More
Data Flow Analyses

Reading: NNH 2.1

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

General Monotonicity Proofs

- We proved RD was monotone for data flow equations for a *specific program*
- Here's a more general proof, for the assignment flow function:
  - To show: If $RD_{entry}(\ell) \subseteq RD_{entry}'(\ell)$ then $RD_{exit}(\ell) \subseteq RD_{exit}'(\ell)$
    - case: $B^\ell = [x := a]^\ell$
  - Assume $RD_{entry}(\ell) \subseteq RD_{entry}'(\ell)$
  - Now $kill_{RD}([x := a]^\ell) = \{ (x, *) \}$ (where * is any label or ?)
  - Thus $RD_{entry}(\ell) \setminus kill_{RD}(B^\ell) \subseteq RD_{entry}'(\ell) \setminus kill_{RD}(B^\ell)$
  - And $gen_{RD}([x := a]^\ell) = \{ (x, \ell) \}$
  - Therefore $(RD_{entry}(\ell) \setminus kill_{RD}(B^\ell)) \cup gen_{RD}(B^\ell) \subseteq (RD_{entry}'(\ell) \setminus kill_{RD}(B^\ell)) \cup gen_{RD}(B^\ell)$
    - And we are done with the case for $[x := a]^\ell$

Live Variables Analysis

A variable is *live* at program point p if there exists a path from p to a use of the variable that does not re-define the variable.

- Live Variables Analysis
  - Determines which variables ***may*** be live at each program point

Live Variable Analysis Example

[y := x]^1;
[z := 1]^2;
while [y>1]^3 do
    [z := z * y]^4;
    [y := y – 1]^5;
[y := 0]^6;

$LV_{enter}(1) =$
$LV_{exit}(1) =$

$LV_{exit}(2) =$
$LV_{exit}(3) =$
$LV_{exit}(4) =$
$LV_{exit}(5) =$
$LV_{exit}(6) =$

Live Variable Analysis Example

[y := x]^1;

[z := 1]^2;

while [y>1]^3 do

    [z := z * y]^4;

    [y := y – 1]^5;

[y := 0]^6;

$LV_{enter}(1) = \{ x \}$

$LV_{exit}(1) = \{ y \}$

$LV_{exit}(2) = \{ y, z \}$

$LV_{exit}(3) = \{ y, z \}$

$LV_{exit}(4) = \{ y, z \}$

$LV_{exit}(5) = \{ y, z \}$

$LV_{exit}(6) = \emptyset$

Live Variable Analysis Equations

[y := x]^1;
[z := 1]^2;
while [y>1]^3 do
    [z := z * y]^4;
    [y := y – 1]^5;
[y := 0]^6;

$LV_{exit}(1) =$
$LV_{exit}(2) =$
$LV_{exit}(3) =$
$LV_{exit}(4) =$
$LV_{exit}(5) =$
$LV_{exit}(6) =$

$LV_{enter}(1) =$
$LV_{enter}(2) =$
$LV_{enter}(3) =$
$LV_{enter}(4) =$
$LV_{enter}(5) =$
$LV_{enter}(6) =$

Live Variable Analysis Equations

[y := x]^1;
[z := 1]^2;
while [y>1]^3 do
    [z := z * y]^4;
    [y := y – 1]^5;
[y := 0]^6;

$LV_{exit}(1) = LV_{enter}(2)$
$LV_{exit}(2) = LV_{enter}(3)$
$LV_{exit}(3) = LV_{enter}(4) \cup LV_{enter}(6)$
$LV_{exit}(4) = LV_{enter}(5)$
$LV_{exit}(5) = LV_{enter}(3)$
$LV_{exit}(6) = \emptyset$

$LV_{enter}(1) = (LV_{exit}(1) \setminus \{ y \}) \cup \{ x \}$
$LV_{enter}(2) = (LV_{exit}(2) \setminus \{ z \}) \cup \emptyset$
$LV_{enter}(3) = (LV_{exit}(3) \setminus \emptyset) \cup \{ y \}$
$LV_{enter}(4) = (LV_{exit}(4) \setminus \{ z \}) \cup \{ y, z \}$
$LV_{enter}(5) = (LV_{exit}(5) \setminus \{ y \}) \cup \{ y \}$
$LV_{enter}(6) = (LV_{exit}(6) \setminus \{ y \}) \cup \emptyset$

