Static Program Analysis

Lecture 10: Abstract Interpretation I (Galois Connections)

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 10

Abstract Interpretation I

- **Summary:** a theory of sound approximation of the semantics of programs
- **Basic idea:** execution of program on abstract values
- **Examples:**
  - parity (even/odd) or value intervals rather than concrete numbers
  - types rather than concrete values (similar to type-level JVM bytecode interpreter)

Abstract Interpretation I

- **Summary:** a theory of sound approximation of the semantics of programs
- **Basic idea:** execution of program on abstract values
- **Examples:**
  - parity (even/odd) or value intervals rather than concrete numbers
  - types rather than concrete values (similar to type-level JVM bytecode interpreter)
- **Procedure:** run program on (finite) set of abstract values that cover all concrete inputs using abstract operations (for basic operations, statements, ...) that cover all concrete outputs
  $\Longrightarrow$ soundness of approach
- **Preciseness** of information again characterized by partial order

Abstract Interpretation II

- **Advantages:**
  - Abstract interpretation covers conditional branches (`if`/`while`) without further extension
  - Granularity of abstract domain influences precision and complexity of analysis (mutual trade-off)
    - enables refinement approaches (predicate abstraction)
  - Numerous variants for different kinds of programs (functional, concurrent, ...)
  - Soundness is guaranteed if abstract operations are determined according to theory

Abstract Interpretation II

- **Advantages:**
  - Abstract interpretation covers conditional branches (`if`/`while`) without further extension
  - Granularity of abstract domain influences precision and complexity of analysis (mutual trade-off)
    - enables refinement approaches (predicate abstraction)
  - Numerous variants for different kinds of programs (functional, concurrent, ...)
  - Soundness is guaranteed if abstract operations are determined according to theory
- **Disadvantages:**
  - Complexity generally higher than with dataflow analysis
  - Automatic derivation of abstract operations can be difficult

Overview

1. Theoretical foundations (Galois connections)
2. (Concrete &) Abstract semantics of WHILE programs
3. Automatic derivation of abstract semantics
4. Application: verification of 16-bit multiplication
5. Predicate abstraction
6. CEGAR (CounterExample-Guided Abstraction Refinement)

Outline of Lecture 10

Galois Connections I

Definition 10.1 (Galois connection)

Let $(L, \sqsubseteq_L)$ and $(M, \sqsubseteq_M)$ be complete lattices. A pair (α, γ) of monotonic functions

$$\alpha : L \to M \quad \text{and} \quad \gamma : M \to L$$

is called a Galois connection if

$$\forall l \in L : l \sqsubseteq_L \gamma(\alpha(l)) \quad \text{and} \quad \forall m \in M : \alpha(\gamma(m)) \sqsubseteq_M m$$

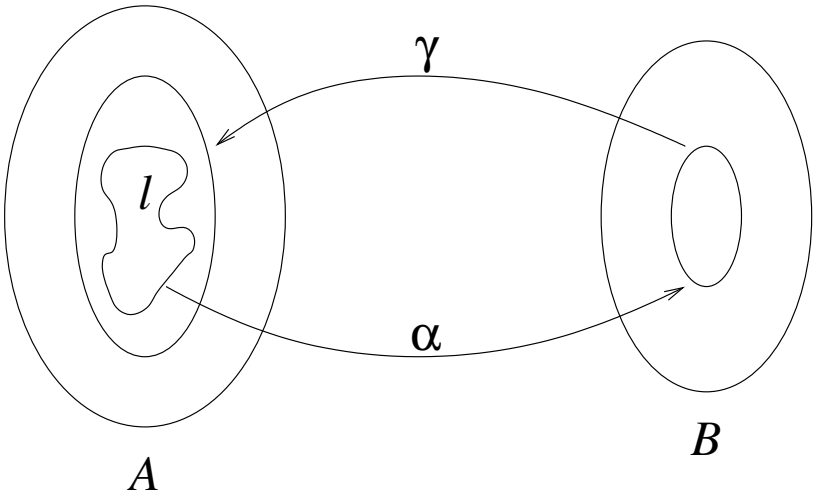
Evariste Galois
(1811–1832)

Interpretation:

