Static Program Analysis

Lecture 7: Dataflow Analysis VI (Undecidability of MOP Solution & Non-ACC Domains)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

http://moves.rwth-aachen.de/teaching/ws-1415/spa/

Winter Semester 2014/15

Outline

The MOP Solution

Definition (MOP solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $Lab = \{l_1, \ldots, l_n\}$. The MOP solution for S is determined by

$$\mathrm{mop}(S) := (\mathrm{mop}(l_1), \ldots, \mathrm{mop}(l_n)) \in D^n$$

where, for every $l \in Lab$,

$$\mathrm{mop}(l) := \bigsqcup \{\varphi_\pi(\iota) \mid \pi \in Path(l)\}.$$

Remark:

- $Path(l)$ is generally infinite

$\Longrightarrow$ not clear how to compute $\mathrm{mop}(l)$

- In fact: MOP solution generally undecidable (later)

MOP vs. Fixpoint Solution I

Example (Constant Propagation)

$c :=$ `if [z > 0]`^{1} `then`
`[x := 2;]`^{2}
`[y := 3;]`^{3}
`else`
`[x := 3;]`^{4}
`[y := 2;]`^{5}
`[z := x+y;]`^{6}
$[\ldots]^7$

Transfer functions

(for $\delta = (\delta(\mathtt{x}), \delta(\mathtt{y}), \delta(\mathtt{z})) \in D$):

$$\begin{aligned}
\varphi_1(a,b,c) &= (a,b,c)\\
\varphi_2(a,b,c) &= (2,b,c)\\
\varphi_3(a,b,c) &= (a,3,c)\\
\varphi_4(a,b,c) &= (3,b,c)\\
\varphi_5(a,b,c) &= (a,2,c)\\
\varphi_6(a,b,c) &= (a,b,a+b)
\end{aligned}$$

1. Fixpoint solution:

$$\begin{aligned}
\mathsf{CP}_1 &= \iota && = (\top,\top,\top)\\
\mathsf{CP}_2 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_3 &= \varphi_2(\mathsf{CP}_2) && = (2,\top,\top)\\
\mathsf{CP}_4 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_5 &= \varphi_4(\mathsf{CP}_4) && = (3,\top,\top)\\
\mathsf{CP}_6 &= \varphi_3(\mathsf{CP}_3) \sqcup \varphi_5(\mathsf{CP}_5) &&\\
&= (2,3,\top) \sqcup (3,2,\top) && = (\top,\top,\top)\\
\mathsf{CP}_7 &= \varphi_6(\mathsf{CP}_6) && = (\top,\top,\top)
\end{aligned}$$

2. MOP solution:

$$\begin{aligned}
\mathsf{mop}(7) &= \varphi_{[1,2,3,6]}(\top,\top,\top) \sqcup\\
&\quad\ \varphi_{[1,4,5,6]}(\top,\top,\top)\\
&= (2,3,5) \sqcup (3,2,5)\\
&= (\top,\top,5)
\end{aligned}$$

MOP vs. Fixpoint Solution II

Theorem (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a dataflow system. Then*

$$\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$$

Reminder: by Definition 4.9,

$$\Phi_S : D^n \to D^n : (d_1, \ldots, d_n) \mapsto (d'_1, \ldots, d'_n)$$

where $Lab = \{1, \ldots, n\}$ and, for each $l \in Lab$,

$$d'_l := \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(d_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

Proof.

on the board □

Remark: as Example 6.2 shows, $\mathsf{mop}(S) \neq \mathsf{fix}(\Phi_S)$ is possible

Outline

Distributivity of Transfer Functions I

A sufficient condition for the coincidence of MOP and Fixpoint Solution is the distributivity of the transfer functions.

Definition 7.1 (Distributivity)

- Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be complete lattices, and let $F : D \to D'$. F is called distributive (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every $d_1, d_2 \in D$,

$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$

- A dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ is called distributive if every $\varphi_l : D \to D$ $(l \in Lab)$ is so.

Distributivity of Transfer Functions II

Example 7.2

1. The Available Expressions dataflow system is distributive:

$$\begin{aligned}
\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l) \\
&= ((A_1 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \cap \\
&\quad\; ((A_2 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \\
&= \varphi_l(A_1) \sqcup \varphi_l(A_2)
\end{aligned}$$

2. The Live Variables dataflow system is distributive: similarly
3. The Constant Propagation dataflow system is not distributive (cf. Example 6.2):

$$\begin{aligned}
(\top, \top, \top) &= \varphi_{\texttt{z:=x+y}}((2,3,\top) \sqcup (3,2,\top)) \\
&\neq \varphi_{\texttt{z:=x+y}}((2,3,\top)) \sqcup \varphi_{\texttt{z:=x+y}}((3,2,\top)) \\
&= (\top, \top, 5)
\end{aligned}$$

Coincidence of MOP and Fixpoint Solution

Theorem 7.3 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a distributive dataflow system. Then*

$$\mathsf{mop}(S) = \mathsf{fix}(\Phi_S)$$

Proof.