General LVA Equations

$LV_{exit}(\ell) = \emptyset$ if $(\ell \in \mathit{final}(S_*))$

$= \cup \{ LV_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_*) \}$ otherwise

$LV_{entry}(\ell) = (LV_{exit}(\ell) \setminus kill_{LV}(B^\ell)) \cup gen_{LV}(B^\ell)$

$kill_{LV}([x := a]^\ell) =$

$kill_{LV}([skip]^\ell) =$

$kill_{LV}([b]^\ell) =$

$gen_{LV}([x := a]^\ell) =$

$gen_{LV}([skip]^\ell) =$

$gen_{LV}([b]^\ell) =$

General LVA Equations

$$LV_{exit}(\ell) = \emptyset \quad \text{if } (\ell \in final(S_*))$$
$$= \cup \{ LV_{entry}(\ell') \mid (\ell, \ell') \in flow^R(S_*) \} \text{ otherwise}$$

$$LV_{entry}(\ell) = (LV_{exit}(\ell) \setminus kill_{LV}(B^\ell)) \cup gen_{LV}(B^\ell)$$

$$kill_{LV}([x := a]^\ell) = \{ x \}$$
$$kill_{LV}([skip]^\ell) = \emptyset$$
$$kill_{LV}([b]^\ell) = \emptyset$$

$$gen_{LV}([x := a]^\ell) = FV(a)$$
$$gen_{LV}([skip]^\ell) = \emptyset$$
$$gen_{LV}([b]^\ell) = FV(b)$$

Data Flow Analysis Characteristics

		Type	
		May	***Must***
Direction	***Forward***	Reaching Definitions	Available Expressions
	Backward	Live Variables	*Very Busy Exp (text)*

Monotone Frameworks

Reading: NNH 2.3, Appendix A.1-A.3

17-654/17-765

Analysis of Software Artifacts

Jonathan Aldrich

Monotone Framework

Reaching Definitions

$RD_{entry}(\ell) = \{(x,?) \mid x \in FV(S_*)\}$ if $\ell = init(S_*)$

$= \cup \{ RD_{exit}(\ell') \mid (\ell', \ell) \in flow(S_*) \}$ otherwise

$RD_{exit}(\ell) = (RD_{entry}(\ell) \setminus kill_{RD}(B^{\ell})) \cup gen_{RD}(B^{\ell})$

Monotone Framework: A Generalization

$Analysis_{\circ}(\ell) = \iota$ if $\ell \in E$

$= \sqcup \{ Analysis_{\bullet}(\ell') \mid (\ell', \ell) \in F \}$ otherwise

$Analysis_{\bullet}(\ell) = f_{\ell}(Analysis_{\circ}(\ell))$

Monotone Framework

$$\text{Analysis}_\circ(\ell) = \iota \quad \text{if } \ell \in E$$
$$= \bigsqcup \{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F \} \quad \text{otherwise}$$

$$\text{Analysis}_\bullet(\ell) = f_\ell(\text{Analysis}_\circ(\ell))$$

where:

- $\circ$ means entry (forward) or exit (backward)
- $\bullet$ means exit (forward) or entry (backward)
- $\sqcup$ is $\bigcup$ (may) or $\bigcap$ (must)
- F is *flow*(S_*) (forward) or *flow*^{R}(S_*) (backward)
- E is { *init*(S_*) } (forward) or *final*(S_*) (backward)
- ι specifies initial or final analysis information, and
- f_ℓ is a transfer function
  - Typically $f_\ell(x) = x \setminus \text{kill}_{\text{Analysis}}(B^\ell) \cup \text{gen}_{\text{Analysis}}(B^\ell)$

