Static Program Analysis

Lecture 13: Abstract Interpretation IV (Application Example: 16-Bit Multiplication)

Summer Semester 2018

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Outline of Lecture 13

Recap: Abstract Semantics of WHILE

Application Example: 16-Bit Multiplication

Overview of Numerical Abstraction Domains

Recap: Abstract Semantics of WHILE

Extraction Functions

Assumption: abstraction determined by pointwise mapping of concrete values

Definition (Extraction function)

- 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) with

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

Example

1. Parity abstraction (cf. Ex. 10.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ 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. Ex. 10.3): $\beta : \mathbb{Z} \to \{+, -, 0\}$ with $\beta = \text{sgn}$
3. Interval abstraction (cf. Ex. 10.4): not definable by extraction function (as *Int* is not of the form 2^A)

Recap: Abstract Semantics of WHILE

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 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; continued)

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

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

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

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

Abstract Semantics of WHILE III

Definition (Abstract transition function)

The abstract transition function is defined by the family of mappings

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

given by $$\text{next}^{\#}_{c,c'}(\mathit{abs}) := \bigcup \{ \mathit{abs}' \in \mathit{Abs} \mid \langle c, \mathit{abs} \rangle \Rightarrow \langle c', \mathit{abs}' \rangle \}$$

Recap: Abstract Semantics of WHILE

Correctness of Abstract Semantics

Theorem (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 (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 12.14).

omitted □

Outline of Lecture 13

Recap: Abstract Semantics of WHILE

Application Example: 16-Bit Multiplication

Overview of Numerical Abstraction Domains

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Inputs:

- `f1`, `f2`: 16-bit input factors

Outputs:

- `out`: 16-bit result
- `ovf`: overflow bit

Operations:

- `lsb(`z`)`: least significant bit of z
- `(`z`:`k`)`: extension of z to k bits by adding leading zeros
- `(`x`,`y`) :=` z: simultaneous assignment to x and y with split of z
- `<<1` / `>>1`: left/right shift by 1 bit

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

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)

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

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)

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

/	f1	f2	out	ovf
1	0101	0011	?	?

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf
1	0101	0011	?	?
2	0101	0011	0000	?

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while } [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3 \texttt{ do} \\
\quad\quad \texttt{if } [\texttt{lsb(f1)=1}]^4 \texttt{ then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if } [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8 \texttt{ then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf
1	0101	0011	?	?
2	0101	0011	0000	?
3	0101	0011	0000	0

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

/	f1	f2	out	ovf
1	0101	0011	?	?
2	0101	0011	0000	?
3	0101	0011	0000	0
4	0101	0011	0000	0

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

/	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

/	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	(out : 5 = 00000)
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	(f2 : 5 = 00011)
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	(f2 : 5 = 00110)
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	(out : 5 = 00011)

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	$(\texttt{out} : 5 = 00011)$
7	0001	1100	1111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

/	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	$(\texttt{out} : 5 = 00011)$
7	0001	1100	1111	0	
8	0000	1100	1111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	$(\texttt{out} : 5 = 00011)$
7	0001	1100	1111	0	
8	0000	1100	1111	0	
10	0000	1100	1111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (4-bit multiplication)

/	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	$(\texttt{out} : 5 = 00011)$
7	0001	1100	1111	0	
8	0000	1100	1111	0	
10	0000	1100	1111	0	
3	0000	1100	1111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (4-bit multiplication)

l	f1	f2	out	ovf	
1	0101	0011	?	?	
2	0101	0011	0000	?	
3	0101	0011	0000	0	
4	0101	0011	0000	0	
5	0101	0011	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0011	0011	0	
8	0010	0011	0011	0	
9	0010	0011	0011	0	$(\texttt{f2} : 5 = 00011)$
3	0010	0110	0011	0	
4	0010	0110	0011	0	
6	0010	0110	0011	0	
7	0010	0110	0011	0	
8	0001	0110	0011	0	
9	0001	0110	0011	0	$(\texttt{f2} : 5 = 00110)$
3	0001	1100	0011	0	
4	0001	1100	0011	0	
5	0001	1100	0011	0	$(\texttt{out} : 5 = 00011)$
7	0001	1100	1111	0	
8	0000	1100	1111	0	
10	0000	1100	1111	0	
3	0000	1100	1111	0	
↓	0000	1100	1111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

/	f1	f2	out	ovf
1	0101	0111	?	?

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

/	f1	f2	out	ovf
1	0101	0111	?	?
2	0101	0111	0000	?

