Static Program Analysis

Lecture 13: Abstract Interpretation III (Abstract Semantics of WHILE)

Winter Semester 2016/17

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-1617/spa/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Lecture 13

Recap: Safe Approximation of Functions and Relations

Systematic Construction of Abstractions

Abstract Semantics of WHILE

Correctness of Abstract Semantics

Recap: Safe Approximation of Functions and Relations

Safe Approximation of Functions I

Definition (Safe approximation)

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $f : L^n \to L$ and $f^\# : M^n \to M$ be functions of rank $n \in \mathbb{N}$. Then $f^\#$ is called a safe approximation of f if, whenever $m_1, \ldots, m_n \in M$,

$$\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \ldots, m_n).$$

Moreover, $f^\#$ is called most precise if the reverse inclusion is also true.

Abstract		**Concrete**
$\vec{m}$	$\xrightarrow{\gamma}$	$\gamma(\vec{m})$
$\downarrow f^\#$		$\downarrow f$
$f^\#(\vec{m}) \sqsupseteq \alpha(f(\gamma(\vec{m})))$	$\xleftarrow{\alpha}$	$f(\gamma(\vec{m}))$

- **Interpretation:** the abstraction $f^\#$ covers all concrete f-results
- **Note:** monotonicity of f and/or $f^\#$ is *not* required (but usually given; see Lemma 12.3)

Encoding Execution Relations by Transition Functions I

- **Reminder:** concrete semantics of WHILE
  - statements $\texttt{skip} \mid x := a \mid c_1 ; c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end} \mid \texttt{while } b \texttt{ do } c \texttt{ end} \in \mathit{Cmd}$
  - states $\Sigma := \{\sigma \mid \sigma : \mathit{Var} \to \mathbb{Z}\}$ (Definition 11.6)
  - execution relation $\to \subseteq (\mathit{Cmd} \times \Sigma) \times ((\mathit{Cmd} \cup \{\downarrow\}) \times \Sigma)$ (Definition 11.9)
- Yields concrete domain $L := (2^\Sigma, \subseteq)$ and concrete transition function:

Definition (Concrete transition function)

The concrete transition function of WHILE is defined by the family of functions

$$\mathrm{next}_{c,c'} : 2^\Sigma \to 2^\Sigma$$

where $c \in \mathit{Cmd}$, $c' \in \mathit{Cmd} \cup \{\downarrow\}$ and, for every $S \subseteq \Sigma$,

$$\mathrm{next}_{c,c'}(S) := \{\sigma' \in \Sigma \mid \exists \sigma \in S : \langle c, \sigma\rangle \to \langle c', \sigma'\rangle\}.$$

Encoding Execution Relations by Transition Functions II

Remarks: next satisfies the following properties

- "Determinism" (cf. Theorem 11.11):
  - for all $c \in Cmd$, $c' \in Cmd \cup \{\downarrow\}$ and $\sigma \in \Sigma$, $|\text{next}_{c,c'}(\{\sigma\})| \leq 1$
  - for all $c \in Cmd$ and $\sigma \in \Sigma$ there exists exactly one $c' \in Cmd \cup \{\downarrow\}$ such that $\text{next}_{c,c'}(\{\sigma\}) \neq \emptyset$
- When is $\text{next}_{c,c'}(S) = \emptyset$? Possible reasons:
  1. $S = \emptyset$
  2. c' is not a possible successor statement of c, e.g.,
     - $c = (\texttt{x := 0})$
     - $c' = \texttt{skip}$
  3. c' is unreachable for all $\sigma \in S$, e.g.,
     - $c = (\texttt{if x = 0 then x := 1 else skip end})$
     - $c' = \texttt{skip}$
     - $\sigma(\texttt{x}) = 0$ for each $\sigma \in S$

Safe Approximation of Execution Relations

Reminder: abstraction determined by Galois connection (α, γ) with $\alpha : L \to M$, $\gamma : M \to L$

- here: $L := 2^{\Sigma}$, M not fixed
- usually $M = Var \to \ldots$ (more efficient) or $M = 2^{Var \to \cdots}$ (more precise)
- write Abs in place of M
- thus $\alpha : 2^{\Sigma} \to Abs$ and $\gamma : Abs \to 2^{\Sigma}$

Definition (Abstract semantics of WHILE)

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

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

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

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

for every $abs \in Abs$ (notation: $\langle c, abs\rangle \Rightarrow \langle c', abs'\rangle$ for $\mathrm{next}^{\#}_{c,c'}(abs) = abs'$).

