

Introduction to Static Analysis

Chapter 3 前半

山本 航平

この章 (前半) でやること

- ・ この章で用いるプログラミング言語とその具体的な意味を定義
- ・ 抽象化の方法
 - ・ 抽象をどう表すか？
 - ・ 抽象と具体の関係は？

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

3.1.1 A Simple Programming Language

- 静的解析を説明する上で簡単なプログラミング言語を定義する

V : 実数の集合

X : 変数の有限集合

$B = \{\text{true}, \text{false}\}$: 真理値の集合

3.1.1 A Simple Programming Language

- 静的解析を説明する上で簡単なプログラミング言語を定義する

$n \in \mathbb{V}$	$C ::=$
$x \in \mathbb{X}$	$ \quad \text{skip}$
$\odot ::= + - * ...$	$ \quad C; C$
$\oslash ::= < \leq == ...$	$ \quad x := E$
$E ::=$	$ \quad \text{input}(x)$
$ \quad n$	$ \quad \text{if}(B)\{C\}\text{else}\{C\}$
$ \quad x$	$ \quad \text{while}(B)\{C\}$
$ \quad E \odot E$	$P ::= C$
$B ::=$	
$ \quad x \oslash n$	

3.1.1 A Simple Programming Language

$n \in \mathbb{V}$

n : 実数

$x \in \mathbb{X}$

x : プログラムの変数

$\odot ::= + | - | * | \dots$

二項演算子

$\oslash ::= < | \leq | == | \dots$

比較演算子

$E ::=$

実数の計算

| n

| x

| $E \odot E$

$B ::=$

真理値の計算

| $x \oslash n$

3.1.1 A Simple Programming Language

$C ::=$

- | **skip** 何もしない
- | $C; C$ コマンドの列
- | $x := E$ 代入
- | **input(x)** 入力の読み取り
- | **if(B){ C }else{ C }** 条件分岐
- | **while(B){ C }** 繰り返し

$P ::= C$ プログラム

3.1.1 A Simple Programming Language

プログラムの例

```
input(x);
y := 0;
while (y < 5) {
    if (x < 0) {
        x := x + 1
    } else {
        skip
    };
    y := y + 1
}
```

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

3.1.2 Concrete Semantics

Concrete Semantics

- ・ プログラムの動作を定義する
- ・ これをもとに静的解析を設計し, soundness を証明する
 - 調べたい性質を証明するのに必要なすべての論理的事実を表現する必要あり

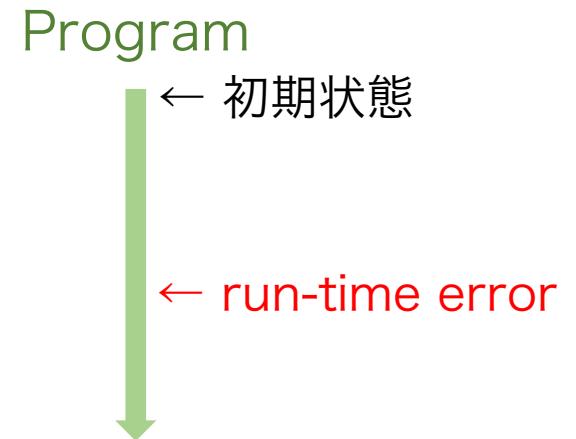
Trace semantics : プログラム状態を連続して記述する

Denotational semantics : 入力と出力の関係だけを記述する

Family of Properties of Interest

どのような性質に着目するか？

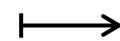
- この章では到達可能性 (reachability) に注目する
→ 状態集合 R に含まれない状態には到達しない
ex.) run-time error, user assertion が発生しないか？
- プログラムの最終状態の到達可能性だけ見ることもできる
ex.) プログラムが絶対に非負の値を返すか？
→ 入力と出力の関係のみを表す semantics で良い
- 到達可能性と事後条件 (post-condition) には深い関係性



An Input-Output Semantics

program semantics :

set of input states



set of output states

input コマンドによって
無数の出力

- Denotational Semantics : 入力と出力の関係だけを記述する
- インタプリタの動作に似ている
- Compositional である
 - 命令の semantics はサブ命令の semantics によって定義される

Memory States

program semantics :

set of input states \longrightarrow set of output states

状態をどう表すか？

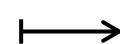
memory state : メモリの状態

control state : 現在のプログラムの位置

Memory States

program semantics :

set of input states



set of output states

状態をどう表すか？

memory state : メモリの状態

~~control state : 現在のプログラムの位置~~

input と output の状態には関係ないので無視

Memory States

メモリ状態 (memory state)

$$\mathbb{M} = \mathbb{X} \longrightarrow \mathbb{V}$$

\mathbb{M} : メモリ状態の集合

\mathbb{X} : 変数の集合

\mathbb{V} : 値の集合

- ここで使うプログラミング言語の変数には実数しか入らない
- ex.) 変数 x が 7, 変数 y が 3 のときのメモリ状態

$$\{x \mapsto 7, y \mapsto 3\}$$

Semantics of Scalar Expressions

$$\llbracket E \rrbracket : \mathbb{M} \longrightarrow \mathbb{V}$$

$$\llbracket n \rrbracket(m) = n$$

$$\llbracket x \rrbracket(m) = m(x)$$

$$\llbracket E_0 \odot E_1 \rrbracket(m) = f_{\odot}(\llbracket E_0 \rrbracket(m), \llbracket E_1 \rrbracket(m))$$

$\llbracket E \rrbracket(m)$: メモリ状態が m のとき式 E の semantics

$\llbracket E \rrbracket : \mathbb{M} \longrightarrow \mathbb{V}$: メモリ状態から実数値への関数

