Static Program Analysis

Lecture 14: Abstract Interpretation IV (Application Example: 16-Bit Multiplication)

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

Outline

1. Recap: Abstract Semantics of WHILE
2. Correctness of Abstract Semantics
3. Application Example: 16-Bit Multiplication

Safe Approximation of Execution Relation

- **Reminder:** abstraction determined by Galois connection (α, γ) with $\alpha : L \to M$ and $\gamma : M \to L$
  - here: $L := 2^\Sigma$, M not fixed (usually $M = Var \to \ldots$ or $M = 2^{Var \to \ldots}$)
  - write Abs in place of M
  - thus $\alpha : 2^\Sigma \to Abs$ and $\gamma : Abs \to 2^\Sigma$
- Yields abstract semantics:

Definition (Abstract semantics of WHILE)

Given $\alpha : 2^\Sigma \to Abs$, an abstract semantics is defined by a family of functions

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

where $c \in Cmd$, $c' \in Cmd \cup \{\downarrow\}$, and each $\text{next}^\#_{c,c'}$ is a safe approximation of $\text{next}_{c,c'}$, i.e.,

$$\alpha(\text{next}_{c,c'}(\gamma(abs))) \sqsubseteq_{Abs} \text{next}^\#_{c,c'}(abs)$$

for every $abs \in Abs$.

Notation: $\langle c, abs \rangle \Rightarrow \langle c', abs' \rangle$ for $\text{next}^\#_{c,c'}(abs) = abs'$.

Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete elements
- If $L = 2^C$ and $M = 2^A$ with $\sqsubseteq_L = \sqsubseteq_M = \subseteq$, then $\beta : C \to A$ is called an extraction function
- β determines Galois connection (α, γ) where

$$\alpha : L \to M : l \mapsto \beta(l) \; (= \{\beta(c) \mid c \in l\})$$
$$\gamma : M \to L : m \mapsto \beta^{-1}(m) \; (= \{c \in C \mid \beta(c) \in m\})$$

Example

1. Parity abstraction (cf. Example 11.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ where
$$\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$$
2. Sign abstraction (cf. Example 11.3): $\beta : \mathbb{Z} \to \{+, -, 0\}$ with $\beta = \text{sgn}$
3. Interval abstraction (cf. Example 11.4): not definable by extraction function (as Int is not of the form 2^A)

Abstract Program States

Now: take values of variables into account

Definition (Abstract program state)

Let $\beta : \mathbb{Z} \to A$ be an extraction function.

- An abstract (program) state is an element of the set
$$\{\rho \mid \rho : Var \to A\},$$
called the abstract state space.
- The abstract domain is denoted by $Abs := 2^{Var \to A}$.
- The abstraction function $\alpha : 2^{\Sigma} \to Abs$ is given by
$$\alpha(S) := \{\beta \circ \sigma \mid \sigma \in S\}$$
for every $S \subseteq \Sigma$.

Abstract Evaluation of Expressions

Definition (Abstract evaluation functions)

Let $\rho : Var \to A$ be an abstract state.

1. $val^{\#}_{\rho} : AExp \to 2^{A}$ is determined by (f arithmetic operation)

$$
\begin{aligned}
val^{\#}_{\rho}(z) &:= \{\beta(z)\} \\
val^{\#}_{\rho}(x) &:= \{\rho(x)\} \\
val^{\#}_{\rho}(f(a_1, \ldots, a_n)) &:= f^{\#}(val^{\#}_{\rho}(a_1), \ldots, val^{\#}_{\rho}(a_n))
\end{aligned}
$$

2. $val^{\#}_{\rho} : BExp \to 2^{\mathbb{B}}$ is determined by (g/h relational/Boolean op.)

$$
\begin{aligned}
val^{\#}_{\rho}(t) &:= \{t\} \\
val^{\#}_{\rho}(g(a_1, \ldots, a_n)) &:= g^{\#}(val^{\#}_{\rho}(a_1), \ldots, val^{\#}_{\rho}(a_n)) \\
val^{\#}_{\rho}(h(b_1, \ldots, b_n)) &:= h^{\#}(val^{\#}_{\rho}(b_1), \ldots, val^{\#}_{\rho}(b_n))
\end{aligned}
$$

Example (Sign abstraction)

Let $\rho(\mathtt{x}) = +$ and $\rho(\mathtt{y}) = -$.

1. $val^{\#}_{\rho}(\mathtt{2 * x + y}) = \{+, -, 0\}$
2. $val^{\#}_{\rho}(\neg(\mathtt{x + 1 > y})) = \{\mathsf{false}\}$

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:

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

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

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

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

Abstract Semantics of WHILE II

Definition (Abstract execution relation for statements; cont.)