- $L = \{\text{sets of concrete values}\}$, $M = \{\text{sets of abstract values}\}$
- $\alpha =$ abstraction function, $\gamma =$ concretisation function
- $l \sqsubseteq_L \gamma(\alpha(l))$: α yields over-approximation
- $\alpha(\gamma(m)) \sqsubseteq_M m$: no loss of precision by abstraction after concretisation
- Usually: $l \neq \gamma(\alpha(l))$, $\alpha(\gamma(m)) = m$ (“Galois insertion”)

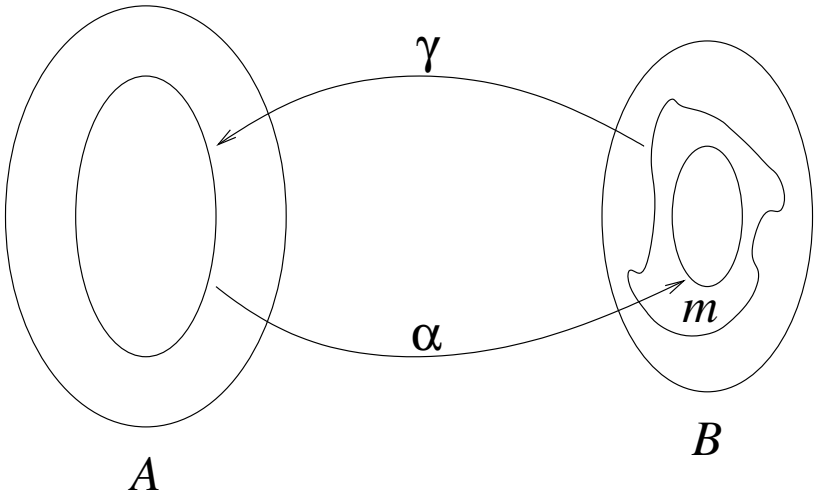
Galois Connections II

For $A = \{\text{concrete values}\}$, $B = \{\text{abstract values}\}$, $L = 2^A$, $M = 2^B$:



$\forall l \in L : l \sqsubseteq_L \gamma(\alpha(l))$

(α yields over-approximation)



$\forall m \in M : \alpha(\gamma(m)) \sqsubseteq_M m$

(no loss of precision by abstraction after concretisation)

Galois Connections III

Example 10.2 (Parity abstraction)

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

$$\gamma : 2^{\{\text{even},\text{odd}\}} \to 2^{\mathbb{Z}}$$

$$\gamma(P) := \bigcup_{p \in P} \mathbb{Z}_p$$

where

$$\mathbb{Z}_{\text{even}} := \{\ldots, -2, 0, 2, \ldots\}$$
$$\mathbb{Z}_{\text{odd}} := \{\ldots, -3, -1, 1, 3, \ldots\}$$

Abstract domain

$$M := (2^{\{\text{even},\text{odd}\}}, \subseteq)$$

$$\alpha : 2^{\mathbb{Z}} \to 2^{\{\text{even},\text{odd}\}}$$

$$\alpha(Z) := \begin{cases} \emptyset & \text{if } Z = \emptyset \\ \{\text{even}\} & \text{if } Z \subseteq \mathbb{Z}_{\text{even}} \\ \{\text{odd}\} & \text{if } Z \subseteq \mathbb{Z}_{\text{odd}} \\ \{\text{even}, \text{odd}\} & \text{otherwise} \end{cases}$$

yields a Galois connection.

Galois Connections III

Example 10.2 (Parity abstraction)

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

$$\gamma : 2^{\{\text{even},\text{odd}\}} \to 2^{\mathbb{Z}}$$

$$\gamma(P) := \bigcup_{p \in P} \mathbb{Z}_p$$

where

$$\mathbb{Z}_{\text{even}} := \{\ldots, -2, 0, 2, \ldots\}$$
$$\mathbb{Z}_{\text{odd}} := \{\ldots, -3, -1, 1, 3, \ldots\}$$