$m(x)$: メモリ状態 m における変数 x の値

f_{\odot} : 演算子 \odot の計算を行う関数

Semantics of Boolean Expressions

$$\llbracket B \rrbracket : \mathbb{M} \longrightarrow \mathbb{B}$$

$$\llbracket x \odot n \rrbracket(m) = f_{\odot}(m(x), n)$$

$\llbracket B \rrbracket(m)$: メモリ状態が m のとき式 B の semantics

$\llbracket B \rrbracket : \mathbb{M} \longrightarrow \mathbb{B}$: メモリ状態から真理値への関数

f_{\odot} : 演算子 \odot の計算を行う関数

Semantics of Commands

$$\llbracket C \rrbracket_{\mathcal{P}} : \wp(\mathbb{M}) \longrightarrow \wp(\mathbb{M})$$

$$\llbracket \text{skip} \rrbracket_{\mathcal{P}}(M) = M$$

$$\llbracket C_0; C_1 \rrbracket_{\mathcal{P}}(M) = \llbracket C_1 \rrbracket_{\mathcal{P}}(\llbracket C_0 \rrbracket_{\mathcal{P}}(M))$$

$$\llbracket x := E \rrbracket_{\mathcal{P}}(M) = \{m[x \mapsto \llbracket E \rrbracket(m)] \mid m \in M\}$$

$$\llbracket \text{input}(x) \rrbracket_{\mathcal{P}}(M) = \{m[x \mapsto n] \mid m \in M, n \in \mathbb{V}\}$$

$$\llbracket \text{if}(B)\{C_0\} \text{else}\{C_1\} \rrbracket_{\mathcal{P}}(M) = \llbracket C_0 \rrbracket_{\mathcal{P}}(\mathcal{F}_B(M)) \cup \llbracket C_1 \rrbracket_{\mathcal{P}}(\mathcal{F}_{\neg B}(M))$$

$$\llbracket \text{while}(B)\{C\} \rrbracket_{\mathcal{P}}(M) = \mathcal{F}_{\neg B} \left(\bigcup_{i \geq 0} (\llbracket C \rrbracket_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right)$$

$\llbracket C \rrbracket_{\mathcal{P}} : \wp(\mathbb{M}) \longrightarrow \wp(\mathbb{M})$: input 状態の集合から output 状態の集合への関数

$\wp(\mathbb{M})$: メモリ状態のべき集合

M : $\wp(\mathbb{M})$ の要素

Semantics of Commands

$$\llbracket \text{skip} \rrbracket_{\mathcal{P}}(M) = M \quad \text{恒等関数}$$

$$\llbracket C_0; C_1 \rrbracket_{\mathcal{P}}(M) = \llbracket C_1 \rrbracket_{\mathcal{P}}(\llbracket C_0 \rrbracket_{\mathcal{P}}(M))$$

それぞれの命令の semantics の組み合わせ

$$\llbracket x := E \rrbracket_{\mathcal{P}}(M) = \{m[x \mapsto \llbracket E \rrbracket(m)] \mid m \in M\}$$

メモリ状態の x の値を E の結果にする

$$\llbracket \text{input}(x) \rrbracket_{\mathcal{P}}(M) = \{m[x \mapsto n] \mid m \in M, n \in \mathbb{V}\}$$

メモリ状態の x の値を可能性のある実数値にする

Semantics of Commands

$$\llbracket \text{if}(B)\{C_0\}\text{else}\{C_1\} \rrbracket_{\mathcal{P}}(M) = \llbracket C_0 \rrbracket_{\mathcal{P}}(\mathcal{F}_B(M)) \cup \llbracket C_1 \rrbracket_{\mathcal{P}}(\mathcal{F}_{\neg B}(M))$$

“filtering” function : \mathcal{F}_B

$$\mathcal{F}_B(M) = \{m \in M \mid \llbracket B \rrbracket(m) = \text{true}\}$$

$\llbracket B \rrbracket(m)$ が **true** となるメモリ状態の集合を返す関数

$$\underline{\llbracket C_0 \rrbracket_{\mathcal{P}}(\mathcal{F}_B(M))} \quad \cup \quad \underline{\llbracket C_1 \rrbracket_{\mathcal{P}}(\mathcal{F}_{\neg B}(M))}$$

条件を満たすものの semantics

条件を満たさないものの semantics



和集合

Example

```
C : if (x > 0) {  
    y := y + x  
} else {  
    y := y - x  
}
```

$$\mathbb{X} = \{\mathbf{x}, \mathbf{y}\}$$

$$M = \{\{\mathbf{x} \mapsto 3, \mathbf{y} \mapsto 0\},
 \{\mathbf{x} \mapsto -3, \mathbf{y} \mapsto 1\}\}$$

このとき $\llbracket C \rrbracket_{\mathcal{P}}(M)$ は？

Example

```
C : if (x > 0) {  
    y := y + x  
} else {  
    y := y - x  
}
```

$\mathbb{X} = \{\mathbf{x}, \mathbf{y}\}$

$M = \{\{\mathbf{x} \mapsto 3, \mathbf{y} \mapsto 0\},$
 $\{\mathbf{x} \mapsto -3, \mathbf{y} \mapsto 1\}\}$

このとき $\llbracket C \rrbracket_{\mathcal{P}}(M)$ は？

$$\llbracket C \rrbracket_{\mathcal{P}}(M) = \llbracket \mathbf{y} := \mathbf{y} + \mathbf{x} \rrbracket_{\mathcal{P}}(\mathcal{F}_{\mathbf{x} > 0}(M)) \cup \\ \llbracket \mathbf{y} := \mathbf{y} - \mathbf{x} \rrbracket_{\mathcal{P}}(\mathcal{F}_{\mathbf{x} \leq 0}(M))$$

Q.1 $\mathcal{F}_{\mathbf{x} > 0}(M) = \{\{\mathbf{x} \mapsto 3, \mathbf{y} \mapsto 0\}\}$

$$\llbracket \mathbf{y} := \mathbf{y} + \mathbf{x} \rrbracket_{\mathcal{P}}(\mathcal{F}_{\mathbf{x} > 0}(M)) \\ = \{m[\mathbf{y} \mapsto \llbracket \mathbf{y} + \mathbf{x} \rrbracket(m)] \mid m \in \mathcal{F}_{\mathbf{x} > 0}(M)\}$$