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while } [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3 \texttt{ do} \\
\quad\quad \texttt{if } [\texttt{lsb(f1)=1}]^4 \texttt{ then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if } [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8 \texttt{ then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf
1	0101	0111	?	?
2	0101	0111	0000	?
3	0101	0111	0000	0

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf
1	0101	0111	?	?
2	0101	0111	0000	?
3	0101	0111	0000	0
4	0101	0111	0000	0

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (with overflow)

/	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	(out : 5 = 00000)
7	0101	0111	0111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (with overflow)

/	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	(out : 5 = 00000)
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	(f2 : 5 = 00111)
3	0010	1110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (with overflow)

/	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	(out : 5 = 00000)
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	(f2 : 5 = 00111)
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	
7	0010	1110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	
7	0010	1110	0011	0	
8	0001	1110	0011	0	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end}; \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	
7	0010	1110	0011	0	
8	0001	1110	0011	0	
9	0001	1110	0011	0	$(\texttt{f2} : 5 = 01110)$

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (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
      end;
      [f1:=f1>>1]^7;
      if [¬(f1=0) ∧ ovf=0]^8 then
        [(ovf,f2):=(f2:17)<<1]^9
      else
        [skip]^10
      end
    end
```

Example run: (with overflow)

/	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	
7	0010	1110	0011	0	
8	0001	1110	0011	0	
9	0001	1110	0011	0	$(\texttt{f2} : 5 = 01110)$
3	0001	1100	0011	1	

Application Example: 16-Bit Multiplication

A 16-Bit Multiplier

Example 13.1 (16-bit multiplier)

$$
\begin{array}{l}
c = [\texttt{out:=0}]^1;[\texttt{ovf:=0}]^2; \\
\quad \texttt{while}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^3\ \texttt{do} \\
\quad\quad \texttt{if}\ [\texttt{lsb(f1)=1}]^4\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,out):=(out:17)+f2}]^5 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^6 \\
\quad\quad \texttt{end;} \\
\quad\quad [\texttt{f1:=f1>>1}]^7; \\
\quad\quad \texttt{if}\ [\neg(\texttt{f1=0}) \wedge \texttt{ovf=0}]^8\ \texttt{then} \\
\quad\quad\quad [\texttt{(ovf,f2):=(f2:17)<<1}]^9 \\
\quad\quad \texttt{else} \\
\quad\quad\quad [\texttt{skip}]^{10} \\
\quad\quad \texttt{end} \\
\quad \texttt{end}
\end{array}
$$

Example run: (with overflow)

l	f1	f2	out	ovf	
1	0101	0111	?	?	
2	0101	0111	0000	?	
3	0101	0111	0000	0	
4	0101	0111	0000	0	
5	0101	0111	0000	0	$(\texttt{out} : 5 = 00000)$
7	0101	0111	0111	0	
8	0010	0111	0111	0	
9	0010	0111	0111	0	$(\texttt{f2} : 5 = 00111)$
3	0010	1110	0011	0	
4	0010	1110	0011	0	
6	0010	1110	0011	0	
7	0010	1110	0011	0	
8	0001	1110	0011	0	
9	0001	1110	0011	0	$(\texttt{f2} : 5 = 01110)$
3	0001	1100	0011	1	
↓	0001	1100	0011	1	

Application Example: 16-Bit Multiplication

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 $m \geq 2$ – see Theorem 13.4)
  - extraction function: $\beta : \mathbb{Z} \to \{0, \ldots, m-1\} : z \mapsto z \bmod m$
  - 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)

Application Example: 16-Bit Multiplication

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 $m \geq 2$ – see Theorem 13.4)
  - extraction function: $\beta : \mathbb{Z} \to \{0, \ldots, m-1\} : z \mapsto z \bmod m$