Outline of Lecture 13

Recap: Safe Approximation of Functions and Relations

Systematic Construction of Abstractions

Abstract Semantics of WHILE

Correctness of Abstract Semantics

Derivation of Abstract Semantics

- **Problem:** most precise safe approximation not always definable

Example 13.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

$$\langle \texttt{if n > 2} \wedge \texttt{x^n + y^n = z^n then n := 1 else n := -1 end}, \{[\mathrm{n}, \mathrm{x}, \mathrm{y}, \mathrm{z} \mapsto +]\}\rangle$$

Derivation of Abstract Semantics

- **Problem:** most precise safe approximation not always definable

Example 13.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

$$\langle \texttt{if n > 2} \wedge \texttt{x^n + y^n = z^n then n := 1 else n := -1 end}, \{[\mathtt{n}, \mathtt{x}, \mathtt{y}, \mathtt{z} \mapsto +]\}\rangle$$

- Result $\mathtt{n} \mapsto +$ possible iff there exist $n > 2$ and $x, y, z \geq 1$ such that $x^n + y^n = z^n$
- Thus: most precise approximation yields $\{[\mathtt{x}, \mathtt{y}, \mathtt{z} \mapsto +, \mathtt{n} \mapsto -]\}$ iff equation unsolvable
- Fermat's Last Theorem: equation not solvable
- Final proof by Andrew Wiles and Richard Taylor in 1995

Derivation of Abstract Semantics

- **Problem:** most precise safe approximation not always definable

Example 13.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 11.3) on

$$\langle \texttt{if n > 2} \wedge \texttt{x\^{}n + y\^{}n = z\^{}n then n := 1 else n := -1 end}, \{[\mathrm{n}, \mathrm{x}, \mathrm{y}, \mathrm{z} \mapsto +]\}\rangle$$

- Result $\mathrm{n} \mapsto +$ possible iff there exist $n > 2$ and $x, y, z \geq 1$ such that $x^n + y^n = z^n$
- Thus: most precise approximation yields $\{[\mathrm{x}, \mathrm{y}, \mathrm{z} \mapsto +, \mathrm{n} \mapsto -]\}$ iff equation unsolvable
- Fermat's Last Theorem: equation not solvable
- Final proof by Andrew Wiles and Richard Taylor in 1995

- More general: solvability of Diophantic equations even undecidable
- Thus: resort to possibly imprecise safe approximations

Systematic Construction of Abstractions

Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete values
- 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

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

Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete values
- 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

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

Example 13.2

1. Parity abstraction (cf. Example 11.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ where

$$\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$$

Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete values
- 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

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

Example 13.2

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 $\beta = \text{sgn}$

Systematic Construction of Abstractions

Extraction Functions

- **Assumption:** abstraction determined by pointwise mapping of concrete values
- 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

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

Example 13.2

1. Parity abstraction (cf. Example 11.2): $\beta : \mathbb{Z} \to \{\mathrm{even}, \mathrm{odd}\}$ where

$$\beta(z) := \begin{cases} \mathrm{even} & \text{if } z \text{ even} \\ \mathrm{odd} & \text{if } z \text{ odd} \end{cases}$$

2. Sign abstraction (cf. Example 11.3): $\beta : \mathbb{Z} \to \{+, -, 0\}$ with $\beta = \mathrm{sgn}$
3. Interval abstraction (cf. Example 11.4): not definable by extraction function (as Int is not of the form 2^A)

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 12.1)

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) \sqsubseteq_M f^{\#}(m_1, \dots, m_n).$$

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 12.1)

$$\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^{\#}(m_1, \ldots, m_n).$$

Theorem 13.3

Let $L = 2^C$ and $M = 2^A$ with $\sqsubseteq_L = \sqsubseteq_M = \subseteq$, $\beta : C \to A$ be an extraction function, and $f : C^n \to C$. Then

$$f^{\#} : M^n \to M : (m_1, \ldots, m_n) \mapsto \{\beta(f(c_1, \ldots, c_n)) \mid \forall i \in \{1, \ldots, n\} : c_i \in \beta^{-1}(m_i)\}$$

is a safe approximation of f.