$m = \{\mathbf{x} \mapsto 3, \mathbf{y} \mapsto 0\}$ として

Q.2 $\llbracket \mathbf{y} + \mathbf{x} \rrbracket(m) = f_+(\llbracket \mathbf{y} \rrbracket(m), \llbracket \mathbf{x} \rrbracket(m))$
 $= f_+(m(\mathbf{y}), m(\mathbf{x}))$
 $= f_+(0, 3)$
 $= 3$

$$\llbracket \mathbf{y} := \mathbf{y} + \mathbf{x} \rrbracket_{\mathcal{P}}(\mathcal{F}_{\mathbf{x} > 0}(M)) = \{\{\mathbf{x} \mapsto 3, \mathbf{y} \mapsto 3\}\}$$

Example

```
C : if (x > 0) {  
    y := y + x  
} else {  
    y := y - x  
}
```

$\mathbb{X} = \{x, y\}$

$M = \{\{x \mapsto 3, y \mapsto 0\},$
 $\{x \mapsto -3, y \mapsto 1\}\}$

このとき $\llbracket C \rrbracket_{\mathcal{P}}(M)$ は？

$$\llbracket C \rrbracket_{\mathcal{P}}(M) = \llbracket y := y + x \rrbracket_{\mathcal{P}}(\mathcal{F}_{x>0}(M)) \cup \\ \llbracket y := y - x \rrbracket_{\mathcal{P}}(\mathcal{F}_{x \leq 0}(M))$$

$$\llbracket y := y + x \rrbracket_{\mathcal{P}}(\mathcal{F}_{x>0}(M)) = \{\{x \mapsto 3, y \mapsto 3\}\}$$

$$\llbracket y := y - x \rrbracket_{\mathcal{P}}(\mathcal{F}_{x \leq 0}(M)) = \{\{x \mapsto -3, y \mapsto 4\}\}$$

$$\llbracket C \rrbracket_{\mathcal{P}}(M) = \{\{x \mapsto 3, y \mapsto 3\}, \\ \{x \mapsto -3, y \mapsto 4\}\}$$

Semantics of Commands

$$[\![\text{while}(B)\{C\}]\!]_{\mathcal{P}}(M) = \mathcal{F}_{\neg B} \left(\bigcup_{i \geq 0} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right)$$

直感的には条件 B が i 回満たされ、最後にその条件を満たさない

ループをちょうど i 回通るプログラムの実行後のメモリ状態を M_i とする

$$M_i = \underline{\mathcal{F}_{\neg B} \left(([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right)} \quad i \text{ 回以上ループ内を通るメモリ状態の集合}$$

条件 B を満たさないメモリ状態の集合

$$\begin{aligned} [\![\text{while}(B)\{C\}]\!]_{\mathcal{P}}(M) &= \bigcup_{i \geq 0} M_i \\ &= \bigcup_{i \geq 0} \mathcal{F}_{\neg B} \left(([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right) \\ &= \mathcal{F}_{\neg B} \left(\bigcup_{i \geq 0} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right) \end{aligned}$$

Example

$C :$

```
while (x < 2) {
    x := x + 1
}
```

$$\mathbb{X} = \{\mathbf{x}\}$$

$$M = \{\{\mathbf{x} \mapsto 0\}, \\ \{\mathbf{x} \mapsto 1\}, \\ \{\mathbf{x} \mapsto 2\}\}$$

このとき $\llbracket C \rrbracket_{\mathcal{P}}(M)$ は？

Example

$C :$

```
while (x < 2) {
    x := x + 1
}
```

$$\mathbb{X} = \{\mathbf{x}\}$$

$$M = \{\{\mathbf{x} \mapsto 0\}, \{\mathbf{x} \mapsto 1\}, \{\mathbf{x} \mapsto 2\}\}$$

このとき $\llbracket C \rrbracket_{\mathcal{P}}(M)$ は？

Q.1 $(\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^0(M)$
 $= \{\{\mathbf{x} \mapsto 0\}, \{\mathbf{x} \mapsto 1\}, \{\mathbf{x} \mapsto 2\}\}$

$$(\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^1(M)$$
$$= \{\{\mathbf{x} \mapsto 1\}, \{\mathbf{x} \mapsto 2\}\}$$

$$(\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^2(M) = \{\{\mathbf{x} \mapsto 2\}\}$$

$$(\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^3(M) = \emptyset$$

Q.2 $\bigcup_{i \geq 0} (\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^i(M)$
 $= \{\{\mathbf{x} \mapsto 0\}, \{\mathbf{x} \mapsto 1\}, \{\mathbf{x} \mapsto 2\}\}$

Q.3

$$\llbracket C \rrbracket_{\mathcal{P}}(M) = \mathcal{F}_{\mathbf{x} \geq 2} \left(\bigcup_{i \geq 0} (\llbracket \mathbf{x} := \mathbf{x} + 1 \rrbracket_{\mathcal{P}} \circ \mathcal{F}_{\mathbf{x} < 2})^i(M) \right)$$
$$= \{\{\mathbf{x} \mapsto 2\}\}$$

Semantics of Commands

`while` の最小不動点を使った表し方

$$[\![\text{while}(B)\{C\}]\!]_{\mathcal{P}}(M) = \mathcal{F}_{\neg B} \left(\bigcup_{i \geq 0} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right)$$

$$\begin{aligned} M_{loop} &= \bigcup_{i \geq 0} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) = M \cup \left(\bigcup_{i \geq 1} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right) \\ &= M \cup [\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B \left(\bigcup_{i \geq 0} ([\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B)^i(M) \right) \end{aligned}$$

- $M \cup [\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B(M_{loop}) = M_{loop}$ は関数 $G : X \mapsto M \cup [\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B(X)$ の不動点
- この等式を満たす最小のメモリ集合 (関数 G の最小不動点) を求める

$$[\![\text{while}(B)\{C\}]\!]_{\mathcal{P}}(M) = \mathcal{F}_{\neg B}(\text{lfp } G)$$

$$G : X \mapsto M \cup [\![C]\!]_{\mathcal{P}} \circ \mathcal{F}_B(X) \quad \text{lfp } G : G \text{ の最小不動点}$$



鳥取砂丘

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

Intuitions Gathered from the Previous Chapter

抽象ドメイン (abstract domain) が与えるもの：

- ・ 抽象要素 (abstract element) の集合
抽象要素：プログラム状態の論理的性質を表す
- ・ 抽象要素を表すデータ構造
- ・ 抽象と具体的の関係
- ・ 解析アルゴリズム (post-condition, union, widening などを計算)