Abstract domain

$$M := (2^{\{\text{even},\text{odd}\}}, \subseteq)$$

$$\alpha : 2^{\mathbb{Z}} \to 2^{\{\text{even},\text{odd}\}}$$

$$\alpha(Z) := \begin{cases} \emptyset & \text{if } Z = \emptyset \\ \{\text{even}\} & \text{if } Z \subseteq \mathbb{Z}_{\text{even}} \\ \{\text{odd}\} & \text{if } Z \subseteq \mathbb{Z}_{\text{odd}} \\ \{\text{even}, \text{odd}\} & \text{otherwise} \end{cases}$$

yields a Galois connection. For example,

- $\gamma(\alpha(\{1, 3, 7\})) = \gamma(\{\text{odd}\}) = \{\ldots, -3, -1, 1, 3, \ldots\} \supseteq \{1, 3, 7\}$
- $\alpha(\gamma(\{\text{even}\})) = \alpha(\{\ldots, -2, 0, 2, \ldots\}) = \{\text{even}\}$

Galois Connections IV

Example 10.3 (Sign abstraction)

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

$$\gamma : 2^{\{+,-,0\}} \to 2^{\mathbb{Z}}$$
$$\gamma(S) := \bigcup_{s \in S} \mathbb{Z}_s$$

where

$$\mathbb{Z}_+ := \{1, 2, 3, \ldots\}$$
$$\mathbb{Z}_- := \{-1, -2, -3, \ldots\}$$
$$\mathbb{Z}_0 := \{0\}$$

Abstract domain

$$M := (2^{\{+,-,0\}}, \subseteq)$$

$$\alpha : 2^{\mathbb{Z}} \to 2^{\{+,-,0\}}$$
$$\alpha(Z) := \{\mathrm{sgn}(z) \mid z \in Z\}$$

where

$$\mathrm{sgn}(z) := \begin{cases} + & \text{if } z > 0 \\ - & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases}$$

yields a Galois connection.

Galois Connections IV

Example 10.3 (Sign abstraction)

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

$$\gamma : 2^{\{+,-,0\}} \to 2^{\mathbb{Z}}$$
$$\gamma(S) := \bigcup_{s \in S} \mathbb{Z}_s$$

where

$$\mathbb{Z}_+ := \{1, 2, 3, \ldots\}$$
$$\mathbb{Z}_- := \{-1, -2, -3, \ldots\}$$
$$\mathbb{Z}_0 := \{0\}$$

Abstract domain

$$M := (2^{\{+,-,0\}}, \subseteq)$$

$$\alpha : 2^{\mathbb{Z}} \to 2^{\{+,-,0\}}$$
$$\alpha(Z) := \{\mathrm{sgn}(z) \mid z \in Z\}$$

where

$$\mathrm{sgn}(z) := \begin{cases} + & \text{if } z > 0 \\ - & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases}$$

yields a Galois connection. For example,

- $\gamma(\alpha(\{0, 1, 3\})) = \gamma(\{+, 0\}) = \{0, 1, 2, 3, \ldots\} \supseteq \{0, 1, 3\}$
- $\alpha(\gamma(\{+, -\})) = \alpha(\mathbb{Z} \setminus \{0\}) = \{+, -\}$

Galois Connections V

Example 10.4 (Interval abstraction (cf. Slide 6.21))

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

Abstract domain

$$M := (\mathit{Int}, \subseteq)$$

where $\mathit{Int} := (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}$

$$\gamma : \mathit{Int} \to 2^{\mathbb{Z}}$$

$$\gamma(J) := \begin{cases} \emptyset & \text{if } J = \emptyset \\ \{z \in \mathbb{Z} \mid z_1 \leq z \leq z_2\} & \\ & \text{if } J = [z_1, z_2] \end{cases}$$

$$\alpha : 2^{\mathbb{Z}} \to \mathit{Int}$$

$$\alpha(Z) := \begin{cases} \emptyset & \text{if } Z = \emptyset \\ [\bigsqcap Z, \bigsqcup Z] & \text{otherwise} \end{cases}$$

yields a Galois connection.