  - 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 13.2 (Properties of modulo congruence)

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

$$(z_1 + z_2) \bmod m \equiv ((z_1 \bmod m) + (z_2 \bmod m)) \bmod m$$
$$(z_1 - z_2) \bmod m \equiv ((z_1 \bmod m) - (z_2 \bmod m)) \bmod m$$
$$(z_1 \cdot z_2) \bmod m \equiv ((z_1 \bmod m) \cdot (z_2 \bmod m)) \bmod m$$

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

Abstract Interpretation of Multiplier

Example 13.3 (Abstraction of 16-bit multiplier; cf. Example 13.1)

Abstract execution for

- $\texttt{f1} = 101_2\ (= 5)$, $\texttt{f2} = 1001010_2\ (= 74)$
- $\texttt{out}$, $\texttt{ovf}$ with arbitrary initial values
- $m = 5$

$\Rightarrow$ initial abstract value:

$$abs_0 = \{[\texttt{f1} \mapsto 101_2, \texttt{f2} \mapsto \underbrace{74 \bmod 5}_{4}, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid r \in \{0, \ldots, 4\}, b \in \mathbb{B}\}$$

Abstract Interpretation of Multiplier

Example 13.3 (Abstraction of 16-bit multiplier; cf. Example 13.1)

Abstract execution for

- $\texttt{f1} = 101_2\ (= 5)$, $\texttt{f2} = 1001010_2\ (= 74)$
- $\texttt{out}$, $\texttt{ovf}$ with arbitrary initial values
- $m = 5$

$\Rightarrow$ initial abstract value:

$$abs_0 = \{[\texttt{f1} \mapsto 101_2, \texttt{f2} \mapsto \underbrace{74 \bmod 5}_{4}, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid r \in \{0, \ldots, 4\}, b \in \mathbb{B}\}$$

- first transitions: on the board

Abstract Interpretation of Multiplier

Example 13.3 (Abstraction of 16-bit multiplier; cf. Example 13.1)

Abstract execution for

- $\texttt{f1} = 101_2\ (= 5)$, $\texttt{f2} = 1001010_2\ (= 74)$
- $\texttt{out}$, $\texttt{ovf}$ with arbitrary initial values
- $m = 5$

$\Rightarrow$ initial abstract value:

$$abs_0 = \{[\texttt{f1} \mapsto 101_2, \texttt{f2} \mapsto \underbrace{74 \bmod 5}_{4}, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid r \in \{0, \ldots, 4\}, b \in \mathbb{B}\}$$

- first transitions: on the board
- generally: for all initial (abstract) values of $\texttt{f1}$ and $\texttt{f2}$, abstract results $\langle \downarrow, abs' \rangle$, and $\rho' \in abs'$,

$$\rho'(\texttt{ovf}) = 1 \vee \rho'(\texttt{out}) = (\texttt{f1} \cdot (\texttt{f2} \bmod 5)) \bmod 5$$

Application Example: 16-Bit Multiplication

Abstract Interpretation of Multiplier

Example 13.3 (Abstraction of 16-bit multiplier; cf. Example 13.1)

Abstract execution for

- $\texttt{f1} = 101_2\ (= 5)$, $\texttt{f2} = 1001010_2\ (= 74)$
- $\texttt{out}$, $\texttt{ovf}$ with arbitrary initial values
- $m = 5$

$\Rightarrow$ initial abstract value:

$$abs_0 = \{[\texttt{f1} \mapsto 101_2, \texttt{f2} \mapsto \underbrace{74 \bmod 5}_{4}, \texttt{out} \mapsto r, \texttt{ovf} \mapsto b] \mid r \in \{0, \ldots, 4\}, b \in \mathbb{B}\}$$

- first transitions: on the board
- generally: for all initial (abstract) values of $\texttt{f1}$ and $\texttt{f2}$, abstract results $\langle \downarrow, abs' \rangle$, and $\rho' \in abs'$,

$$\rho'(\texttt{ovf}) = 1 \vee \rho'(\texttt{out}) = (\texttt{f1} \cdot (\texttt{f2} \bmod 5)) \bmod 5$$

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

Ensuring Correctness I

Theorem 13.4 (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 $0 \leq z < m$ and $z \equiv z_i \pmod{m_i}$ for all $i \in \{1, \ldots, k\}$.

Ensuring Correctness I

Theorem 13.4 (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 $0 \leq z < m$ and $z \equiv z_i \pmod{m_i}$ 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 13.4 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

Application Example: 16-Bit Multiplication