抽象ドメイン：プログラム解析で用いるドメイン

具体ドメイン：プログラムの実際の動作を定義するドメイン

Concrete Domain

具体ドメイン (concrete domain) : (\mathbb{C}, \subseteq)

具体要素 (プログラムの具体的な動作) の集合 : \mathbb{C}

順序関係 (order relation) : \subseteq

- $x, y \in \mathbb{C}$
 $x \subseteq y$: x ならば y (x の方が y より強い性質を表す)
- 具体ドメインをべき集合とするとき、その順序関係 \subseteq は集合の順序関係 \subseteq と同じ意味
- 到達可能性について調べたいとき $\mathbb{C} = \wp(\mathbb{M})$ とする

$$M_0 = \{m \in \mathbb{M} \mid 0 \leq m(x)\}, \quad M_1 = \{m \in \mathbb{M} \mid 1 \leq m(x)\}$$

$$M_1 \subseteq M_0$$

Abstract Domain

抽象ドメイン (abstract domain)

- ・要素同士を比較できる
- ・要素はそれぞれ具体的なレベルに変換できる

c を具体要素, a を抽象要素とする

$c \models a$: c は a が表す論理的性質を満たす

Abstract Domain

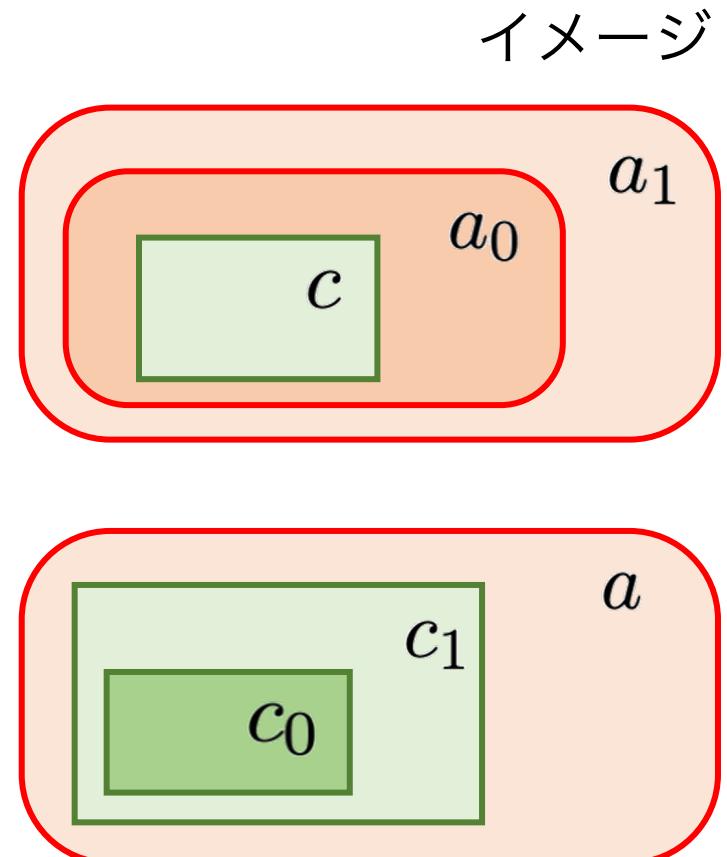
抽象ドメイン (abstract domain) : $(\mathbb{A}, \sqsubseteq)$

抽象要素 (プログラム状態の論理的性質) の集合 : \mathbb{A}

順序関係(order relation) : \sqsubseteq

- すべての $c \in \mathbb{C}, a_0, a_1 \in \mathbb{A}$ に対して
 $c \models a_0, a_0 \sqsubseteq a_1$ ならば $c \models a_1$

- すべての $c_0, c_1 \in \mathbb{C}, a \in \mathbb{A}$ に対して
 $c_0 \subseteq c_1, c_1 \models a$ ならば $c_0 \models a$



Example (Abstraction)

$\mathbb{X} = \{\mathbf{x}, \mathbf{y}\}$ $M_0, M_1 \in \mathbb{C}$, $M^\sharp \in \mathbb{A}$ とする

$$M_0 = \{m \in \mathbb{M} \mid 0 \leq m(\mathbf{x}) \leq m(\mathbf{y}) \leq 8\}$$

$$M_1 = \{m \in \mathbb{M} \mid 0 \leq m(\mathbf{x})\}$$

抽象要素 M^\sharp : \mathbf{x} の範囲 : 区間 $[0, 10]$

\mathbf{y} の範囲 : 区間 $[0, 80]$

Q. $M_0 \models M^\sharp$?

Q. $M_1 \models M^\sharp$?

Example (Abstraction)

$\mathbb{X} = \{\mathbf{x}, \mathbf{y}\}$ $M_0, M_1 \in \mathbb{C}$, $M^\sharp \in \mathbb{A}$ とする

$$M_0 = \{m \in \mathbb{M} \mid 0 \leq m(\mathbf{x}) \leq m(\mathbf{y}) \leq 8\}$$

$$M_1 = \{m \in \mathbb{M} \mid 0 \leq m(\mathbf{x})\}$$

抽象要素 M^\sharp : \mathbf{x} の範囲 : 区間 $[0, 10]$

\mathbf{y} の範囲 : 区間 $[0, 80]$

Q. $M_0 \models M^\sharp$? Yes

Q. $M_1 \models M^\sharp$? No $M_1 \not\models M^\sharp$

ex.) $\{\mathbf{x} \mapsto 100, \mathbf{y} \mapsto 50\} \in M_1$

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

Relation between Concrete Elements and Abstract Elements

- ・関係 \models を使って具体要素と抽象要素の関係を表す方法は常に最適ではない
- ・以下の2つの関数を用いてその関係を表す
 - ・具体化関数 (concretization function)
抽象要素からそれが表す具体要素への関数
 - ・抽象化関数 (abstraction function)
具体要素からそれを表す抽象要素への関数

Concretization Function

具体化関数 (concretization function) $\gamma : \mathbb{A} \longrightarrow \mathbb{C}$

- 抽象要素 a に対して $\gamma(a)$:

a を満たす具体ドメイン \mathbb{C} の最大の要素

$$\forall c \in \mathbb{C}, a \in \mathbb{A}, \quad c \models a \iff c \subseteq \gamma(a)$$

- 単調である

\sqsubset に対して大きな抽象要素はより大きな具体的な状態を占める

→ 静的解析の精度が落ちる

ex.)

