Static Program Analysis

Lecture 12: Abstract Interpretation III (Abstract Semantics of WHILE)

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 12

Recap: Safe Approximation of Functions and Relations

Safe Approximation of Functions

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 or $f^\#$ is *not* required (but usually given; see Lemma 11.3)

Safe Approximation of Functions IV

Lemma

If $f : L^n \to L$ and $f^\# : M^n \to M$ are monotonic, then $f^\#$ is a safe approximation of f iff, for all $l_1, \ldots, l_n \in L$,

$$\alpha(f(l_1, \ldots, l_n)) \sqsubseteq_M f^\#(\alpha(l_1), \ldots, \alpha(l_n)).$$

Proof.

on the board □

Encoding Execution Relations by Transition Functions

- **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 10.6)
  - execution relation $\to \subseteq (\mathit{Cmd} \times \Sigma) \times ((\mathit{Cmd} \cup \{\downarrow\}) \times \Sigma)$ (Definition 10.9)
- 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 $S \subseteq \Sigma$,

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

Recap: Safe Approximation of Functions and Relations

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
- often $M = Var \to \ldots$ (more efficient) or $M = 2^{Var \to \ldots}$ (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

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

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

Outline of Lecture 12

Derivation of Abstract Semantics

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

Example 12.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 10.3) on

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

Derivation of Abstract Semantics

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

Example 12.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 10.3) 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 12.1 (Fermat's Last Theorem)

Sign abstraction (cf. Example 10.3) on

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

- Result $\texttt{n} \mapsto +$ 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 $\{[\texttt{x}, \texttt{y}, \texttt{z} \mapsto +, \texttt{n} \mapsto -]\}$ 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

Extraction Functions

Assumption: abstraction determined by pointwise mapping of concrete values

Definition 12.2 (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

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

Extraction Functions

Assumption: abstraction determined by pointwise mapping of concrete values

Definition 12.2 (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 12.3

1. Parity abstraction (cf. Ex. 10.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ where $\beta(z) := \begin{cases} \text{even} & \text{if } z \text{ even} \\ \text{odd} & \text{if } z \text{ odd} \end{cases}$

Extraction Functions

Assumption: abstraction determined by pointwise mapping of concrete values

Definition 12.2 (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 12.3

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 $\beta = \text{sgn}$

Extraction Functions

Assumption: abstraction determined by pointwise mapping of concrete values

Definition 12.2 (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 12.3

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)

Safe Approximation by Extraction Functions

Reminder: most precise safe approximation condition (Definition 11.1)

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

Safe Approximation by Extraction Functions

Reminder: most precise safe approximation condition (Definition 11.1)

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

Theorem 12.4

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

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

is the most precise safe approximation of f.

Safe Approximation by Extraction Functions

Reminder: most precise safe approximation condition (Definition 11.1)

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

Theorem 12.4

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

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

is the most precise safe approximation of f.

Proof.

on the board □

Safe Approximation of Arithmetic Operations

Example 12.5 (Sign abstraction)

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

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

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

and $\{+,0\} *^\# \{-\} = \{+\} *^\# \{-\} \cup \{0\} *^\# \{-\}$
$= \{-\} \cup \{0\}$
$= \{-,0\}$

etc.

Safe Approximation of Boolean Operations

Example 12.6 (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.

Safe Approximation of Boolean Operations

Example 12.6 (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 $\{\text{true}, \text{false}\} \wedge^\# \{\text{true}\} = \{\text{true}\} \wedge^\# \{\text{true}\} \cup \{\text{false}\} \wedge^\# \{\text{true}\}$
   $= \{\text{true}\} \cup \{\text{false}\}$
   $= \{\text{true}, \text{false}\}$

   etc.

Outline of Lecture 12

Abstract Semantics of WHILE using Extraction Functions

Abstract Program States

Now: take values of variables into account

Definition 12.7 (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 Semantics of WHILE using Extraction Functions

Abstract Program States

Now: take values of variables into account

Definition 12.7 (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 Semantics of WHILE using Extraction Functions

Abstract Program States

Now: take values of variables into account

Definition 12.7 (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 12.8 (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)

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

Abstract Evaluation of Expressions

Definition 12.8 (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)

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

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

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

Abstract Evaluation of Expressions

Definition 12.8 (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)

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

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

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

Example 12.9 (Sign abstraction)

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

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

Abstract Semantics of WHILE I

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

Definition 12.10 (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 12.10 (Abstract execution relation for statements; continued)

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

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

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

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

Abstract Semantics of WHILE III

Definition 12.11 (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 $\text{next}^{\#}_{c,c'}(\mathit{abs}) := \bigcup \{ \mathit{abs}' \in \mathit{Abs} \mid \langle c, \mathit{abs} \rangle \Rightarrow \langle c', \mathit{abs}' \rangle \}$

Abstract Semantics of WHILE III

Definition 12.11 (Abstract transition function)

