, 5\}, y_1, y_2 \in \mathbb{B}^{16}, abs, abs' \in Abs :$
$abs = \{[\texttt{f1} \mapsto y_1, \texttt{f2} \mapsto y_2^\#, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid$
$r \in \{0, \ldots, m_i - 1\}, b \in \mathbb{B}\}, \langle c, abs\rangle \Rightarrow^+ abs'$
$$\implies \left(\forall \rho' \in abs' : \rho'(\texttt{ovf}) = 0 \implies \rho'(\texttt{out}) \overset{(*)}{=} (y_1 \cdot y_2^\#)^\#\right)$$
(where $x^\# := x \bmod m_i$)

Proof:

- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\texttt{f1}) = y_1$, $\sigma(\texttt{f2}) = y_2$, $\langle c, \sigma\rangle \rightarrow^+ \sigma'$, $\sigma'(\texttt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^\#$ for $i \in \{1, \ldots, 5\}$
- Theorem 16.4 yields unique $z < m$ such that $z \equiv z_i$ (mod m_i) for all $i \in \{1, \ldots, 5\}$
- On the other hand, correctness of modulo abstraction implies $\rho'(\texttt{ovf}) = 0$ and

$$\begin{aligned}(\sigma'(\texttt{out}))^\# &= \rho'(\texttt{out}) && \text{(correctness of abstraction)}\\ &= (y_1 \cdot y_2^\#)^\# && (*)\\ &= (y_1 \cdot y_2)^\# && \text{(Lemma 16.2)}\end{aligned}$$

$\implies \sigma'(\texttt{out}) = z = y_1 \cdot y_2$