抽象要素 M^\sharp : x の範囲: 区間 $[0, 10]$ y の範囲: 区間 $[0, 80]$

$$\gamma(M^\sharp) = \{m \in \mathbb{M} \mid 0 \leq m(x) \leq 10 \wedge 0 \leq m(y) \leq 80\}$$

Abstraction Function

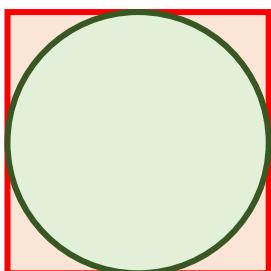
最適抽象： 具体要素 c に対して以下の性質を持つ抽象要素 a

(best abstraction)

- a は c の抽象化である
- その他の c の抽象化は a よりも大きい

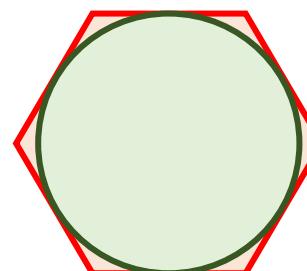
抽象化関数 (Abstraction function) $\alpha : \mathbb{C} \longrightarrow \mathbb{A}$

- 具体要素からその最適抽象への関数
- 単調である
- 常に抽象化関数が存在するとは限らない



区間抽象

最適な抽象化
が存在



凸多角形抽象

最適な抽象化
が存在しない

Galois Connection

ガロア接続 (Galois connection) :

以下を満たす具体化関数 γ と抽象化関数 α の組

$$\forall c \in \mathbb{C}, \forall a \in \mathbb{A}, \quad \alpha(c) \sqsubseteq a \iff c \subseteq \gamma(a)$$

この組の表し方 : $(\mathbb{C}, \subseteq) \xrightleftharpoons[\gamma]{\alpha} (\mathbb{A}, \sqsubseteq)$

性質

- γ と α は単調な関数である
- $\forall c \in \mathbb{C}, c \subseteq \gamma(\alpha(c))$
→ 元の具体要素よりも大きくなる
- $\forall a \in \mathbb{A}, \alpha(\gamma(a)) \sqsubseteq a$
→ 元の抽象要素よりも小さくなる (reduction)

Galois Connection

ガロア接続 (Galois connection) :

以下を満たす具体化関数 γ と抽象化関数 α の組

$$\forall c \in \mathbb{C}, \forall a \in \mathbb{A}, \quad \alpha(c) \sqsubseteq a \iff c \subseteq \gamma(a)$$

$$\forall a \in \mathbb{A}, \quad \alpha(\gamma(a)) \sqsubseteq a$$

証明

a を抽象要素とする

順序関係 \sqsubseteq の反射性から $\gamma(a) \subseteq \gamma(a)$

ガロア接続の定義より $\alpha(\gamma(a)) \sqsubseteq a$

Galois Connection

ガロア接続 (Galois connection) :

以下を満たす具体化関数 γ と抽象化関数 α の組

$$\forall c \in \mathbb{C}, \forall a \in \mathbb{A}, \quad \alpha(c) \sqsubseteq a \iff c \subseteq \gamma(a)$$

γ は単調

証明

a_0, a_1 を $a_0 \sqsubseteq a_1$ を満たす抽象要素とする

$\alpha(\gamma(a_0)) \sqsubseteq a_0$ より $\alpha(\gamma(a_0)) \sqsubseteq a_0 \sqsubseteq a_1$

ガロア接続の定義より $\gamma(a_0) \subseteq \gamma(a_1)$

Comparing Abstraction Formalizations

- ・抽象化関数が定義できるとき
抽象化関数を用いてプログラムを最適に抽象化して解析できる
- ・抽象化関数が定義できないとき
解析の結果, 論理的に比較不可能な複数の答えから選択する必要がある



出雲大社

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

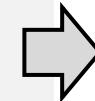
3.2.2 Non-Relational Abstraction

非関係抽象 (non-relational abstraction) :

すべての変数同士が関係を持たない抽象

抽象化のステップ

1. それぞれの変数が取り得る値を収集
2. 変数ごとに1つの抽象要素で過大近似
抽象要素は1つの変数に対する数値の制約とする



Value abstraction

値抽象 (value abstraction) : 具体ドメイン ($\wp(\mathbb{V})$, \subseteq) の抽象化

Example (Signs)

符号値抽象ドメイン $\mathbb{A}_{\mathcal{S}}$ の要素: $[\geq 0], [\leq 0], [= 0], \top, \perp$

\top : 値のすべての集合を表す

\perp : 値の空集合を表す

具体化関数 $\gamma_{\mathcal{S}}$: $[\geq 0] \mapsto \{n \in \mathbb{V} \mid n \geq 0\}$