Galois Connections V

Example 10.4 (Interval abstraction (cf. Slide 6.21))

Concrete domain

$$L := (2^{\mathbb{Z}}, \subseteq)$$

Abstract domain

$$M := (\mathit{Int}, \subseteq)$$

where $\mathit{Int} := (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}$

$$\gamma : \mathit{Int} \to 2^{\mathbb{Z}}$$

$$\gamma(J) := \begin{cases} \emptyset & \text{if } J = \emptyset \\ \{z \in \mathbb{Z} \mid z_1 \leq z \leq z_2\} & \\ & \text{if } J = [z_1, z_2] \end{cases}$$

$$\alpha : 2^{\mathbb{Z}} \to \mathit{Int}$$

$$\alpha(Z) := \begin{cases} \emptyset & \text{if } Z = \emptyset \\ [\bigsqcap Z, \bigsqcup Z] & \text{otherwise} \end{cases}$$

yields a Galois connection. For example,

- $\gamma(\alpha(\{1, 3, 5, \ldots\})) = \gamma([1, +\infty]) = \{1, 2, 3, 4, 5, \ldots\} \supseteq \{1, 3, 5, \ldots\}$
- $\alpha(\gamma([-1, 1])) = \alpha(\{-1, 0, 1\}) = [-1, 1]$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$
2. γ *is uniquely determined by* α *as follows:* $\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \sqsubseteq_M m\}$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$
2. γ *is uniquely determined by* α *as follows:* $\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \sqsubseteq_M m\}$
3. α *is uniquely determined by* γ *as follows:* $\alpha(l) = \bigsqcap\{m \in M \mid l \sqsubseteq_L \gamma(m)\}$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$
2. γ *is uniquely determined by* α *as follows:* $\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \sqsubseteq_M m\}$
3. α *is uniquely determined by* γ *as follows:* $\alpha(l) = \bigsqcap\{m \in M \mid l \sqsubseteq_L \gamma(m)\}$
4. α *is completely distributive: for every* $L' \subseteq L$, $\alpha(\bigsqcup L') = \bigsqcup\{\alpha(l) \mid l \in L'\}$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$
2. γ *is uniquely determined by* α *as follows:* $\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \sqsubseteq_M m\}$
3. α *is uniquely determined by* γ *as follows:* $\alpha(l) = \bigsqcap\{m \in M \mid l \sqsubseteq_L \gamma(m)\}$
4. α *is completely distributive: for every* $L' \subseteq L$, $\alpha(\bigsqcup L') = \bigsqcup\{\alpha(l) \mid l \in L'\}$
5. γ *is completely multiplicative: for every* $M' \subseteq M$, $\gamma(\bigsqcap M') = \bigsqcap\{\gamma(m) \mid m \in M'\}$

Properties of Galois Connections

Lemma 10.5

Let (α, γ) be a Galois connection with $\alpha : L \to M$ and $\gamma : M \to L$, and let $l \in L$, $m \in M$, $L' \subseteq L$, $M' \subseteq M$.

1. $\alpha(l) \sqsubseteq_M m \iff l \sqsubseteq_L \gamma(m)$
2. γ *is uniquely determined by* α *as follows:* $\gamma(m) = \bigsqcup\{l \in L \mid \alpha(l) \sqsubseteq_M m\}$
3. α *is uniquely determined by* γ *as follows:* $\alpha(l) = \bigsqcap\{m \in M \mid l \sqsubseteq_L \gamma(m)\}$
4. α *is completely distributive: for every* $L' \subseteq L$, $\alpha(\bigsqcup L') = \bigsqcup\{\alpha(l) \mid l \in L'\}$
5. γ *is completely multiplicative: for every* $M' \subseteq M$, $\gamma(\bigsqcap M') = \bigsqcap\{\gamma(m) \mid m \in M'\}$

Proof.

on the board □

Outline of Lecture 10

Excursus: Concrete Semantics of WHILE Programs

Reminder: Syntax of WHILE

The syntax of WHILE Programs is defined by the following context-free grammar (cf. Definition 1.3):

$a ::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in \mathit{AExp}$

$b ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in \mathit{BExp}$

$c ::= \texttt{skip} \mid x := a \mid c_1 \,;\, c_2 \mid \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end} \mid \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end} \in \mathit{Cmd}$

Excursus: Concrete Semantics of WHILE Programs

Program States

- Meaning of expression = value (in the usual sense)
- Depends on the values of the variables in the expression

Program States

- Meaning of expression = value (in the usual sense)
- Depends on the values of the variables in the expression

Definition 10.6 (Program state)

A (program) state is an element of the set

$$\Sigma := \{\sigma \mid \sigma : \mathit{Var} \to \mathbb{Z}\},$$

called the state space.

Thus $\sigma(x)$ denotes the value of $x \in \mathit{Var}$ in state $\sigma \in \Sigma$.

