

Analysis Correctness

(continued)

Reading: NNH 2.2 (optional)

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

Set Comprehension Notation

- $\{ (x,y) \mid x \in S_1, y \in S_2, x < y \}$
 - All pairs (x,y)
 - Such that x is in set S_1 and y is in set S_2
 - And $x < y$
- Common notation error on the homework
 - $\{ (x,y) \mid \forall x \in S_1, \forall y \in S_2, x < y \}$
 - Don't need \forall , quantification is implicit in set comprehension

2/8/2005

3

Soundness Example: Sign Analysis

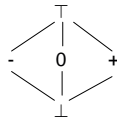
Transfer functions

- σ is input data flow value

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

- $SA(n, \sigma) = \text{sign}(n)$ // returns sign of n
- $SA(x, \sigma) = \sigma(x)$
- $SA(a_1 + a_2, \sigma) = +$ // is this sound?
- $SA(a_1 \text{ op}_a a_2, \sigma) = \top$ for $\text{op}_a \neq +$

Custom Lattice



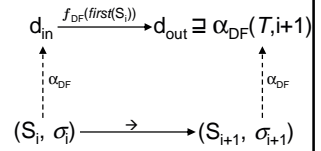
2/8/2005

4

Local Soundness

To show:

- if $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1}) \in T$
- and $d_{in} = \alpha_{DF}(T, i)$
- and $d_{out} = f_{DF}(\text{first}(S_i), d_{in})$
- then $\alpha_{DF}(T, i+1) \sqsubseteq d_{out}$

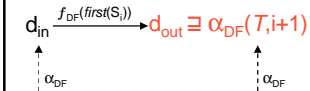


Intuitively, translating from concrete to abstract and applying the flow function will safely approximate \sqsubseteq taking a step in the trace and translating from concrete to abstract

2/8/2005

5

Local Soundness Fails



Program:

$[z := y-7]^1$
 $[x := y+z]^2$

- $\sigma_1 = (x=0, y=3, z=-4)$
- $\sigma_2 = (x=-1, y=3, z=-4)$
- $d_{in} = \alpha_{DF}(T, 1) = (x=0, y=+, z=-)$
- $\alpha_{DF}(T, 2) = (x=-, y=+, z=-)$
- $d_{out} = f^{SA}([x := y+z]^2, d_{in})$
 $= d_{in} [x \mapsto SA(y+z, d_{in})]$
 $= d_{in} [x \mapsto +]$
 $= (x=+, y=+, z=-)$

• $(x=+, y=+, z=-) \not\sqsubseteq (x=-, y=+, z=-)$

2/8/2005

14

The Proof Fails Too (it better!)

To show:

- if $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1}) \in T$
- and $d_{in} = \alpha_{CP}(T, i)$
- and $d_{out} = f_{CP}(\text{first}(S_i), d_{in})$
- then $\alpha_{CP}(T, i+1) \sqsubseteq d_{out}$

- Case: $S_i = [x := a]^i$
 - $\sigma_{i+1} = \sigma_i [x \mapsto \mathcal{A}(a, \sigma_i)]$
 - $d_{in} = \alpha_{CP}(T, i) = \text{sign}(\sigma_i)$
 - $d_{out} = f_{CP}([x := a]^i, d_{in})$
 $= d_{in} [x \mapsto SA(a, d_{in})]$
 - $\alpha_{CP}(T, i+1) = \text{sign}(\sigma_{i+1})$
 $= \text{sign}(\sigma_i [x \mapsto \mathcal{A}(a, \sigma_i)])$
 - Lemma: $\forall a, \sigma, \text{sign}(\mathcal{A}(a, \sigma))$
 $\sqsubseteq SA(a, \text{sign}(\sigma))$
 - Thus $\text{sign}(\sigma_i [x \mapsto \mathcal{A}(a, \sigma_i)])$
 $\sqsubseteq \text{sign}(\sigma_i) [x \mapsto SA(a, \text{sign}(\sigma_i))]$