- $\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$: Theorem 6.3
- $\mathsf{fix}(\Phi_S) \sqsubseteq \mathsf{mop}(S)$: as $\mathsf{fix}(\Phi_S)$ is the *least* fixpoint of Φ_S, it suffices to show that $\Phi_S(\mathsf{mop}(S)) = \mathsf{mop}(S)$ (on the board)

□

Outline

Undecidability of the MOP Solution

Theorem 7.4 (Undecidability of MOP solution)

The MOP solution for Constant Propagation is undecidable.

Proof.

Based on undecidability of Modified Post Correspondence Problem:
Let Γ be some alphabet, $n \in \mathbb{N}$, and $u_1, \ldots, u_n, v_1, \ldots, v_n \in \Gamma^+$.
Do there exist $i_1, \ldots, i_m \in \{1, \ldots, n\}$ with $m \geq 1$ and $i_1 = 1$ such that
$u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m}$?

(on the board) $\square$

Outline

Dataflow Analysis with Non-ACC Domains

- **Reminder:** $(D, \sqsubseteq)$ satisfies ACC if each ascending chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ eventually stabilises, i.e., there exists $n \in \mathbb{N}$ such that $d_n = d_{n+1} = \ldots$
- If height (= maximal chain size - 1) of $(D, \sqsubseteq)$ is m, then fixpoint computation terminates after at most $|Lab| \cdot m$ iterations
- **But:** if $(D, \sqsubseteq)$ has non-stabilising ascending chains
  $\Longrightarrow$ algorithm may not terminate
- **Solution:** use widening operators to enforce termination

Widening Operators

Definition 7.5 (Widening operator)

Let $(D, \sqsubseteq)$ be a complete lattice. A mapping $\nabla : D \times D \to D$ is called widening operator if

- for every $d_1, d_2 \in D$,
$$d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$$
and
- for all ascending chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$, the ascending chain $d_0^\nabla \sqsubseteq d_1^\nabla \sqsubseteq \ldots$ eventually stabilises where
$$d_0^\nabla := d_0 \text{ and } d_{i+1}^\nabla := d_i^\nabla \nabla d_{i+1} \text{ for each } i \in \mathbb{N}$$

Remarks:

- $(d_i^\nabla)_{i \in \mathbb{N}}$ is clearly an ascending chain as
$$d_{i+1}^\nabla = d_i^\nabla \nabla d_{i+1} \sqsupseteq d_i^\nabla \sqcup d_{i+1} \sqsupseteq d_i^\nabla$$
- In contrast to $\sqcup$, ∇ does not have to be commutative, associative, monotonic, nor absorptive ($d \nabla d = d$)
- The requirement $d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$ guarantees soundness of widening

Outline

Example: Interval Analysis

Interval Analysis

The goal of Interval Analysis is to determine, for each (interesting) program point, a safe interval for the values of the (interesting) program variables.

Interval analysis is actually a generalisation of constant propagation ($\approx$ interval analysis with 1-element intervals)

Example 7.6 (Interval Analysis)

```
var a[100]:  int;
...
i := 0;
while i <= 42 do
  if i >= 0 ∧ i < 100 then     ⟸
    a[i] := i;
  i := i + 1;
```

Here: redundant array bounds check can be removed

The Domain of Interval Analysis

- The domain $(Int, \subseteq)$ of intervals over $\mathbb{Z}$ is defined by

  $$Int := \{[z_1, z_2] \mid z_1 \in \mathbb{Z} \cup \{-\infty\}, z_2 \in \mathbb{Z} \cup \{+\infty\}, z_1 \leq z_2\} \cup \{\emptyset\}$$

  where
  - $-\infty \leq z$ and $z \leq +\infty$ (for all $z \in \mathbb{Z}$)
  - $\emptyset \subseteq J$ (for all $J \in Int$)
  - $[y_1, y_2] \subseteq [z_1, z_2]$ iff $y_1 \geq z_1$ and $y_2 \leq z_2$
- $(Int, \subseteq)$ is a complete lattice with (for every $\mathcal{I} \subseteq Int$)

  $$\bigsqcup \mathcal{I} = \begin{cases} \emptyset & \text{if } \mathcal{I} = \emptyset \text{ or } \mathcal{I} = \{\emptyset\} \\ [Z_1, Z_2] & \text{otherwise} \end{cases}$$