Evaluation of Expressions

Definition 10.7 (Evaluation function)

Let $\sigma \in \Sigma$ be a state.

1. $val_\sigma : AExp \to \mathbb{Z} : a \to val_\sigma(a)$ yields the value of a in state σ
2. $val_\sigma : BExp \to \mathbb{B} : b \to val_\sigma(b)$ yields the value of b in state σ

Evaluation of Expressions

Definition 10.7 (Evaluation function)

Let $\sigma \in \Sigma$ be a state.

1. $val_\sigma : AExp \to \mathbb{Z} : a \to val_\sigma(a)$ yields the value of a in state σ
2. $val_\sigma : BExp \to \mathbb{B} : b \to val_\sigma(b)$ yields the value of b in state σ

Example 10.8

Let $\sigma(\mathtt{x}) = 1$ and $\sigma(\mathtt{y}) = 2$.

1. $val_\sigma(2 \ * \ \mathtt{x} \ + \ \mathtt{y}) = 4$
2. $val_\sigma(\neg(\mathtt{x} \ + \ 1 \ > \ \mathtt{y})) = \text{true}$

Derivation Rules

- Definition employs derivation rules of the form

$$\text{Name}\ \frac{\text{Premise(s)}}{\text{Conclusion}}$$

  - meaning: if every premise is fulfilled, then conclusion can be drawn
  - a rule with no premises is called an axiom

Derivation Rules

- Definition employs derivation rules of the form

$$\text{Name}\ \frac{\text{Premise(s)}}{\text{Conclusion}}$$

  - meaning: if every premise is fulfilled, then conclusion can be drawn
  - a rule with no premises is called an axiom
- Iterated application yields complete derivation tree
  - initial program and state at root
  - premises as children of inner nodes
  - axioms at leafs

Execution of Statements I

Definition 10.9 (Execution relation for statements)

If $c \in \mathit{Cmd}_\downarrow$ for $\mathit{TCmd} := \mathit{Cmd} \cup \{\downarrow\}$ and $\sigma \in \Sigma$, then $\langle c, \sigma \rangle$ is called a configuration. The execution relation

$$\rightarrow \subseteq (\mathit{Cmd} \times \Sigma) \times (\mathit{Cmd}_\downarrow \times \Sigma)$$

is defined by the following rules:

$$\text{(skip)}\ \frac{}{\langle \texttt{skip}, \sigma \rangle \rightarrow \langle \downarrow, \sigma \rangle}$$

$$\text{(asgn)}\ \frac{}{\langle x := a, \sigma \rangle \rightarrow \langle \downarrow, \sigma[x \mapsto \mathit{val}_\sigma(a)] \rangle}$$

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

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

Execution of Statements II

Definition 10.9 (Execution relation for statements; continued)

$$\text{(if1)}\ \frac{val_\sigma(b) = \text{true}}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma\rangle \to \langle c_1, \sigma\rangle}$$

$$\text{(if2)}\ \frac{val_\sigma(b) = \text{false}}{\langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}, \sigma\rangle \to \langle c_2, \sigma\rangle}$$

$$\text{(wh1)}\ \frac{val_\sigma(b) = \text{true}}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma\rangle \to \langle c\,;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma\rangle}$$

$$\text{(wh2)}\ \frac{val_\sigma(b) = \text{false}}{\langle \texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}, \sigma\rangle \to \langle \downarrow, \sigma\rangle}$$

Remark: $\downarrow$ indicates successful termination of the program

An Execution Example

Example 10.10

- $\underbrace{\texttt{y := 1; while } \underbrace{\neg\texttt{(x=1)}}_{b} \texttt{ do } \underbrace{\underbrace{\texttt{y := y*x}}_{c_1}\texttt{; }\underbrace{\texttt{x := x-1}}_{c_2}}_{c_0} \texttt{ end}}_{c}$
- Claim: $\langle c, \sigma\rangle \to^+ \langle\downarrow, \sigma_{1,6}\rangle$ for every $\sigma \in \Sigma$ with $\sigma(\mathtt{x}) = 3$
- Notation: $\sigma_{i,j}$ means $\sigma(\mathtt{x}) = i$, $\sigma(\mathtt{y}) = j$
- Derivation: on the board

Determinism Property of Execution Relation

This operational semantics is well defined in the following sense:

Theorem 10.11

The execution relation for statements is deterministic, i.e., whenever $c \in \mathit{Cmd}$, $\sigma \in \Sigma$ *and* $\kappa_1, \kappa_2 \in \mathit{Cmd}_\downarrow \times \Sigma$ *such that* $\langle c, \sigma \rangle \to \kappa_1$ *and* $\langle c, \sigma \rangle \to \kappa_2$, *then* $\kappa_1 = \kappa_2$.