The abstract transition function is defined by the family of mappings

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

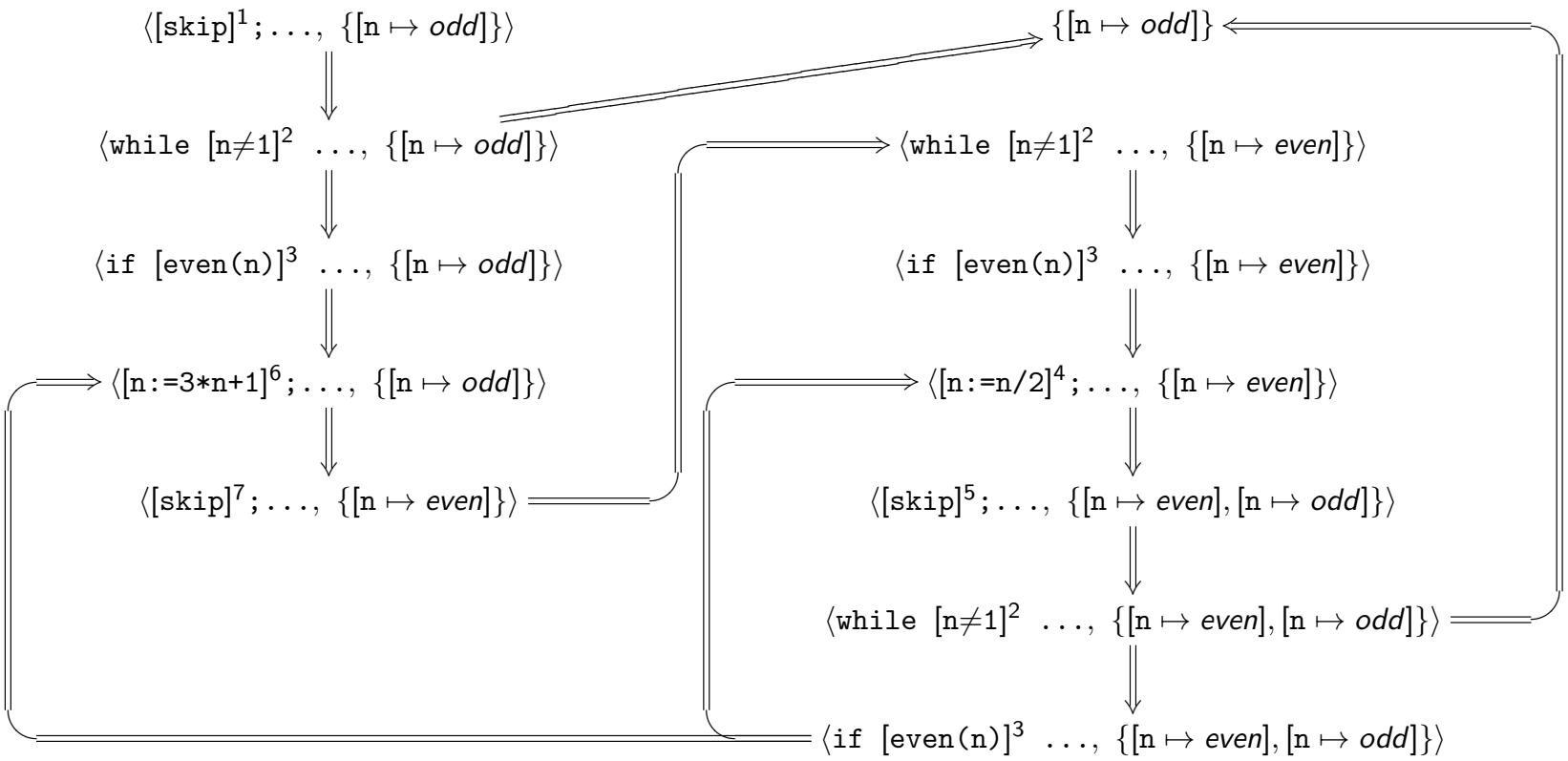
given by $\text{next}^{\#}_{c,c'}(\mathit{abs}) := \bigcup\{\mathit{abs}' \in \mathit{Abs} \mid \langle c, \mathit{abs}\rangle \Rightarrow \langle c', \mathit{abs}'\rangle\}$

Example 12.12 (Hailstone Sequences; cf. Example 11.7)

```
[skip]^1;
while [¬(n = 1)]^2 do
  if [even(n)]^3 then
    [n := n / 2]^4;[skip]^5
  else
    [n := 3 * n + 1]^6;[skip]^7
  end
end
```

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

Abstrakte Interpretation von HAILSTONE



Outline of Lecture 12

Correctness of Abstract Semantics

Theorem 12.13 (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* $abs \in Abs$, $\alpha(\text{next}_{c,c'}(\gamma(abs))) \subseteq \text{next}^{\#}_{c,c'}(abs)$.

Correctness of Abstract Semantics

Theorem 12.13 (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 12.14 (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 □

Correctness of Abstract Semantics

Theorem 12.13 (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* $abs \in Abs$, $\alpha(\text{next}_{c,c'}(\gamma(abs))) \sqsubseteq \text{next}^{\#}_{c,c'}(abs)$.

The soundness proof employs the following auxiliary lemma.

Lemma 12.14 (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 □

Proof (Theorem 12.13).

on the board □