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

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

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

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

Abstract Semantics of WHILE III

Definition (Abstract transition function)

The abstract transition function is defined by the family of mappings

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

given by

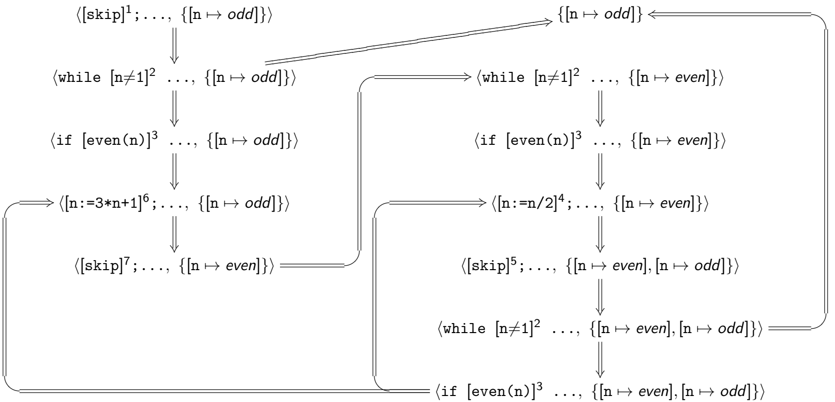
$$\mathrm{next}^{\#}_{c,c'}(abs) := \bigsqcup \{abs' \in Abs \mid \langle c, abs\rangle \Rightarrow \langle c', abs'\rangle\}$$

Example (Hailstone Sequences; cf. Example 13.1)

$[\texttt{skip}]^1$;
while $[\neg(\texttt{n = 1})]^2$ do
  if $[\texttt{even(n)}]^3$ then
    $[\texttt{n := n / 2}]^4$;$[\texttt{skip}]^5$;
  else
    $[\texttt{n := 3 * n + 1}]^6$;$[\texttt{skip}]^7$;

Execution relation with parity abstraction: see following slide (courtesy B. König)

Abstrakte Interpretation von Hailstone



Outline

Correctness of Abstract Semantics

Theorem 14.1 (Soundness of abstract semantics)

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

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

Correctness of Abstract Semantics

Theorem 14.1 (Soundness of abstract semantics)

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

$$\alpha(\text{next}_{c,c'}(\gamma(abs))) \subseteq \text{next}^\#_{c,c'}(abs).$$

The soundness proof employs the following auxiliary lemma.

Lemma 14.2 (Soundness of abstract evaluation)

Let $\beta : \mathbb{Z} \to A$ *be an extraction function.*

1. *For every* $a \in AExp$ *and* $\sigma \in \Sigma$, $\beta(val_\sigma(a)) \in val^\#_{\beta\circ\sigma}(a)$.
2. *For every* $b \in BExp$ *and* $\sigma \in \Sigma$, $val_\sigma(b) \in val^\#_{\beta\circ\sigma}(b)$.

Proof (Lemma 14.2).

omitted □

Correctness of Abstract Semantics

Theorem 14.1 (Soundness of abstract semantics)

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

$$\alpha(\mathsf{next}_{c,c'}(\gamma(abs))) \subseteq \mathsf{next}^{\#}_{c,c'}(abs).$$

The soundness proof employs the following auxiliary lemma.

Lemma 14.2 (Soundness of abstract evaluation)

Let $\beta : \mathbb{Z} \to A$ *be an extraction function.*

1. *For every* $a \in AExp$ *and* $\sigma \in \Sigma$, $\beta(val_\sigma(a)) \in val^{\#}_{\beta\circ\sigma}(a)$.
2. *For every* $b \in BExp$ *and* $\sigma \in \Sigma$, $val_\sigma(b) \in val^{\#}_{\beta\circ\sigma}(b)$.

Proof (Lemma 14.2).

omitted □

Proof (Theorem 14.1).

on the board □

Outline

A 16-Bit Multiplier

Example 14.3 (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)

A 16-Bit Multiplier

Example 14.3 (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 $\langle c, \sigma\rangle \to^+ \langle\downarrow, \sigma'\rangle$, 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 14.6)
  - extraction function: $\beta : \mathbb{Z} \to \{0, \ldots, m-1\} : z \mapsto z \bmod m$
    (see Exercise 9.1)
  - 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)

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 14.6)
  - extraction function: $\beta : \mathbb{Z} \to \{0, \ldots, m-1\} : z \mapsto z \bmod m$
    (see Exercise 9.1)
  - 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 14.4 (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 14.5 (Abstraction of 16-bit multiplier; cf. Example 14.3)

Abstract execution for

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

$\Longrightarrow$ initial abstract value:

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

First transitions: on the board

Abstract Interpretation of Multiplier

Example 14.5 (Abstraction of 16-bit multiplier; cf. Example 14.3)

Abstract execution for

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

$\Longrightarrow$ initial abstract value:

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

First transitions: on the board

Problem: choose which values of m to deduce correctness of concrete result from correctness of all abstract results?