Systematic Construction of Abstractions

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 12.1)

$$\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \ldots, m_n).$$

Theorem 13.3

Let $L = 2^C$ and $M = 2^A$ with $\sqsubseteq_L = \sqsubseteq_M = \subseteq$, $\beta : C \to A$ be an extraction function, and $f : C^n \to C$. Then

$$f^\# : M^n \to M : (m_1, \ldots, m_n) \mapsto \{\beta(f(c_1, \ldots, c_n)) \mid \forall i \in \{1, \ldots, n\} : c_i \in \beta^{-1}(m_i)\}$$

is a safe approximation of f.

Proof.

on the board □

Safe Approximation of Arithmetic Operations

Example 13.4 (Sign abstraction)

For $C = \mathbb{Z}$, $A = \{+, -, 0\}$, $\beta = \text{sgn}$:

$+^{\#}$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{+\}$	$\{+, -, 0\}$	$\{+\}$
$\{-\}$	$\{+, -, 0\}$	$\{-\}$	$\{-\}$
$\{0\}$	$\{+\}$	$\{-\}$	$\{0\}$

$*^{\#}$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{+\}$	$\{-\}$	$\{0\}$
$\{-\}$	$\{-\}$	$\{+\}$	$\{0\}$
$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$

and
$$\begin{aligned}\{+, 0\} *^{\#} \{-\} &= \{+\} *^{\#} \{-\} \cup \{0\} *^{\#} \{-\} \\ &= \{-\} \cup \{0\} \\ &= \{-, 0\}\end{aligned}$$

etc.

Safe Approximation of Boolean Operations

Example 13.5 (Sign abstraction)

1. Relational operations: for $C = \mathbb{Z} \cup \mathbb{B}$, $A = \{+, -, 0\} \cup \mathbb{B}$, $\beta = \mathrm{sgn}$:

$=^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\mathrm{true}, \mathrm{false}\}$	$\{\mathrm{false}\}$	$\{\mathrm{false}\}$
$\{-\}$	$\{\mathrm{false}\}$	$\{\mathrm{true}, \mathrm{false}\}$	$\{\mathrm{false}\}$
$\{0\}$	$\{\mathrm{false}\}$	$\{\mathrm{false}\}$	$\{\mathrm{true}\}$

$>^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\mathrm{true}, \mathrm{false}\}$	$\{\mathrm{true}\}$	$\{\mathrm{true}\}$
$\{-\}$	$\{\mathrm{false}\}$	$\{\mathrm{true}, \mathrm{false}\}$	$\{\mathrm{false}\}$
$\{0\}$	$\{\mathrm{false}\}$	$\{\mathrm{true}\}$	$\{\mathrm{false}\}$

and $(\{+, 0\} =^\# \{0\}) = (\{+\} =^\# \{0\} \cup \{0\} =^\# \{0\}) = \{\mathrm{false}\} \cup \{\mathrm{true}\} = \{\mathrm{true}, \mathrm{false}\}$ etc.

Safe Approximation of Boolean Operations

Example 13.5 (Sign abstraction)

1. Relational operations: for $C = \mathbb{Z} \cup \mathbb{B}$, $A = \{+, -, 0\} \cup \mathbb{B}$, $\beta = \text{sgn}$:

$=^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$	$\{\text{false}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{false}\}$	$\{\text{true}\}$

$>^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{true}\}$	$\{\text{true}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{true}\}$	$\{\text{false}\}$

   and $(\{+, 0\} =^\# \{0\}) = (\{+\} =^\# \{0\} \cup \{0\} =^\# \{0\}) = \{\text{false}\} \cup \{\text{true}\} = \{\text{true}, \text{false}\}$ etc.