Monotone Framework

$$\text{Analysis}_\circ(\ell) = \iota \qquad \text{if } \ell \in E$$
$$= \bigsqcup \{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F \} \qquad \text{otherwise}$$

$$\text{Analysis}_\bullet(\ell) = f_\ell(\text{Analysis}_\circ(\ell))$$

	RD	**AE**	**LV**
$\bigsqcup$			
F			
E			
ι			

Monotone Framework

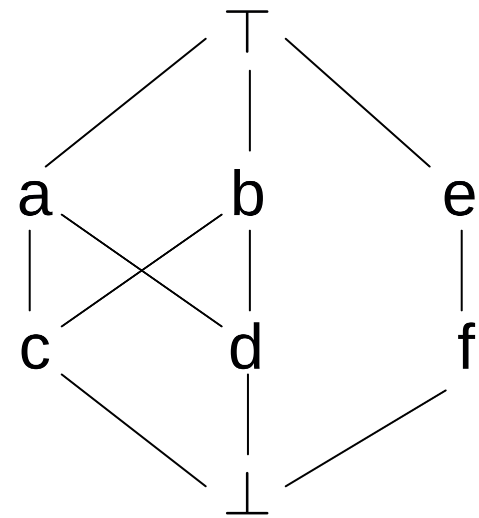
$$\text{Analysis}_\circ(\ell) = \iota \qquad \text{if } \ell \in E$$
$$\qquad\qquad = \bigsqcup \{ \text{Analysis}_\bullet(\ell') \mid (\ell', \ell) \in F \} \qquad \text{otherwise}$$

$$\text{Analysis}_\bullet(\ell) = f_\ell(\text{Analysis}_\circ(\ell))$$

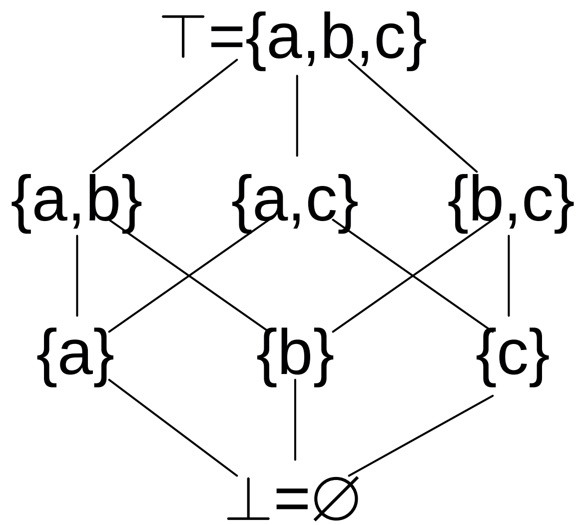
	RD	AE	LV
$\sqcup$	$\cup$	$\cap$	$\cup$
F	$flow(S_*)$	$flow(S_*)$	$flow^R(S_*)$
E	$\{ init(S_*) \}$	$\{ init(S_*) \}$	$final(S_*)$
ι	$\{ (x,?) \mid x \in FV(S_*) \}$	$\emptyset$	$\emptyset$

Complete Lattice

- Not all data flow analyses use sets
  - Lattice: a more general concept
- A set *L* with:
  - A partial order $\sqsubseteq$
  - A combination operator $\sqcup$
  - A least element $\bot = \sqcup(\emptyset)$
  - A greatest element $\top = \sqcup(L)$
  - Each subset *Y* of *L* has a least upper bound $\sqcup(Y)$
- Typically we want the lattice to have finite height
  - A finite number of elements on each path from $\bot$ to $\top$
    - See NNH Appendix A.3