Ensuring Correctness I

Theorem 14.6 (Chinese Remainder Theorem; without proof)

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*

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

Ensuring Correctness I

Theorem 14.6 (Chinese Remainder Theorem; without proof)

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

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

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 14.6 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

Ensuring Correctness I

Theorem 14.6 (Chinese Remainder Theorem; without proof)

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*

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

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 14.6 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 Correctness II

Proof (Correctness of abstraction).

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

Ensuring Correctness II

Proof (Correctness of abstraction).

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

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

Proof (Correctness of abstraction).

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

Known: $\forall i \in \{1, \ldots, 5\}, y_1, y_2 \in \mathbb{B}^{16}, abs, abs' \in Abs :$

$$abs = \{[\mathtt{f1} \mapsto y_1, \mathtt{f2} \mapsto y_2^{\#}, \mathtt{out} \mapsto r, \mathtt{ovf} \mapsto b] \mid r \in \{0, \ldots, m_i - 1\}, b \in \mathbb{B}\}, \langle c, abs \rangle \Rightarrow^+ \langle \downarrow, abs' \rangle$$
$$\implies \left(\forall \rho' \in abs' : \rho'(\mathtt{ovf}) = 0 \implies \rho'(\mathtt{out}) \stackrel{(*)}{=} (y_1 \cdot y_2^{\#})^{\#} \right)$$

(where $x^{\#} := x \bmod m_i$)

Proof:

- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\mathtt{f1}) = y_1$, $\sigma(\mathtt{f2}) = y_2$, $\langle c, \sigma \rangle \to^+ \langle \downarrow, \sigma' \rangle$, $\sigma'(\mathtt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^{\#}$ for $i \in \{1, \ldots, 5\}$

Ensuring Correctness II

Proof (Correctness of abstraction).

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

Known: $\forall i \in \{1, \ldots, 5\}, y_1, y_2 \in \mathbb{B}^{16}, abs, abs' \in Abs :$

$$abs = \{[\texttt{f1} \mapsto y_1, \texttt{f2} \mapsto y_2^{\#}, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid r \in \{0, \ldots, m_i - 1\}, b \in \mathbb{B}\}, \langle c, abs \rangle \Rightarrow^{+} \langle \downarrow, abs' \rangle$$

$$\implies \left(\forall \rho' \in abs' : \rho'(\texttt{ovf}) = 0 \implies \rho'(\texttt{out}) \stackrel{(*)}{=} (y_1 \cdot y_2^{\#})^{\#} \right)$$

(where $x^{\#} := x \bmod m_i$)

Proof:

- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\texttt{f1}) = y_1$, $\sigma(\texttt{f2}) = y_2$, $\langle c, \sigma \rangle \to^{+} \langle \downarrow, \sigma' \rangle$, $\sigma'(\texttt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^{\#}$ for $i \in \{1, \ldots, 5\}$
- Theorem 14.6 yields unique $z < m$ such that $z \equiv z_i \pmod{m_i}$ for all $i \in \{1, \ldots, 5\}$

Ensuring Correctness II

Proof (Correctness of abstraction).

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

Known: $\forall i \in \{1, \ldots, 5\}, y_1, y_2 \in \mathbb{B}^{16}, abs, abs' \in Abs :$

$$abs = \{[\mathtt{f1} \mapsto y_1, \mathtt{f2} \mapsto y_2^{\#}, \mathtt{out} \mapsto r, \mathtt{ovf} \mapsto b] \mid r \in \{0, \ldots, m_i - 1\}, b \in \mathbb{B}\}, \langle c, abs \rangle \Rightarrow^+ \langle \downarrow, abs' \rangle$$

$$\implies \left(\forall \rho' \in abs' : \rho'(\mathtt{ovf}) = 0 \implies \rho'(\mathtt{out}) \overset{(*)}{=} (y_1 \cdot y_2^{\#})^{\#} \right)$$

(where $x^{\#} := x \bmod m_i$)

Proof:

- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\mathtt{f1}) = y_1$, $\sigma(\mathtt{f2}) = y_2$, $\langle c, \sigma \rangle \rightarrow^+ \langle \downarrow, \sigma' \rangle$, $\sigma'(\mathtt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^{\#}$ for $i \in \{1, \ldots, 5\}$
- Theorem 14.6 yields unique $z < m$ such that $z \equiv z_i \pmod{m_i}$ for all $i \in \{1, \ldots, 5\}$
- On the other hand, correctness of modulo abstraction implies $\rho'(\mathtt{ovf}) = 0$ and

$$\begin{aligned} (\sigma'(\mathtt{out}))^{\#} &= \rho'(\mathtt{out}) && \text{(correctness of abstraction)} \\ &= (y_1 \cdot y_2^{\#})^{\#} && (*) \\ &= (y_1 \cdot y_2)^{\#} && \text{(Lemma 14.4)} \end{aligned}$$

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

□