Static Program Analysis

Lecture 14: Abstract Interpretation III (Abstract Interpretation of WHILE Programs)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/spa11/

Summer Semester 2011

Outline

Safe Approximation of Functions

Definition

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 it is called most precise safe approximation if the reverse inclusion is also true.

- **Interpretation:** the abstraction $f^\#$ of f covers all concrete results
- **Note:** monotonicity of f and/or $f^\#$ is *not* required (but usually given; see Lemma 13.5)

Safe Approximation of Execution Relation I

- **Reminder:** concrete semantics of WHILE
  - states $\Sigma := \{\sigma \mid \sigma : \mathit{Var} \to \mathbb{Z}\}$ (Definition 12.6)
  - execution relation $\to \subseteq (\mathit{Cmd} \times \Sigma) \times (\mathit{Cmd} \times \Sigma \cup \Sigma)$ (Definition 12.9)
- Yields concrete domain $L := 2^{\Sigma}$ 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 $S \subseteq \Sigma$,

$$\mathrm{next}_{c,c'}(S) := \{\sigma' \in \Sigma \mid c' \in \mathit{Cmd}, \exists \sigma \in S : \langle c, \sigma \rangle \to \langle c', \sigma' \rangle\} \text{ and}$$
$$\mathrm{next}_{c,\downarrow}(S) := \{\sigma' \in \Sigma \mid \exists \sigma \in S : \langle c, \sigma \rangle \to \sigma'\}$$

Safe Approximation of Execution Relation II

- **Reminder:** abstraction determined by Galois connection (α, γ) with $\alpha : L \to M$ and $\gamma : M \to L$
  - here: $L := 2^{\Sigma}$, M not fixed (usually $M = Var \to \ldots$ or $M = 2^{Var \to \cdots}$)
  - write Abs in place of M
  - thus $\alpha : 2^{\Sigma} \to Abs$ and $\gamma : Abs \to 2^{\Sigma}$
- Yields abstract semantics:

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 $\text{next}^{\#}_{c,c'}(abs) = abs'$ and
- $\langle c, abs \rangle \Rightarrow abs'$ for $\text{next}^{\#}_{c,\downarrow}(a) = abs'$

Outline

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\mathtt{n}\}$
- Notation: $[\mathtt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$
$$\langle \texttt{n := 2 * n + 1}, \{[\texttt{n} \mapsto \text{even}], [\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$
$$\langle \texttt{n := 2 * n + 1}, \{[\texttt{n} \mapsto \text{even}], [\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$
$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

$$\langle \texttt{n := 2 * n + 1}, \{[\texttt{n} \mapsto \text{even}], [\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \langle \texttt{c; while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle$$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

$$\langle \texttt{n := 2 * n + 1}, \{[\texttt{n} \mapsto \text{even}], [\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \langle \texttt{c; while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{even}]\}\rangle \not\Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

Safe Approximation of Execution Relation III

Example 14.1 (Parity abstraction (cf. Example 12.2))

- $Abs = 2^{Var \to \{\text{even},\text{odd}\}}$
- $Var = \{\texttt{n}\}$
- Notation: $[\texttt{n} \mapsto p] \in abs \in Abs$ for $p \in \{\text{even}, \text{odd}\}$
- Some abstract transitions:

$$\langle \texttt{n := 3 * n + 1}, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

$$\langle \texttt{n := 2 * n + 1}, \{[\texttt{n} \mapsto \text{even}], [\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \{[\texttt{n} \mapsto \text{odd}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle \Rightarrow \langle c\texttt{; while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{odd}]\}\rangle$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{even}]\}\rangle \not\Rightarrow \{[\texttt{n} \mapsto \text{even}]\}$$

$$\langle \texttt{while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{even}]\}\rangle \Rightarrow \langle c\texttt{; while } \neg\texttt{(n=1) do } c, \{[\texttt{n} \mapsto \text{even}]\}\rangle$$

Example: Hailstone Sequences

Example 14.2 (Hailstone Sequences)

```
[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;
```

- additional `skip` statements only for labels
- abstract transition system for $n \in \mathbb{Z}_{\text{odd}}$: on the board

Example: Hailstone Sequences

Example 14.2 (Hailstone Sequences)

```
[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;
```

- additional `skip` statements only for labels
- abstract transition system for $n \in \mathbb{Z}_{\text{odd}}$: on the board

- Collatz Conjecture: given any $n > 0$, the program finally returns 1 (that is, every Hailstone Sequence terminates with 1)
- see `http://en.wikipedia.org/wiki/Collatz_conjecture`
- AKA $3n + 1$ Conjecture, Ulam Conjecture, Kakutani's Problem, Thwaites' Conjecture, Hasse's Algorithm, or Syracuse Problem
- New proof attempt by Gerhard Opfer from Hamburg University (`http://preprint.math.uni-hamburg.de/public/papers/hbam/hbam2011-09.pdf`)