$[\leq 0] \mapsto \{n \in \mathbb{V} \mid n \leq 0\}$

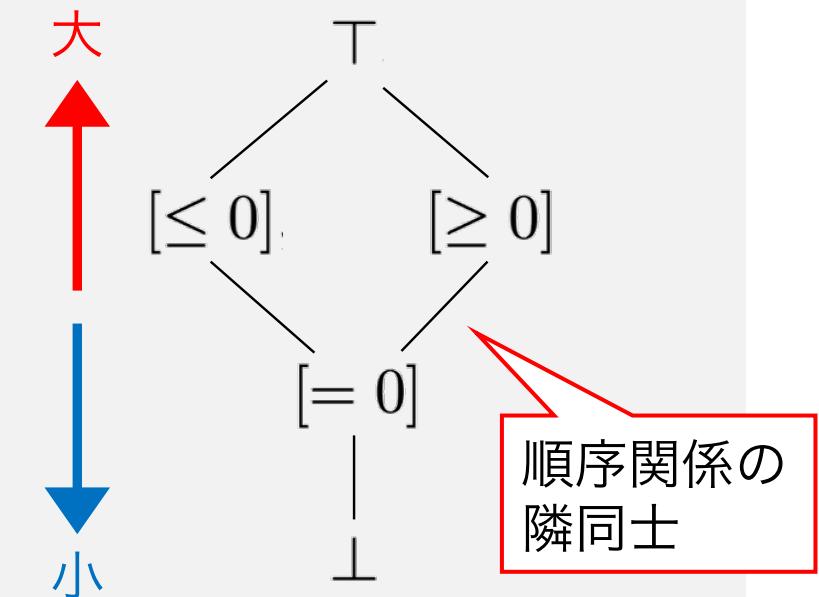
$[= 0] \mapsto \{0\}$

$\top \mapsto \mathbb{V}$

$\perp \mapsto \emptyset$

抽象化関数 $\alpha_{\mathcal{S}}$: 値の集合を過大近似したときの最小の要素を返す

ハッセ図 (Hasse diagram)



Example (Signs)

符号抽象の方法は複数ある

ex.) 要素 $[\geq 0]$, $[\leq 0]$, \top , \perp のみを持つ抽象ドメイン

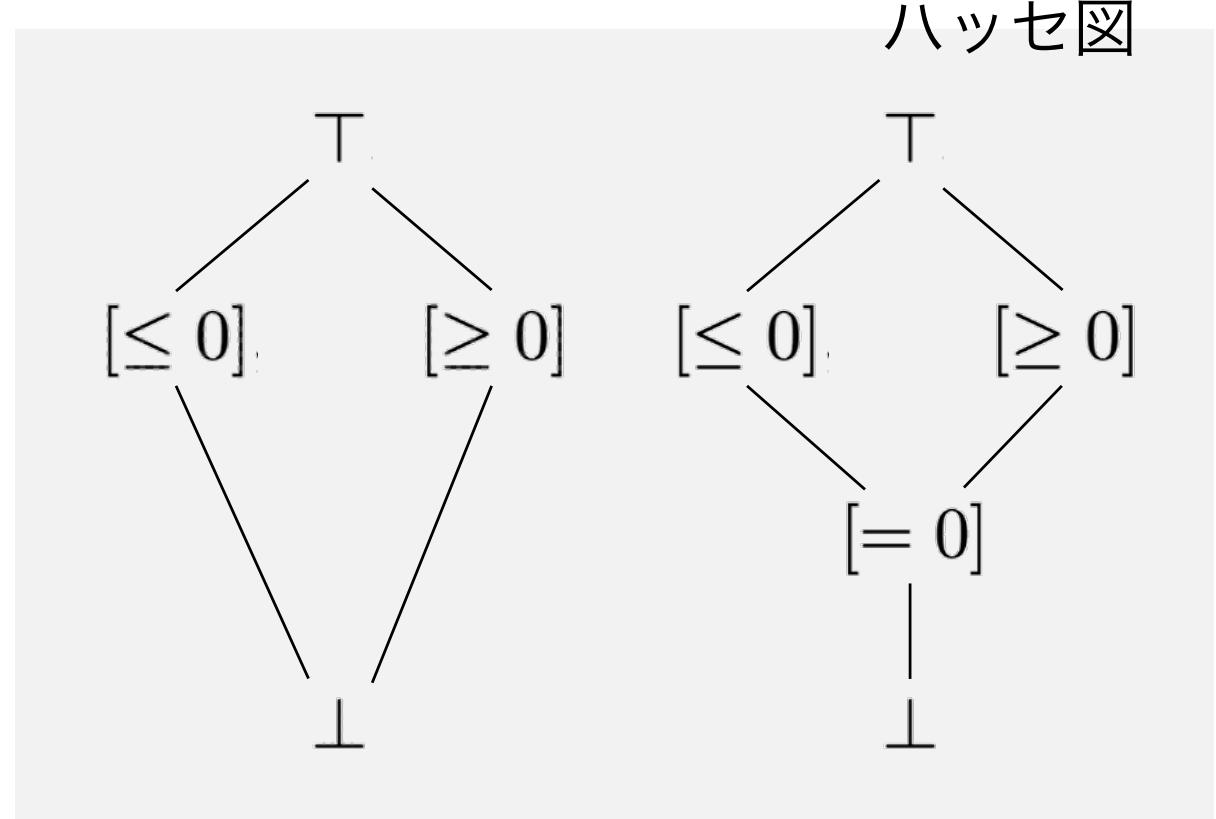
値の集合 $\{0\}$ の抽象化を考える

最小の抽象要素は

$[\geq 0]$? $[\leq 0]$? \rightarrow 比較できない

$\{0\}$ に対する最適な抽象要素が定義できない

\rightarrow 抽象化関数 $\alpha_{\mathcal{S}}$ が存在しない



Example (Intervals)

区間値抽象ドメイン $A_{\mathcal{I}}$ の要素：

- \perp : 値の空集合を表す
- (n_0, n_1) : n_0 は $-\infty$ or 値、 n_1 は $+\infty$ or 値、 $n_0 \leq n_1$
 n_0 から n_1 までの区間を表す

