



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/>

Recap: Safe Approximation of Functions and Relations

Safe Approximation of Functions

Definition (Safe approximation)

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

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

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

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

- **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)

Recap: Safe Approximation of Functions and Relations

Safe Approximation of Functions IV

Lemma

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

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

Proof.

on the board



Recap: Safe Approximation of Functions and Relations

Encoding Execution Relations by Transition Functions

- **Reminder: concrete semantics** of WHILE
 - **statements** $\text{skip} \mid x := a \mid c_1; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end} \mid \text{while } b \text{ do } c \text{ end} \in \text{Cmd}$
 - **states** $\Sigma := \{\sigma \mid \sigma : \text{Var} \rightarrow \mathbb{Z}\}$ (Definition 10.6)
 - **execution relation** $\rightarrow \subseteq (\text{Cmd} \times \Sigma) \times ((\text{Cmd} \cup \{\downarrow\}) \times \Sigma)$ (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

$$\text{next}_{c,c'} : 2^\Sigma \rightarrow 2^\Sigma$$

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

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

Recap: Safe Approximation of Functions and Relations

Safe Approximation of Execution Relations

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

- here: $L := 2^\Sigma$, M not fixed
- often $M = \text{Var} \rightarrow \dots$ (more efficient) or $M = 2^{\text{Var} \rightarrow \dots}$ (more precise)
- write *Abs* in place of M
- thus $\alpha : 2^\Sigma \rightarrow \text{Abs}$ and $\gamma : \text{Abs} \rightarrow 2^\Sigma$

Definition (Abstract semantics of WHILE)

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

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

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

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

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

Systematic Construction of Abstractions

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

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

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

Definition 12.2 (Extraction function)

- If $L = 2^C$ and $M = 2^A$ with $\sqsubseteq_L = \sqsubseteq_M = \subseteq$, then $\beta : C \rightarrow A$ is called an **extraction function**
- β determines **Galois connection** (α, γ) with

$$\alpha : L \rightarrow M : l \mapsto \beta(l) \quad (= \{\beta(c) \mid c \in l\})$$

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

Example 12.3

1. Parity abstraction (cf. Ex. 10.2): $\beta : \mathbb{Z} \rightarrow \{\text{even}, \text{odd}\}$ 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): $\beta : \mathbb{Z} \rightarrow \{+, -, 0\}$ 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)

Systematic Construction of Abstractions

Safe Approximation by Extraction Functions

Reminder: **most precise safe approximation** condition (Definition 11.1)

$$\alpha(f(\gamma(m_1), \dots, \gamma(m_n))) = f^\#(m_1, \dots, m_n).$$

Theorem 12.4

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

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

is the **most precise safe approximation** of f .

Proof.

on the board



Systematic Construction of Abstractions

Safe Approximation of Arithmetic Operations

Example 12.5 (Sign abstraction)

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

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

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

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

etc.

Systematic Construction of Abstractions

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 $C = A = \mathbb{B}$:

$$\neg\# = \neg \quad \wedge\# = \wedge \quad \vee\# = \vee$$

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

etc.

Abstract Semantics of WHILE using Extraction Functions

Abstract Program States

Now: take values of variables into account

Definition 12.7 (Abstract program state)

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

- An **abstract (program) state** is an element of the set

$$\{\rho \mid \rho : \text{Var} \rightarrow A\},$$

called the **abstract state space**.

- The **abstract domain** is denoted by $Abs := 2^{\text{Var} \rightarrow A}$.
- The **abstraction function** $\alpha : 2^\Sigma \rightarrow Abs$ is given by

$$\alpha(S) := \{\beta \circ \sigma \mid \sigma \in S\}$$

for every $S \subseteq \Sigma$.

Abstract Semantics of WHILE using Extraction Functions

Abstract Evaluation of Expressions

Definition 12.8 (Abstract evaluation functions)

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

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

$$val_{\rho}^{\#}(z) := \{\beta(z)\}$$

$$val_{\rho}^{\#}(x) := \{\rho(x)\}$$

$$val_{\rho}^{\#}(f(a_1, \dots, a_n)) := f^{\#}(val_{\rho}^{\#}(a_1), \dots, val_{\rho}^{\#}(a_n))$$

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

$$val_{\rho}^{\#}(t) := \{t\}$$

$$val_{\rho}^{\#}(g(a_1, \dots, a_n)) := g^{\#}(val_{\rho}^{\#}(a_1), \dots, val_{\rho}^{\#}(a_n))$$

$$val_{\rho}^{\#}(h(b_1, \dots, b_n)) := h^{\#}(val_{\rho}^{\#}(b_1), \dots, val_{\rho}^{\#}(b_n))$$

Example 12.9 (Sign abstraction)

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

1. $val_{\rho}^{\#}(2 * x + y) = \{+, -, 0\}$

2. $val_{\rho}^{\#}(\neg(x + 1 > y)) = \{\text{false}\}$

Abstract Semantics of WHILE using Extraction Functions

Abstract Semantics of WHILE I

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

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:

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

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

$$\frac{\langle c_1, abs \rangle \Rightarrow \langle c'_1, abs' \rangle \quad c'_1 \neq \downarrow}{\text{(seq1)} \quad \langle c_1 ; c_2, abs \rangle \Rightarrow \langle c'_1 ; c_2, abs' \rangle}$$