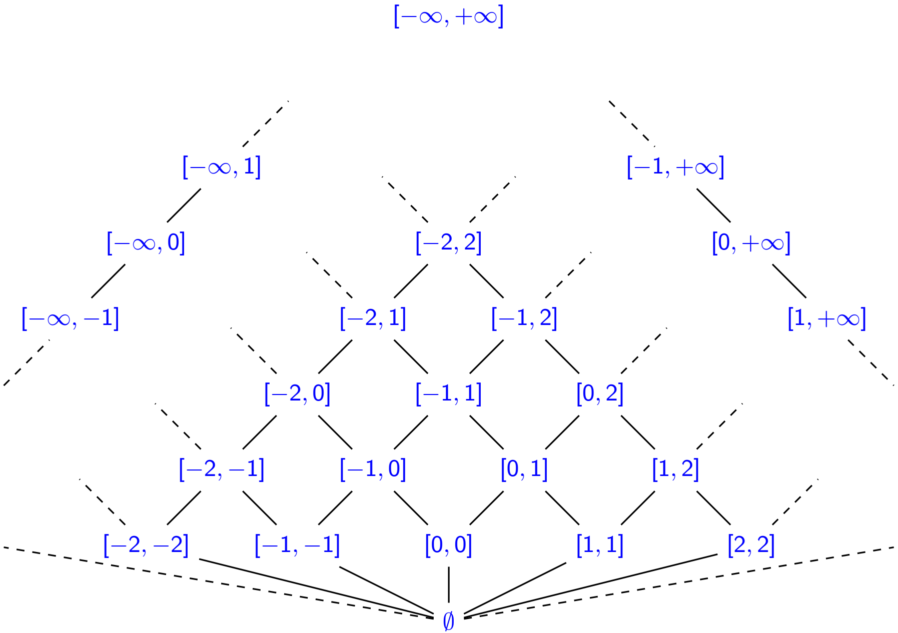
  where

  $$\begin{aligned} Z_1 &:= \textstyle\bigsqcap_{\mathbb{Z} \cup \{-\infty\}} \{z_1 \mid [z_1, z_2] \in \mathcal{I}\} \\ Z_2 &:= \textstyle\bigsqcup_{\mathbb{Z} \cup \{+\infty\}} \{z_2 \mid [z_1, z_2] \in \mathcal{I}\} \end{aligned}$$

  (and thus $\bot = \emptyset$, $\top = [-\infty, +\infty]$)
- Clearly $(Int, \subseteq)$ has infinite ascending chains, such as

  $$\emptyset \subseteq [1, 1] \subseteq [1, 2] \subseteq [1, 3] \subseteq \ldots$$

The Complete Lattice of Interval Analysis



Outline

Formalising Interval Analysis I

The dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ is given by

- set of labels $Lab := Lab_c$
- extremal labels $E := \{\mathrm{init}(c)\}$ (forward problem)
- flow relation $F := \mathrm{flow}(c)$ (forward problem)
- complete lattice $(D, \sqsubseteq)$ where
  - $D := \{\delta \mid \delta : Var_c \to Int\}$
  - $\delta_1 \sqsubseteq \delta_2$ iff $\delta_1(x) \subseteq \delta_2(x)$ for every $x \in Var_c$
- $\iota := \top_D : Var_c \to Int : x \mapsto \top_{Int}$ (with $\top_{Int} = [-\infty, +\infty]$)
- φ: see next slide

Formalising Interval Analysis II

Transfer functions $\{\varphi_l \mid l \in Lab\}$ are defined by

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \texttt{skip} \text{ or } B^l \in BExp \\ \delta[x \mapsto val_\delta(a)] & \text{if } B^l = (x\ \texttt{:=}\ a) \end{cases}$$

where

$$val_\delta(x) := \delta(x) \qquad val_\delta(z) := [z, z]$$

$$val_\delta(a_1 + a_2) := val_\delta(a_1) \oplus val_\delta(a_2)$$
$$val_\delta(a_1 - a_2) := val_\delta(a_1) \ominus val_\delta(a_2)$$
$$val_\delta(a_1 * a_2) := val_\delta(a_1) \odot val_\delta(a_2)$$

with

$$\emptyset \oplus J := J \oplus \emptyset := \emptyset \ominus J := \ldots := \emptyset$$
$$[y_1, y_2] \oplus [z_1, z_2] := [y_1 + z_1, y_2 + z_2]$$
$$[y_1, y_2] \ominus [z_1, z_2] := [y_1 - z_2, y_2 - z_1]$$
$$[y_1, y_2] \odot [z_1, z_2] := \left[\bigsqcap_{y \in [y_1, y_2], z \in [z_1, z_2]} y \cdot z, \bigsqcup_{y \in [y_1, y_2], z \in [z_1, z_2]} y \cdot z\right]$$

Remarks:

- Possible refinement of DFA to take conditional blocks b^l into account
  - essentially: b as edge label, $\varphi_l(\delta)(x) = \delta(x) \setminus \{z \in \mathbb{Z} \mid x = z \implies \neg b\}$ (cf. "DFA with Conditional Branches" later)
- Important: soundness and optimality of abstract operations, e.g., $\oplus$:
  - soundness: $z_1 \in J_1, z_2 \in J_2 \implies z_1 + z_2 \in J_1 \oplus J_2$
  - optimality: $J_1 \oplus J_2$ as small as possible

Outline

Recap: Widening Operators

Definition (Widening operator)

Let $(D, \sqsubseteq)$ be a complete lattice. A mapping $\nabla : D \times D \to D$ is called widening operator if

- for every $d_1, d_2 \in D$,
$$d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$$
and
- for all ascending chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$, the ascending chain $d_0^\nabla \sqsubseteq d_1^\nabla \sqsubseteq \ldots$ eventually stabilises where
$$d_0^\nabla := d_0 \text{ and } d_{i+1}^\nabla := d_i^\nabla \nabla d_{i+1} \text{ for each } i \in \mathbb{N}$$

Remarks:

- $(d_i^\nabla)_{i \in \mathbb{N}}$ is clearly an ascending chain as
$$d_{i+1}^\nabla = d_i^\nabla \nabla d_{i+1} \sqsupseteq d_i^\nabla \sqcup d_{i+1} \sqsupseteq d_i^\nabla$$
- In contrast to $\sqcup$, ∇ does not have to be commutative, associative, monotonic, nor absorptive ($d \nabla d = d$)
- The requirement $d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$ guarantees soundness of widening

Applying Widening to Interval Analysis

- A widening operator: $\nabla : Int \times Int \to Int$ with

$$\begin{aligned}
\emptyset \nabla J &:= J \nabla \emptyset := J \\
[x_1, x_2] \nabla [y_1, y_2] &:= [z_1, z_2] \quad \text{where} \\
z_1 &:= \begin{cases} x_1 & \text{if } x_1 \leq y_1 \\ -\infty & \text{otherwise} \end{cases} \\
z_2 &:= \begin{cases} x_2 & \text{if } x_2 \geq y_2 \\ +\infty & \text{otherwise} \end{cases}
\end{aligned}$$

- Widening turns infinite ascending chain

$$J_0 = \emptyset \subseteq J_1 = [1,1] \subseteq J_2 = [1,2] \subseteq J_3 = [1,3] \subseteq \ldots$$

  into a finite one:

$$\begin{aligned}
J_0^\nabla &= J_0 = \emptyset \\
J_1^\nabla &= J_0^\nabla \nabla J_1 = \emptyset \nabla [1,1] = [1,1] \\
J_2^\nabla &= J_1^\nabla \nabla J_2 = [1,1] \nabla [1,2] = [1,+\infty] \\
J_3^\nabla &= J_2^\nabla \nabla J_3 = [1,+\infty] \nabla [1,3] = [1,+\infty]
\end{aligned}$$

- In fact, the maximal chain size arising with this operator is 4:

$$\emptyset \subseteq [3,7] \subseteq [3,+\infty] \subseteq [-\infty,+\infty]$$

Worklist Algorithm with Widening I

Goal: extend Algorithm 5.3 by widening to ensure termination

Algorithm 7.7 (Worklist algorithm with widening)

Input: *dataflow system* $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

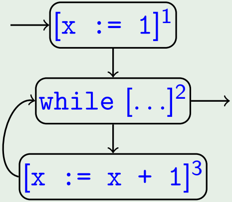
Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialize W*
  for $l \in Lab$ **do** *% Initialise AI*
    if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
  while $W \neq \varepsilon$ **do**
    $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
    if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
      $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \nabla \varphi_l(\mathsf{AI}_l)$;
      for $(l', l'') \in F$ **do**
        if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$, *denoted by* $\mathsf{fix}^\nabla(\Phi_S)$

Remark: due to widening, only $\mathsf{fix}^\nabla(\Phi_S) \sqsupseteq \mathsf{fix}(\Phi_S)$ is guaranteed (cf. Thm. 5.6)

Worklist Algorithm with Widening II

Example 7.8



Transfer functions (for $\delta(\mathtt{x}) = J$):

$$\begin{aligned}
\varphi_1(J) &= [1, 1] \\
\varphi_2(J) &= J \\
\varphi_3(\emptyset) &= \emptyset \\
\varphi_3([x_1, x_2]) &= [x_1 + 1, x_2 + 1]
\end{aligned}$$

Application of worklist algorithm (on the board)

1. without widening: does not terminate
2. with widening: terminates with expected result for AI_2 ($[1, +\infty]$)