Proof.

omitted □

Determinism Property of Execution Relation

This operational semantics is well defined in the following sense:

Theorem 10.11

The execution relation for statements is deterministic, i.e., whenever $c \in \mathit{Cmd}$, $\sigma \in \Sigma$ *and* $\kappa_1, \kappa_2 \in \mathit{Cmd}_\downarrow \times \Sigma$ *such that* $\langle c, \sigma \rangle \to \kappa_1$ *and* $\langle c, \sigma \rangle \to \kappa_2$, *then* $\kappa_1 = \kappa_2$.

Proof.

omitted □

More on formal semantics of programming languages:
Semantics and Verification of Software in next summer semester

Outline of Lecture 10

Safe Approximation of Functions I

Definition 10.12 (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**
$\vec{m}$	$\xrightarrow{\gamma}$	$\gamma(\vec{m})$
$\downarrow f^\#$		$\downarrow f$
$f^\#(\vec{m}) \sqsupseteq \alpha(f(\gamma(\vec{m})))$	$\xleftarrow{\alpha}$	$f(\gamma(\vec{m}))$

Safe Approximation of Functions I

Definition 10.12 (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 10.14)

Safe Approximation of Functions II

Example 10.13 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 10.2): $L = (2^{\mathbb{Z}}, \subseteq)$, $M = (2^{\{\text{even},\text{odd}\}}, \subseteq)$
   - $n = 0$: for $f = \text{one} \subseteq 2^{\mathbb{Z}} : () \mapsto \{1\}$,
     - $\text{one}^\#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}, \text{odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \not\subseteq \{\text{even}\} = \text{one}^\#()$

Safe Approximation of Functions II

Example 10.13 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 10.2): $L = (2^{\mathbb{Z}}, \subseteq)$, $M = (2^{\{\text{even},\text{odd}\}}, \subseteq)$
   - $n = 0$: for $f = \text{one} \subseteq 2^{\mathbb{Z}} : () \mapsto \{1\}$,
     - $\text{one}^\#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}, \text{odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \not\subseteq \{\text{even}\} = \text{one}^\#()$
   - $n = 1$: for $f = \text{dec} : 2^{\mathbb{Z}} \to 2^{\mathbb{Z}} : Z \mapsto \{z - 1 \mid z \in Z\}$,
     - $\text{dec}^\#(\{\text{even}\}) = \{\text{odd}\}$ is most precise: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} = \text{dec}^\#(\{\text{even}\})$
     - $\text{dec}^\#(\{\text{even}\}) = \{\text{odd}, \text{even}\}$ is (only) safe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \subsetneq \{\text{odd}, \text{even}\} = \text{dec}^\#(\{\text{even}\})$
     - $\text{dec}^\#(\{\text{even}\}) = \emptyset$ is unsafe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \not\subseteq \emptyset = \text{dec}^\#(\{\text{even}\})$

Safe Approximation of Functions II

Example 10.13 (Safeness: $\alpha(f(\gamma(m_1), \ldots, \gamma(m_n))) \sqsubseteq_M f^\#(m_1, \ldots, m_n)$)