2. Boolean connectives: for $C = A = \mathbb{B}$:

$$\neg^\# = \neg \qquad \wedge^\# = \wedge \qquad \vee^\# = \vee$$

   and
$$\begin{aligned}\{\text{true}, \text{false}\} \wedge^\# \{\text{true}\} &= \{\text{true}\} \wedge^\# \{\text{true}\} \cup \{\text{false}\} \wedge^\# \{\text{true}\} \\ &= \{\text{true}\} \cup \{\text{false}\} \\ &= \{\text{true}, \text{false}\}\end{aligned}$$

   etc.

Outline of Lecture 13

Recap: Safe Approximation of Functions and Relations

Systematic Construction of Abstractions

Abstract Semantics of WHILE

Correctness of Abstract Semantics

Abstract Program States

Now: take values of variables into account

Definition 13.6 (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.

Abstract Program States

Now: take values of variables into account

Definition 13.6 (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 $Abs := 2^{Var \to A}$.

Abstract Program States

Now: take values of variables into account

Definition 13.6 (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 13.7 (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)

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

Abstract Evaluation of Expressions

Definition 13.7 (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 operation)
$$\begin{aligned} val^\#_\rho(t) &:= \{t\} \\ val^\#_\rho(g(a_1, \ldots, a_n)) &:= g^\#(val^\#_\rho(a_1), \ldots, val^\#_\rho(a_n)) \\ val^\#_\rho(h(b_1, \ldots, b_n)) &:= h^\#(val^\#_\rho(b_1), \ldots, val^\#_\rho(b_n)) \end{aligned}$$

Abstract Evaluation of Expressions

Definition 13.7 (Abstract evaluation functions)

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

1. $\mathit{val}^{\#}_{\rho} : \mathit{AExp} \to 2^{A}$ is determined by (f arithmetic operation)
$$\begin{aligned} \mathit{val}^{\#}_{\rho}(z) &:= \{\beta(z)\} \\ \mathit{val}^{\#}_{\rho}(x) &:= \{\rho(x)\} \\ \mathit{val}^{\#}_{\rho}(f(a_1,\ldots,a_n)) &:= f^{\#}(\mathit{val}^{\#}_{\rho}(a_1),\ldots,\mathit{val}^{\#}_{\rho}(a_n)) \end{aligned}$$
2. $\mathit{val}^{\#}_{\rho} : \mathit{BExp} \to 2^{\mathbb{B}}$ is determined by (g/h relational/Boolean operation)
$$\begin{aligned} \mathit{val}^{\#}_{\rho}(t) &:= \{t\} \\ \mathit{val}^{\#}_{\rho}(g(a_1,\ldots,a_n)) &:= g^{\#}(\mathit{val}^{\#}_{\rho}(a_1),\ldots,\mathit{val}^{\#}_{\rho}(a_n)) \\ \mathit{val}^{\#}_{\rho}(h(b_1,\ldots,b_n)) &:= h^{\#}(\mathit{val}^{\#}_{\rho}(b_1),\ldots,\mathit{val}^{\#}_{\rho}(b_n)) \end{aligned}$$

Example 13.8 (Sign abstraction)

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

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

Abstract Semantics of WHILE I

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

Definition 13.9 (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 13.9 (Abstract execution relation for statements; continued)

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

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

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

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

Abstract Semantics of WHILE III

Definition 13.10 (Abstract transition function)

The abstract transition function is defined by the family of mappings

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

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

Abstract Semantics of WHILE III

Definition 13.10 (Abstract transition function)