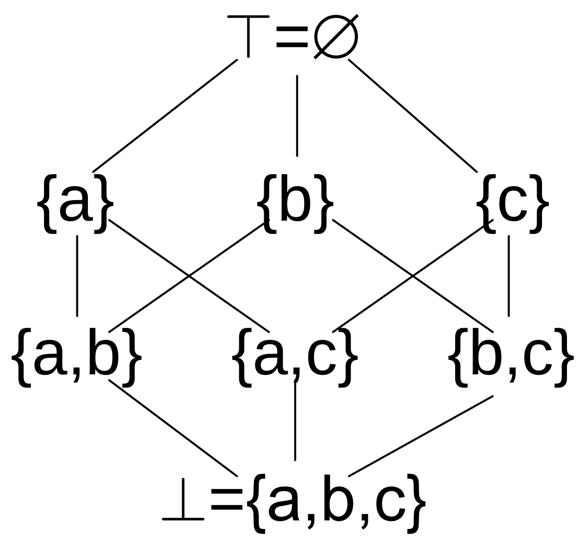


Example: Subset Lattice



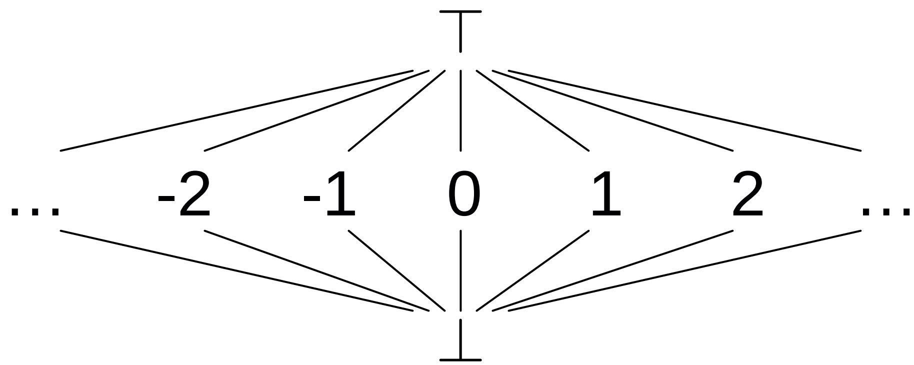
- Reaching Definitions
- The set $L=\mathcal{P}(\{a,b,c\})$ with:
  - $\sqsubseteq$ = $\subseteq$
  - $\sqcup$ = $\cup$ (may analysis)
  - $\bot$ = $\emptyset$ (the most precise and starting element)
  - $\top$ = {a,b,c} (the least precise element)

Example: Superset Lattice



- Available Expressions
- The set $L=\mathcal{P}(\{a,b,c\})$ with:
  - $\sqsubseteq$ = $\supseteq$
  - $\sqcup$ = $\cap$ (must analysis)
  - $\bot$ = {a,b,c} (the most precise and starting element)
  - $\top$ = $\emptyset$ (the least precise element)

Constant Propagation Lattice

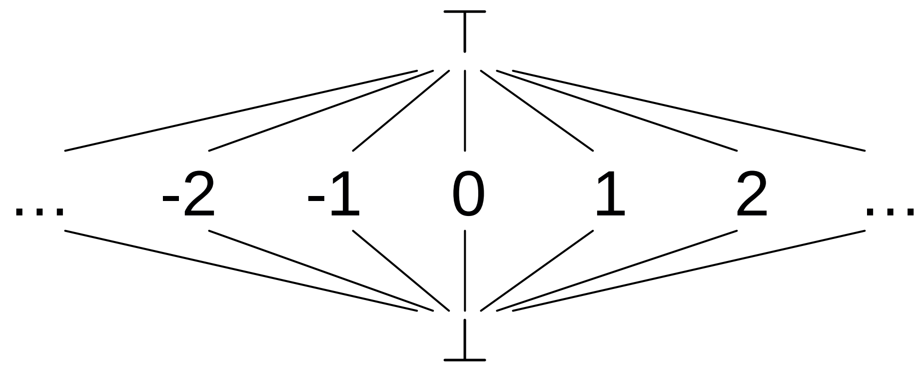


- More efficient than the set of possible values
  - Don't want to store sets
  - If more than one value, give up and assume any ($\top$)
- The set $L = \{\bot, \top\} \cup \text{NAT}$ with:
  - $x \sqsubseteq \top$, $\bot \sqsubseteq x$, $x \sqsubseteq x$
  - $x \sqcup \bot = x$, $x \sqcup \top = \top$, $n \sqcup m = \top$ (for $n \neq m$)
- $\iota = \top$

Tuple Lattices

- Motivation: Constant Propagation
  - Need to hold constants for each variable in the program
- $L_T = L_1 \times L_2 \times L_3 \times \ldots \times L_N$
  - element of tuple lattice is a tuple of elements from each variable's lattice
  - i^{th} component of tuple is info about i^{th} variable/stmt
- $\sqsubseteq_T$ and $\sqcup_T$ are defined pointwise
  - $<\ldots, e_i, \ldots,> \sqsubseteq_T <\ldots, f_i, \ldots,> \equiv \forall i \,.\, e_i \sqsubseteq f_i$
  - $<\ldots, e_i, \ldots,> \sqcup_T <\ldots, f_i, \ldots,> \equiv <\ldots, e_i \sqcup f_i, \ldots,>$
- $\top_T = <\top, \ldots, \top>$
- $\bot_T = <\bot, \ldots, \bot>$
- $\iota_T = <\iota_1, \ldots, \iota_n>$