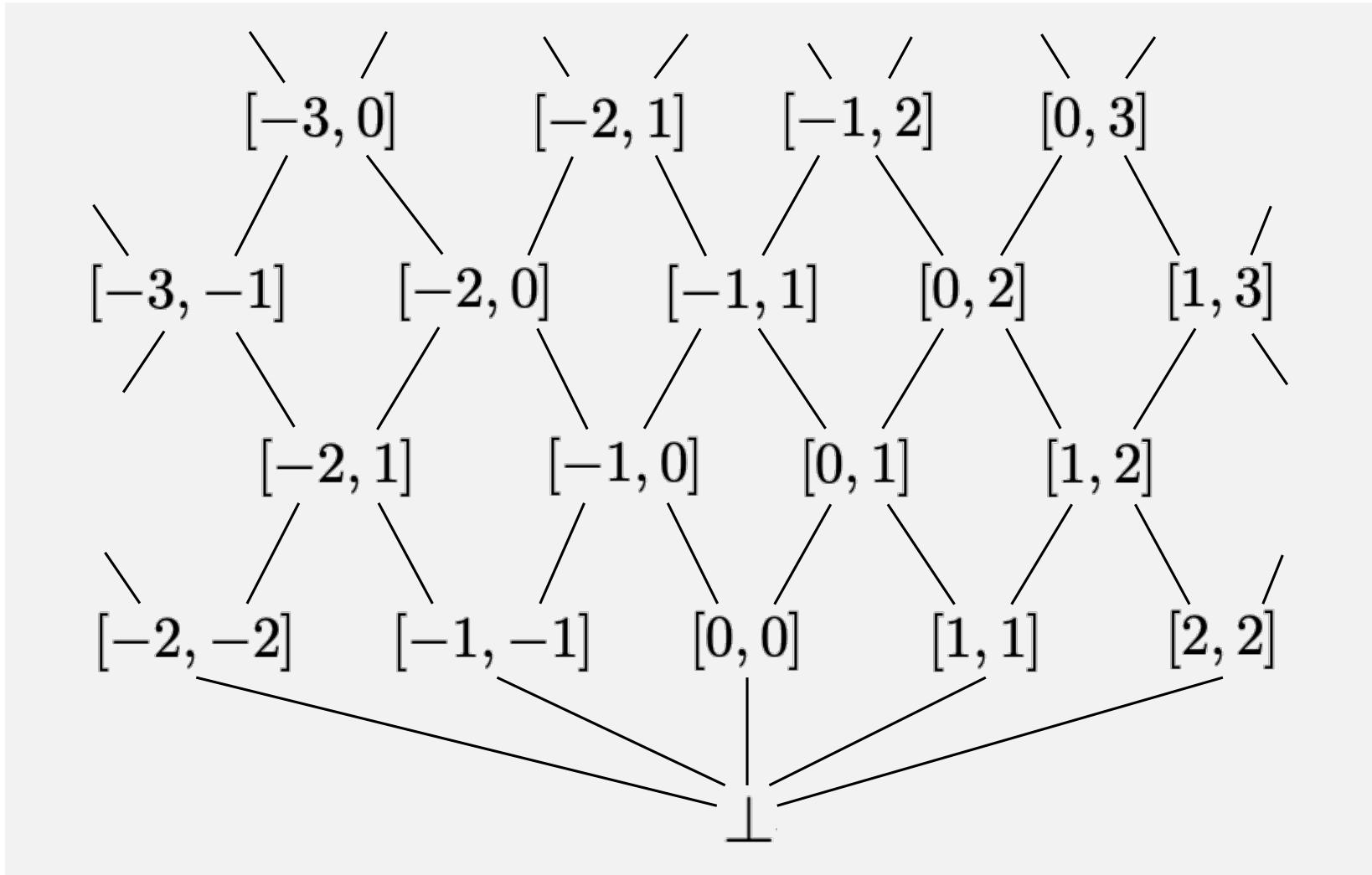
具体化関数 $\gamma_{\mathcal{I}}$:

$$\begin{aligned}\perp &\longmapsto \emptyset \\ [n_0, n_1] &\longmapsto \{n \in \mathbb{V} \mid n_0 \leq n \leq n_1\} \\ [n_0, +\infty) &\longmapsto \{n \in \mathbb{V} \mid n_0 \leq n\} \\ (-\infty, n_1] &\longmapsto \{n \in \mathbb{V} \mid n \leq n_1\} \\ (-\infty, +\infty) &\longmapsto \mathbb{V}\end{aligned}$$

順序関係 $\sqsubseteq_{\mathcal{I}}$ は具体化したときの区間包含の関係

抽象化関数 $\alpha_{\mathcal{I}}$ も定義できる

Example (Intervals)



Example (Congruences)

抽象ドメイン $\mathbb{A}_{\mathcal{C}}$ の要素：

- \perp : 値の空集合を表す
- $(n, p) : p = 0 \text{ or } 0 \leq n < p$
(p で割ったときに余りが n となる値の集合を表す)

具体化関数 $\gamma_{\mathcal{C}} : \perp \longmapsto \emptyset$

$$(n, p) \longmapsto \{n + kp \mid k \in \mathbb{Z}\}$$

$$\text{ex.) } (3, 5) \longmapsto \{\dots, -7, -2, 3, 8, \dots\}$$

順序関係 $\sqsubseteq_{\mathcal{C}} : a_0 \sqsubseteq_{\mathcal{C}} a_1 \iff \gamma_{\mathcal{C}}(a_0) \subseteq \gamma_{\mathcal{C}}(a_1)$

抽象化関数 $\alpha_{\mathcal{C}}$ も定義できる

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

Non-relational Abstraction

値抽象 : $(\mathbb{A}_{\mathcal{V}}, \sqsubseteq_{\mathcal{V}})$

具体化関数 $\gamma_{\mathcal{V}} : \mathbb{A}_{\mathcal{V}} \longrightarrow \wp(\mathbb{V})$ 最小要素 $\perp_{\mathcal{V}}$ 最大要素 $\top_{\mathcal{V}}$

非関係抽象 (non-relational abstraction)

- 抽象要素の集合 $\mathbb{A}_{\mathcal{N}} = \mathbb{X} \longrightarrow \mathbb{A}_{\mathcal{V}}$
- 順序関係 $\sqsubseteq_{\mathcal{N}} : M_0^\sharp \sqsubseteq_{\mathcal{N}} M_1^\sharp \iff \forall x \in \mathbb{X}, M_0^\sharp(x) \sqsubseteq_{\mathcal{V}} M_1^\sharp(x)$
(すべての変数で同じ順序関係が成り立つ)
- 具体化関数 $\gamma_{\mathcal{N}} :$
$$\gamma_{\mathcal{N}} : M^\sharp \longmapsto \{m \in \mathbb{M} \mid \forall x \in \mathbb{X}, m(x) \in \gamma_{\mathcal{V}}(M^\sharp(x))\}$$