Ensuring Correctness I

Theorem 13.4 (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 $0 \leq z < m$ and $z \equiv z_i \pmod{m_i}$ 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 13.4 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 Correctness II

Proof (Correctness of abstraction).

To show: $\forall y_1, y_2 \in \mathbb{B}^{16}, \sigma, \sigma' \in \Sigma : \sigma(\texttt{f1}) = y_1, \sigma(\texttt{f2}) = y_2, \langle c, \sigma \rangle \to^+ \langle \downarrow, \sigma' \rangle$
$\implies \sigma'(\texttt{ovf}) = 1 \vee \sigma'(\texttt{out}) = y_1 \cdot y_2$

Ensuring Correctness II

Proof (Correctness of abstraction).

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

$\implies \sigma'(\texttt{ovf}) = 1 \vee \sigma'(\texttt{out}) = y_1 \cdot y_2$

Known: $\forall i \in \{1, \ldots, 5\}, y_1, y_2 \in \mathbb{B}^{16}, abs, abs' \in Abs : \langle c, abs \rangle \Rightarrow^+ \langle \downarrow, abs' \rangle,$

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

$\implies \left(\forall \rho' \in abs' : \rho'(\texttt{ovf}) = 1 \vee \rho'(\texttt{out}) \overset{(*)}{=} (y_1 \cdot y_2^\#)^\# \right)$ (where $x^\# := x \bmod m_i$)

Ensuring Correctness II

Proof (Correctness of abstraction).

To show: $\forall y_1, y_2 \in \mathbb{B}^{16}, \sigma, \sigma' \in \Sigma : \sigma(\texttt{f1}) = y_1, \sigma(\texttt{f2}) = y_2, \langle c, \sigma \rangle \to^+ \langle \downarrow, \sigma' \rangle$
$\implies \sigma'(\texttt{ovf}) = 1 \vee \sigma'(\texttt{out}) = y_1 \cdot y_2$

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

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

Ensuring Correctness II

Proof (Correctness of abstraction).

To show: $\forall y_1, y_2 \in \mathbb{B}^{16}, \sigma, \sigma' \in \Sigma : \sigma(\texttt{f1}) = y_1, \sigma(\texttt{f2}) = y_2, \langle c, \sigma \rangle \to^+ \langle \downarrow, \sigma' \rangle$
$\implies \sigma'(\texttt{ovf}) = 1 \vee \sigma'(\texttt{out}) = y_1 \cdot y_2$

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

Proof:
- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\texttt{f1}) = y_1$, $\sigma(\texttt{f2}) = y_2$, $\langle c, \sigma \rangle \to^+ \langle \downarrow, \sigma' \rangle$, $\sigma'(\texttt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^\#$ for $i \in \{1, \ldots, 5\}$
- Thm. 13.4 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: $\forall y_1, y_2 \in \mathbb{B}^{16}, \sigma, \sigma' \in \Sigma : \sigma(\texttt{f1}) = y_1, \sigma(\texttt{f2}) = y_2, \langle c, \sigma\rangle \to^+ \langle\downarrow, \sigma'\rangle$
$\implies \sigma'(\texttt{ovf}) = 1 \vee \sigma'(\texttt{out}) = y_1 \cdot y_2$

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

Proof:
- Let $y_1, y_2 \in \mathbb{B}^{16}$, $\sigma(\texttt{f1}) = y_1$, $\sigma(\texttt{f2}) = y_2$, $\langle c, \sigma\rangle \to^+ \langle\downarrow, \sigma'\rangle$, $\sigma'(\texttt{ovf}) = 0$, and $z_i := (y_1 \cdot y_2)^\#$ for $i \in \{1, \ldots, 5\}$
- Thm. 13.4 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 $\rho'(\texttt{ovf}) = 0$ and

$$\begin{aligned}(\sigma'(\texttt{out}))^\# &= \rho'(\texttt{out}) && \text{(correctness of abstraction)}\\ &= (y_1 \cdot y_2^\#)^\# && (*)\\ &= (y_1 \cdot y_2)^\# && \text{(Lemma 13.2)}\end{aligned}$$
$$\implies \sigma'(\texttt{out}) = z = y_1 \cdot y_2$$

□

Outline of Lecture 13

Recap: Abstract Semantics of WHILE

Application Example: 16-Bit Multiplication

Overview of Numerical Abstraction Domains

Non-Relational Abstraction Domains

Here, abstract values are independently referring to single variables.

Example 13.5 (Non-relational domains)

- Signs (cf. Example 10.3): $\mathrm{sgn}(x) = s$ ($x \in \mathit{Var}$, $s \in \{+, -, 0\}$)