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

Abstract Semantics of WHILE using Extraction Functions

Abstract Semantics of WHILE II

Definition 12.10 (Abstract execution relation for statements; continued)

$$\exists \rho \in abs : true \in val_{\rho}^{\#}(b)$$

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

$$\exists \rho \in abs : false \in val_{\rho}^{\#}(b)$$

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

$$\exists \rho \in abs : true \in val_{\rho}^{\#}(b)$$

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

$$\exists \rho \in abs : false \in val_{\rho}^{\#}(b)$$

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

Abstract Semantics of WHILE using Extraction Functions

Abstract Semantics of WHILE III

Definition 12.11 (Abstract transition function)

The **abstract transition function** is defined by the family of mappings

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

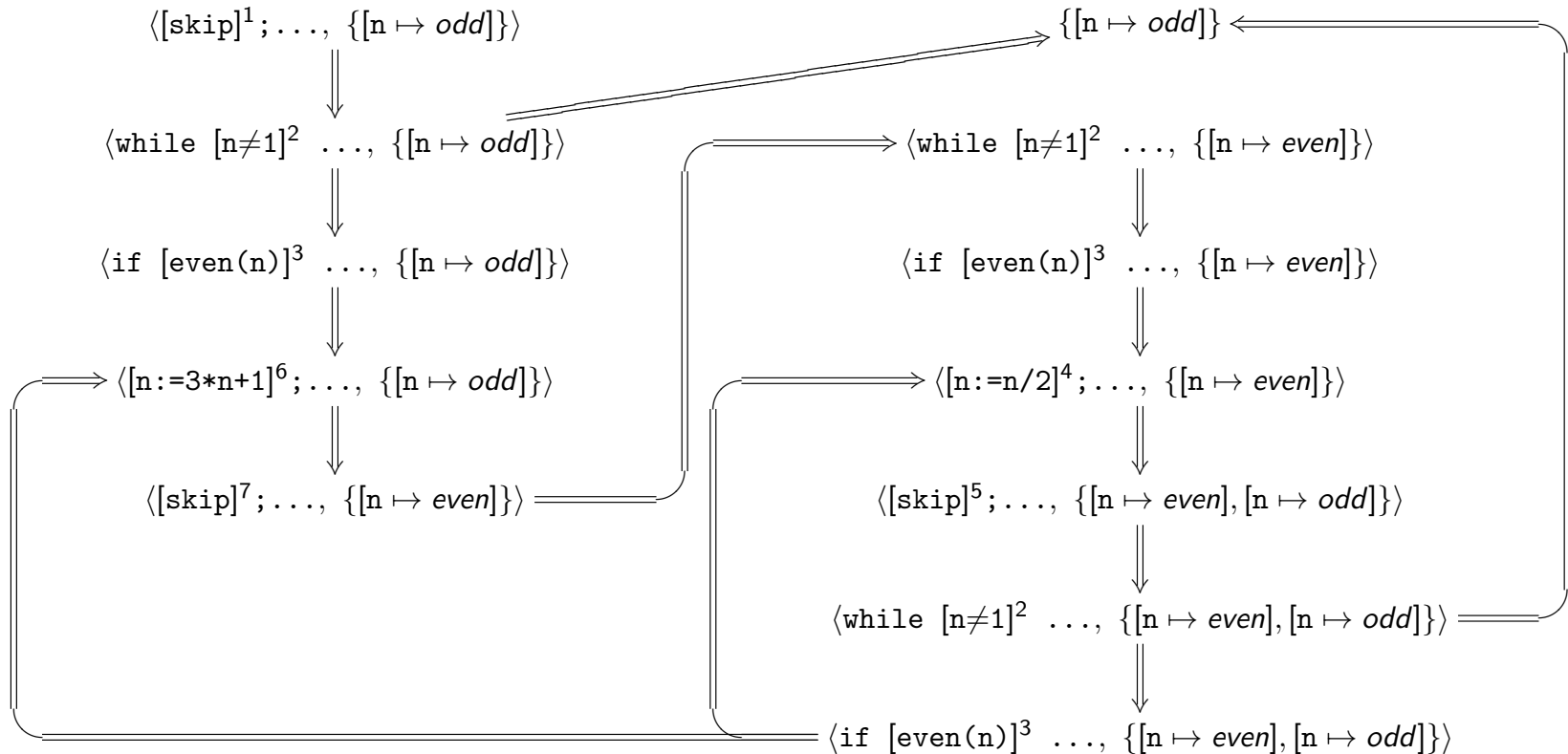
given by $\text{next}_{c,c'}^{\#}(abs) := \bigcup \{abs' \in \text{Abs} \mid \langle c, abs \rangle \Rightarrow \langle c', abs' \rangle\}$

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



Correctness of Abstract Semantics

Correctness of Abstract Semantics

Theorem 12.13 (Soundness of abstract semantics)

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

The soundness proof employs the following auxiliary lemma.

Lemma 12.14 (Soundness of abstract evaluation)

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

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

Proof (Lemma 12.14).

omitted

Proof (Theorem 12.13).

on the board