Constant Propagation Transfer Fns



- $f^{CP}[\![x := a]\!](\sigma) = \sigma\,[x \mapsto CP[\![a]\!](\sigma)]$
- $f^{CP}[\![skip]\!](\sigma) = \sigma$
- $f^{CP}[\![b]\!](\sigma) = \sigma$

- $CP[\![n]\!](\sigma) = n$
- $CP[\![x]\!](\sigma) = \sigma(x)$
- $CP[\![a_1\ op_a\ a_2]\!](\sigma) = CP[\![a_1]\!](\sigma)\ \widehat{op}_a\ CP[\![a_2]\!](\sigma)$

- $z_1\ \widehat{op}_a\ z_2$ $= z_1\ \widehat{op}_a\ z_2$ if $z_1, z_2 \in$ NAT
  $= \top$ if $z_1 = \top$ or $z_2 = \top$
  $= z_1\ (z_2)$ if $z_2\ (z_1) = \bot$

Example

[a := 1]^1

[b := 2]^2

while [a < 2]^3 do

    [b := b * 1]^4;

    [a := a + 1]^5;

Iter	**Position**	**a**	**b**
0	--	$\bot$	$\bot$
1	entry(1)	$\top$	$\top$
2	exit(1)	1	$\top$
3	entry(2)	1	$\top$
4	exit(2)	1	2
5	entry(3)	1	2
6	exit(3)	1	2
7	entry(4)	1	2
8	exit(4)	1	2
9	entry(5)	1	2
10	exit(5)	2	2
11	entry(3)	$\top$	2
12	exit(3)	$\top$	2
13	entry(4)	$\top$	2
14	exit(4)	$\top$	2
15	entry(5)	$\top$	2
17	exit(5)	$\top$	2

Monotonicity Condition

- If $\sigma_1 \sqsubseteq \sigma_2$ then $f_\ell(\sigma_1) \sqsubseteq f_\ell(\sigma_2)$
- Check for $f^{CP}[\![x := a]\!](\sigma)$
  - Assume $\sigma_1 \sqsubseteq \sigma_2$
  - Lemma: $CP[\![a]\!](\sigma_1) \sqsubseteq CP[\![a]\!](\sigma_2)$
    - Proof by induction on the structure of a
    - Base case: $CP[\![n]\!](\sigma_1) = CP[\![n]\!](\sigma_2) = n$
    - Base case: $CP[\![x]\!](\sigma_1) = \sigma_1(x) \sqsubseteq \sigma_2(x) = CP[\![x]\!](\sigma_2)$
    - Inductive case: $CP[\![a_1\ op_a\ a_2]\!](\sigma)$
      - By the induction hypothesis we have:
        - » $CP[\![a_1]\!](\sigma_1) \sqsubseteq CP[\![a_1]\!](\sigma_2)$
        - » $CP[\![a_2]\!](\sigma_1) \sqsubseteq CP[\![a_2]\!](\sigma_2)$
      - By case analysis on the definition of $\widehat{op}_a$ we can prove
        - » $CP[\![a_1]\!](\sigma_1)\ \widehat{op}_a\ CP[\![a_2]\!](\sigma_1) \sqsubseteq CP[\![a_1]\!](\sigma_2)\ \widehat{op}_a\ CP[\![a_2]\!](\sigma_2)$
      - Therefore $CP[\![a_1\ op_a\ a_2]\!](\sigma_1) \sqsubseteq CP[\![a_1\ op_a\ a_2]\!](\sigma_2)$
  - Therefore: $\sigma_1[x \mapsto CP[\![a]\!](\sigma_1)] \sqsubseteq \sigma_2[x \mapsto CP[\![a]\!](\sigma_2)]$
- Must check for other f^{CP} as well