- Intervals (cf. Example 10.4): $x \in J$ ($x \in \mathit{Var}$, $J \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}$)
- Parities (cf. Example 10.2): $x \in \mathbb{Z}_p$ ($x \in \mathit{Var}$, $p \in \{\mathrm{even}, \mathrm{odd}\}$)
- Congruences (cf. Lemma 13.2): $x \bmod m = k$ ($x \in \mathit{Var}$, $m > 1$, $k \in \{0, \ldots, m-1\}$)

Non-Relational Abstraction Domains

Here, abstract values are independently referring to single variables.

Example 13.5 (Non-relational domains)

- Signs (cf. Example 10.3): $\mathrm{sgn}(x) = s$ ($x \in \mathit{Var}$, $s \in \{+, -, 0\}$)
- Intervals (cf. Example 10.4): $x \in J$ ($x \in \mathit{Var}$, $J \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}$)
- Parities (cf. Example 10.2): $x \in \mathbb{Z}_p$ ($x \in \mathit{Var}$, $p \in \{\mathrm{even}, \mathrm{odd}\}$)
- Congruences (cf. Lemma 13.2): $x \bmod m = k$ ($x \in \mathit{Var}$, $m > 1$, $k \in \{0, \ldots, m-1\}$)

Observations

- Expressive power:
  - Signs $<$ Intervals (since $+ \cong [1, +\infty]$, ...)
  - Parities $<$ Congruences (since x even $\iff$ $x \bmod 2 = 0$, ...)
  - Intervals and Congruences are “incomparable”
- Congruences can prove disequalities (“$x \neq y$”) but not inequalities (“$x \leq y$”)
  - e.g., $x \bmod m = k_x$, $y \bmod m = k_y$, $k_x \neq k_y$ $\implies$ no zero division in $1/(x-y)$
  - but $x \leq y$ generally not representable
- Non-relational domains efficient to represent and manipulate
- Signs, Parities, and Congruences definable by extraction functions

Relational Abstraction Domains

Here, interdependencies between variables are captured:

Example 13.6 (Relational domains)

- Difference Bound Matrices (DBMs): conjunctions of $x - y \leq c$ and $\pm x \leq c$ ($x, y \in \mathit{Var}$, $c \in \mathbb{Z}$)
- Octagons: conjunctions of $ax + by \leq c$ ($x, y \in \mathit{Var}$, $a, b \in \{-1, 0, 1\}$, $c \in \mathbb{Z}$)
- Octahedra: conjunctions of $a_1 x_1 + \ldots + a_n x_n \leq c$ ($x_i \in \mathit{Var}$, $a_i \in \{-1, 0, 1\}$, $c \in \mathbb{Z}$)
- Polyhedra: conjunctions of $a_1 x_1 + \ldots + a_n x_n \leq c$ ($x_i \in \mathit{Var}$, $a_i \in \mathbb{Z}$, $c \in \mathbb{Z}$)

Relational Abstraction Domains

Here, interdependencies between variables are captured:

Example 13.6 (Relational domains)

- Difference Bound Matrices (DBMs): conjunctions of $x - y \leq c$ and $\pm x \leq c$ ($x, y \in \mathit{Var}$, $c \in \mathbb{Z}$)
- Octagons: conjunctions of $ax + by \leq c$ ($x, y \in \mathit{Var}$, $a, b \in \{-1, 0, 1\}$, $c \in \mathbb{Z}$)
- Octahedra: conjunctions of $a_1 x_1 + \ldots + a_n x_n \leq c$ ($x_i \in \mathit{Var}$, $a_i \in \{-1, 0, 1\}$, $c \in \mathbb{Z}$)
- Polyhedra: conjunctions of $a_1 x_1 + \ldots + a_n x_n \leq c$ ($x_i \in \mathit{Var}$, $a_i \in \mathbb{Z}$, $c \in \mathbb{Z}$)

Observations

- Expressive power:
  - DBMs $<$ Octagons $<$ Octahedra $<$ Polyhedra
  - Intervals $<$ DBMs (since $x \in [c_1, c_2] \iff -x \leq -c_1 \wedge x \leq c_2$)
- Can prove inequalities but not (general) disequalities
- Representation and manipulation generally more involved
  - Polyhedra require computation of convex hulls (exponential in $|\mathit{Var}|$)

Combining Non-Relational and Relational Domains

Linear Congruences combine features of Congruences and Polyhedra:

- Given by conjunctions of

$$(a_1 x_1 + \ldots + a_n x_n) \bmod m = z$$

  ($x_i \in \mathit{Var}$, $a_i \in \mathbb{Z}$, $m > 1$, $z \in \mathbb{Z}$)

- Typical application:

$$2x + 1 \bmod m = k_x, y \bmod m = k_y, k_x \neq k_y \implies \text{no zero division in } 1/(2x + 1 - y)$$

- Again usable for proving disequalities but not inequalities