Non-relational Abstraction

- 最小要素 $\perp_{\mathcal{N}}$: すべての変数を値抽象ドメインの最小要素にする関数

$$\forall x \in \mathbb{X}, \perp_{\mathcal{N}}(x) = \perp_{\mathcal{V}}$$

- 値抽象で抽象化関数 $\alpha_{\mathcal{V}}$ が存在するとき

$$\alpha_{\mathcal{N}} : M \longmapsto \left((x \in \mathbb{X}) \longmapsto \alpha_{\mathcal{V}}(\{m(x) \mid m \in M\}) \right)$$

- 計算機での表現 : タプルや配列

Example (Non-relational abstraction)

$\mathbb{X} = \{x, y, z\}$ で以下のメモリ状態を考える

$$m_0 : \quad x \mapsto 25 \quad y \mapsto 7 \quad z \mapsto -12$$

$$m_1 : \quad x \mapsto 28 \quad y \mapsto -7 \quad z \mapsto -11$$

$$m_2 : \quad x \mapsto 20 \quad y \mapsto 0 \quad z \mapsto -10$$

$$m_3 : \quad x \mapsto 35 \quad y \mapsto 8 \quad z \mapsto -9$$

このとき $\{m_0, m_1, m_2, m_3\}$ の最適抽象は以下のように定義できる

- ・ 符号抽象化の場合

$$M^\sharp : \quad x \mapsto [\geq 0] \quad y \mapsto \top \quad z \mapsto [\leq 0]$$

- ・ 区間抽象化の場合

$$M^\sharp : \quad x \mapsto [25, 35] \quad y \mapsto [-7, 8] \quad z \mapsto [-12, -9]$$

Contents

3.1 Semantics

- ・プログラミング言語の定義
- ・具体的な意味 (concrete semantics) の定義

3.2 Abstractions

- ・具体ドメイン (concrete domain) と抽象ドメイン (abstract domain) とその関係
- ・具体化関数 (concretization function) と抽象化関数 (abstraction function)
- ・値抽象 (value abstraction)
- ・非関係抽象 (non-relational abstraction)
- ・関係抽象 (relational abstraction)

Relational Abstraction

関係抽象 (relational abstraction) : 複数の変数間で制約がある抽象

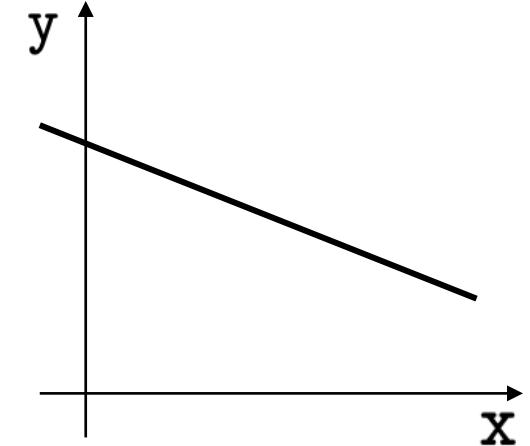
$$\text{ex.) } x + y \leq 3, \quad x^2 - y \leq 0$$

- Linear equalities
- Convex polyhedra
- Octagons

関係抽象ドメインのより良い計算機表現の選択は非関係抽象ドメインよりも難しい

Linear Equalities

- Linear equalities の要素：
- \perp : 空集合を表す
 - 変数間の線形等式

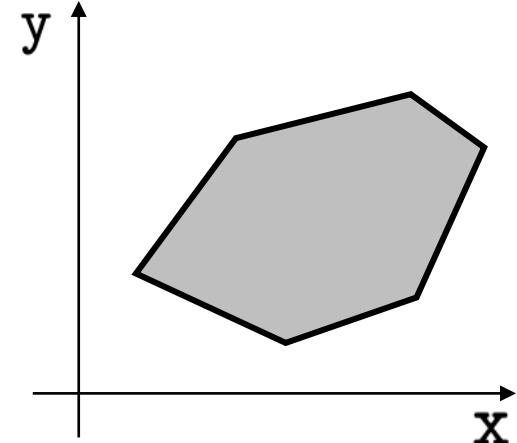


- 幾何学的に、抽象要素は \mathbb{V}^N (N は変数の数) のアフィン空間
アフィン空間：ユークリッド空間から絶対的な原点・座標と長さや角度などの計量の概念を取り除いた空間
- 具体化関数、抽象化関数はともに存在する
- メモリ状態の集合 M の最適抽象は M のすべてのメモリを含んだ最小のアフィン空間

Convex Polyhedra

Convex polyhedra の要素：

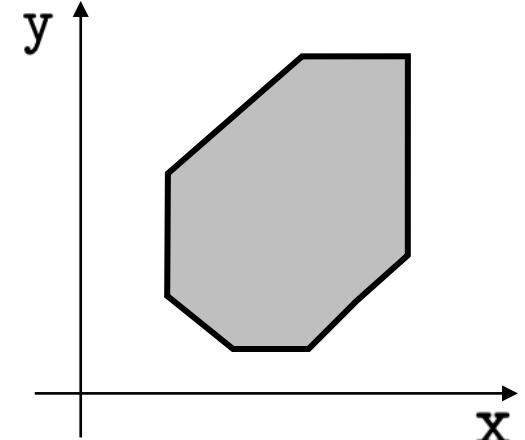
- \perp : 空集合を表す
- 変数間の線形不等式



- 幾何学的に、抽象要素は \mathbb{V}^N (N は変数の数) の凸多面体
- 最適抽象化関数が存在しないことがある ex.) 円板の抽象化
- コストが大きい
 - 不等式の数に制限がない
 - 変数の数を増やすと指数的にコストが大きくなる
- 不等式の制約を少なくすれば良い
 - 静的解析の精度は落ちるがコストは低下する

Octagons

Convex polyhedra のコストを減らしたい



Octagons の要素：

- \perp : 空集合を表す

- 2つ以下の変数間の線形不等式 (係数は $-1, 0, 1$)

$$\pm x \pm y \leq c \quad \text{or} \quad \pm x \leq c$$

- 幾何学的に、抽象要素は “octagonal” shapes に一致する
2次元では最大で8角形になる
- 具体化関数、抽象化関数はともに存在する

Summary

- ・プログラミング言語とその具体的な意味を定義した
- ・抽象の表し方, 具体と抽象の関係について定義した