1. Parity abstraction (cf. Example 10.2): $L = (2^{\mathbb{Z}}, \subseteq)$, $M = (2^{\{\text{even},\text{odd}\}}, \subseteq)$
   - $n = 0$: for $f = \text{one} \subseteq 2^{\mathbb{Z}} : () \mapsto \{1\}$,
     - $\text{one}^\#() = \{\text{odd}\}$ is most precise: $\alpha(\{1\}) = \{\text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}, \text{odd}\}$ is (only) safe: $\alpha(\{1\}) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \text{one}^\#()$
     - $\text{one}^\#() = \{\text{even}\}$ is unsafe: $\alpha(\{1\}) = \{\text{odd}\} \not\subseteq \{\text{even}\} = \text{one}^\#()$
   - $n = 1$: for $f = \text{dec} : 2^{\mathbb{Z}} \to 2^{\mathbb{Z}} : Z \mapsto \{z - 1 \mid z \in Z\}$,
     - $\text{dec}^\#(\{\text{even}\}) = \{\text{odd}\}$ is most precise: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} = \text{dec}^\#(\{\text{even}\})$
     - $\text{dec}^\#(\{\text{even}\}) = \{\text{odd}, \text{even}\}$ is (only) safe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \subsetneq \{\text{odd}, \text{even}\} = \text{dec}^\#(\{\text{even}\})$
     - $\text{dec}^\#(\{\text{even}\}) = \emptyset$ is unsafe: $\alpha(\text{dec}(\gamma(\{\text{even}\}))) = \{\text{odd}\} \not\subseteq \emptyset = \text{dec}^\#(\{\text{even}\})$
   - $n = 2$: for $f = + : 2^{\mathbb{Z}} \times 2^{\mathbb{Z}} \to 2^{\mathbb{Z}} : (z_1, z_2) \mapsto z_1 + z_2$,
     - $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{odd}\}$ is m.p.: $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} = \{\text{even}\} +^\# \{\text{odd}\}$
     - $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{even}, \text{odd}\}$ is (only) safe: $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} \subsetneq \{\text{even}, \text{odd}\} = \{\text{even}\} +^\# \{\text{odd}\}$
     - $\{\text{even}\} +^\# \{\text{odd}\} = \{\text{even}\}$ is unsafe: $\alpha(\gamma(\{\text{even}\}) + \gamma(\{\text{odd}\})) = \{\text{odd}\} \not\subseteq \{\text{even}\} = \{\text{even}\} +^\# \{\text{odd}\}$

Safe Approximation of Functions III

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

Example 10.13 (continued)

Most precise approximations (with $L = (2^{\mathbb{Z}}, \subseteq)$):

2. Sign abstraction (cf. Example 10.3): $M = (2^{\{+,-,0\}}, \subseteq)$
   - $n = 0$: $\text{one}^\#() = \{+\}$
   - $n = 1$: $\text{dec}^\#(\{+\}) = \{+, 0\}$, $\quad -^\#(\{+\}) = \{-\}$
   - $n = 2$: $\{+\} +^\# \{+\} = \{+\}$, $\quad \{+\} -^\# \{+\} = \{+, -, 0\}$, $\quad \{+\} \cdot^\# \{-\} = \{-\}$

Safe Approximation of Functions III

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

Example 10.13 (continued)

Most precise approximations (with $L = (2^{\mathbb{Z}}, \subseteq)$):

2. Sign abstraction (cf. Example 10.3): $M = (2^{\{+,-,0\}}, \subseteq)$
   - $n = 0$: $\text{one}^\#() = \{+\}$
   - $n = 1$: $\text{dec}^\#(\{+\}) = \{+, 0\}$, $\quad -^\#(\{+\}) = \{-\}$
   - $n = 2$: $\{+\} +^\# \{+\} = \{+\}$, $\quad \{+\} -^\# \{+\} = \{+, -, 0\}$, $\quad \{+\} \cdot^\# \{-\} = \{-\}$
3. Interval abstraction (cf. Example 10.4): $M = ((\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{+\infty\}) \cup \{\emptyset\}, \subseteq)$
   - $n = 0$: $\text{one}^\#() = [1, 1]$
   - $n = 1$: $\text{dec}^\#([z_1, z_2]) = [z_1 - 1, z_2 - 1]$, $\quad -^\#([z_1, z_2]) = [-z_2, -z_1]$
   - $n = 2$: $[y_1, y_2] +^\# [z_1, z_2] = [y_1 + z_1, y_2 + z_2]$
     $[y_1, y_2] -^\# [z_1, z_2] = [y_1 - z_2, y_2 - z_1]$
     $[y_1, y_2] \cdot^\# [z_1, z_2] = [\bigsqcap\{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}, \bigsqcup\{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}]$

   (thus, $+^\#/-^\#/\cdot^\# = \oplus/\ominus/\odot$ from Slide 7.8)

Safe Approximation of Functions IV

Lemma 10.14

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* $l_1, \ldots, l_n \in L$,

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

Safe Approximation of Functions IV

Lemma 10.14

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 □