Outline

Derivation of Abstract Semantics

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

Example 14.3 (Fermat's Last Theorem)

Sign abstraction (cf. Example 12.3) on

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

Derivation of Abstract Semantics

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

Example 14.3 (Fermat's Last Theorem)

Sign abstraction (cf. Example 12.3) on

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

- Result $\texttt{n} = 1$ possible iff there exist $n > 2$ and $x, y, z \geq 1$ such that $x^n + y^n = z^n$
- 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 14.3 (Fermat's Last Theorem)

Sign abstraction (cf. Example 12.3) on

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

- Result $\texttt{n} = 1$ possible iff there exist $n > 2$ and $x, y, z \geq 1$ such that $x^n + y^n = z^n$
- Fermat's Last Theorem: equation not solvable
- Final proof by Andrew Wiles and Richard Taylor in 1995

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

Extraction Functions

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

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

and

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

Extraction Functions

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

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

and

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

Example 14.4

1. Parity abstraction (cf. Example 12.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ 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 elements
- 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

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

and

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

Example 14.4

1. Parity abstraction (cf. Example 12.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ 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 12.3): $\beta : \mathbb{Z} \to \{+, -, 0\}$ with $\beta = \text{sgn}$

Extraction Functions

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

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

and

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

Example 14.4

1. Parity abstraction (cf. Example 12.2): $\beta : \mathbb{Z} \to \{\text{even}, \text{odd}\}$ 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 12.3): $\beta : \mathbb{Z} \to \{+, -, 0\}$ with $\beta = \text{sgn}$
3. Interval abstraction (cf. Example 12.4): not definable by extraction function (as *Int* is not of the form 2^A)

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 13.3)

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

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 13.3)

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

Theorem 14.5

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.

Safe Approximation by Extraction Functions

Reminder: safe approximation condition (Definition 13.3)

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

Theorem 14.5

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 14.6 (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.

Example 14.7 (Sign abstraction)

1. Relational operations:
   - $C = \mathbb{Z} \cup \mathbb{B}$, $A = \{+, -, 0\} \cup \mathbb{B}$, $\beta = \text{sgn}$
$=^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$	$\{\text{false}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{false}\}$	$\{\text{true}\}$
$>^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{true}\}$	$\{\text{true}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{true}\}$	$\{\text{false}\}$
   - $\{+, 0\} =^\# \{0\} = \{+\} =^\# \{0\} \cup \{0\} =^\# \{0\} = \{\text{false}\} \cup \{\text{true}\} = \{\text{true}, \text{false}\}$ etc.

Safe Approximation of Boolean Operations

Example 14.7 (Sign abstraction)

1. Relational operations:
   - $C = \mathbb{Z} \cup \mathbb{B}$, $A = \{+, -, 0\} \cup \mathbb{B}$, $\beta = \text{sgn}$
$=^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$	$\{\text{false}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{false}\}$	$\{\text{true}\}$
$>^\#$	$\{+\}$	$\{-\}$	$\{0\}$
$\{+\}$	$\{\text{true}, \text{false}\}$	$\{\text{true}\}$	$\{\text{true}\}$
$\{-\}$	$\{\text{false}\}$	$\{\text{true}, \text{false}\}$	$\{\text{false}\}$
$\{0\}$	$\{\text{false}\}$	$\{\text{true}\}$	$\{\text{false}\}$
   - $\{+, 0\} =^\# \{0\} = \{+\} =^\# \{0\} \cup \{0\} =^\# \{0\} = \{\text{false}\} \cup \{\text{true}\} = \{\text{true}, \text{false}\}$ etc.
2. Boolean connectives:
   - $C = A = \mathbb{B}$, $\neg^\# = \neg$, $\wedge^\# = \wedge$, ...
   - $\{\text{true}, \text{false}\} \wedge^\# \{\text{true}\} = \{\text{true}\} \wedge^\# \{\text{true}\} \cup \{\text{false}\} \wedge^\# \{\text{true}\} = \{\text{true}\} \cup \{\text{false}\} = \{\text{true}, \text{false}\}$ etc.

Abstract Program States

Now: take values of variables into account

Definition 14.8 (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 14.8 (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 14.8 (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 14.9 (Abstract evaluation functions)

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

1. $val^{\#}_{\rho} : AExp \to 2^A$ is determined by

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

Abstract Evaluation of Expressions

Definition 14.9 (Abstract evaluation functions)

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

1. $val^{\#}_{\rho} : AExp \to 2^{A}$ is determined by

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

2. $val^{\#}_{\rho} : BExp \to 2^{\mathbb{B}}$ is determined by

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

Abstract Evaluation of Expressions

Definition 14.9 (Abstract evaluation functions)

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

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

Example 14.10 (Sign abstraction)

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

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