The abstract transition function is defined by the family of mappings

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

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

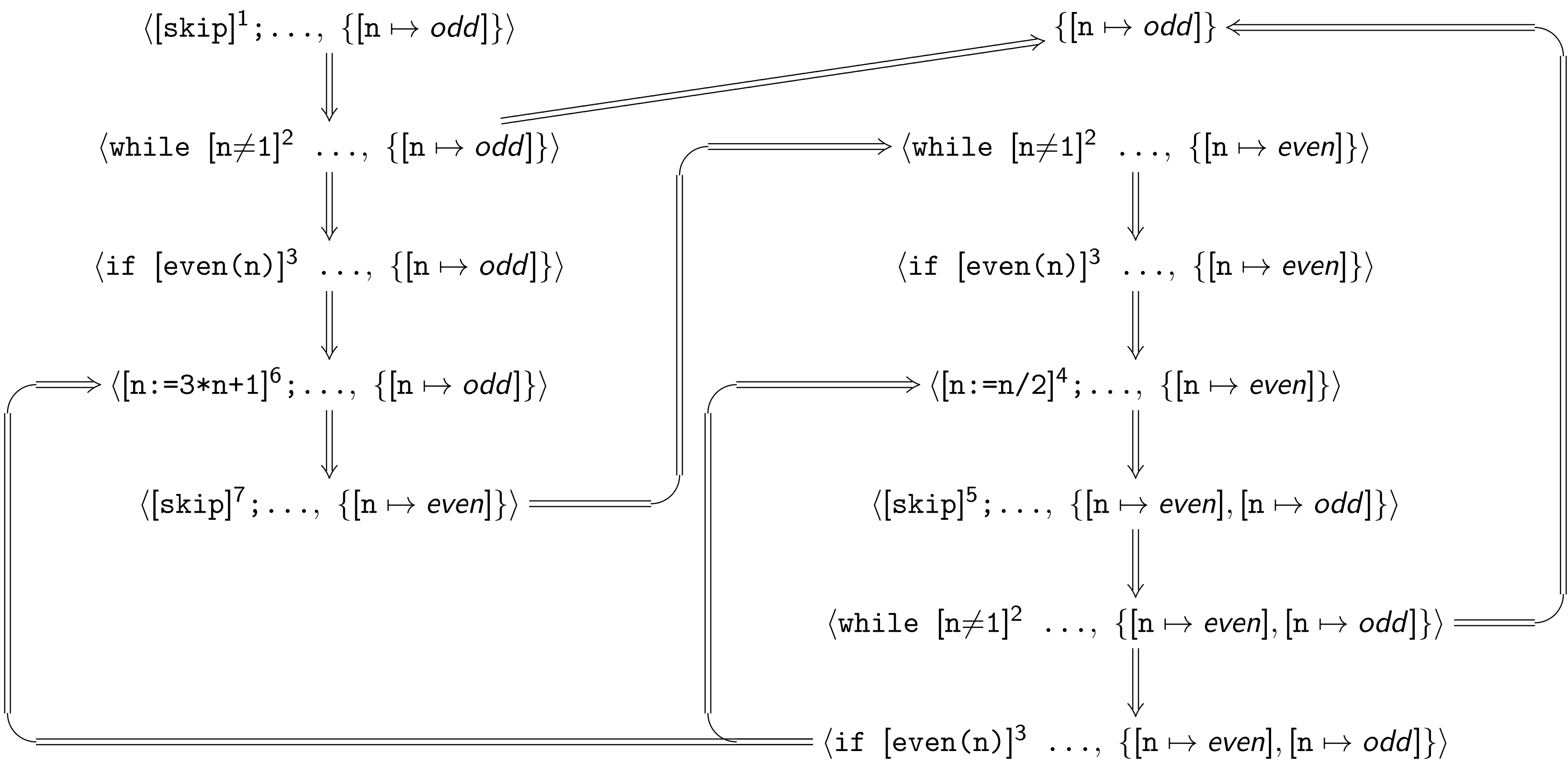
Example 13.11 (Hailstone Sequences; cf. Example 12.7)

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

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Abstrakte Interpretation von HAILSTONE



Outline of Lecture 13

Correctness of Abstract Semantics

Theorem 13.12 (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* $abs \in Abs$, $\alpha(\mathrm{next}_{c,c'}(\gamma(abs))) \sqsubseteq \mathrm{next}^{\#}_{c,c'}(abs)$.

Correctness of Abstract Semantics

Theorem 13.12 (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 $abs \in Abs$, $\alpha(\mathrm{next}_{c,c'}(\gamma(abs))) \sqsubseteq \mathrm{next}^{\#}_{c,c'}(abs)$.

The soundness proof employs the following auxiliary lemma.

Lemma 13.13 (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 13.13).

omitted □

Correctness of Abstract Semantics

Theorem 13.12 (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 13.13 (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 13.13).

omitted □

Proof (Theorem 13.12).

on the board □