2/8/2005

15

The Proof Fails Too (*it better!*)

To show:

if $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1}) \in T$
and $d_{in} = \alpha_{CP}(T, i)$
and $d_{out} = f_{CP}(first(S_i), d_{in})$
then $\alpha_{CP}(T, i+1) \sqsubseteq d_{out}$

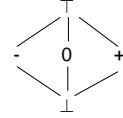
- Lemma: $\forall a, \sigma_i \text{ sign}(\mathcal{A}(a, \sigma_i)) \sqsubseteq SA(a, \text{sign}(\sigma_i))$
 - Proof by induction on structure of a
 - Case: $a = n$
 - $\text{sign}(\mathcal{A}(n, \sigma_i)) = \text{sign}(n)$
 - $SA(n, \text{sign}(\sigma_i)) = \text{sign}(n)$
 - $\text{sign}(n) \sqsubseteq \text{sign}(n)$ OK!
 - Case: $a = a_1 + a_2$
 - Assume ind. hyp. for a_1 and a_2
 - Therefore $\text{sign}(\mathcal{A}(a_k, \sigma_i)) = \text{sign}(a_k)$, $k \in 1, 2$
 - Subcase: $SA(a_k, \text{sign}(\sigma_i)) = -$
Then $\text{sign}(\mathcal{A}(a_k, \sigma_i)) = -$
Now $SA(a, \text{sign}(\sigma_i)) = +$
But clearly $\text{sign}(\mathcal{A}(a, \sigma_i)) = -$
So $\text{sign}(\mathcal{A}(a, \sigma_i)) \not\sqsubseteq SA(a, \text{sign}(\sigma_i))$
Fails!

2/8/2005

18

Corrected Sign Analysis

Custom Lattice



- Transfer functions

- σ is input data flow value

- $f^{SA}([x := a], \sigma) = \sigma[x \mapsto SA(a, \sigma)]$

- $f^{SA}([\text{skip}], \sigma) = \sigma$

- $f^{SA}([b], \sigma) = \sigma$

- $SA(n, \sigma) = \text{sign}(n)$ // returns sign of n

- $SA(x, \sigma) = \sigma(x)$

- $SA(a_1 + a_2, \sigma) = +$ when $SA(a_1, \sigma) = SA(a_2, \sigma) = +$

- $SA(a_1 + a_2, \sigma) = -$ when $SA(a_1, \sigma) = SA(a_2, \sigma) = -$

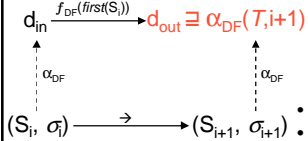
2/8/2005

$SA(a_1 \text{ op}_a a_2, \sigma) = \top$

otherwise

19

Local Soundness Succeeds



Program:

$[z := y-7]^!$
 $[x := y+z]^?$

- $\sigma_1 = (x=0, y=3, z=-4)$
- $\sigma_2 = (x=-1, y=3, z=-4)$
- $d_{in} = \alpha_{DF}(T, 1) = (x=0, y=+, z=-)$
- $\alpha_{DF}(T, 2) = (x=-, y=+, z=-)$
- $d_{out} = f^{SA}([x := y+z]^?, d_{in})$
= $d_{in}[x \mapsto SA(y+z, d_{in})]$
= $d_{in}[x \mapsto \top]$
= $(x=\top, y=+, z=-)$
- $(x=\top, y=+, z=-) \sqsupseteq (x=-, y=+, z=-)$

Trace T:

t	pp	x	y	z
0	1	0	3	0
1	2	0	3	-4
2	-	-1	3	-4

2/8/2005

28

The Proof Succeeds Now

To show:

if $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1}) \in T$
and $d_{in} = \alpha_{CP}(T, i)$
and $d_{out} = f_{CP}(first(S_i), d_{in})$
then $\alpha_{CP}(T, i+1) \sqsubseteq d_{out}$

- Case: $S_i = [x := a]^f$
 - $\sigma_{i+1} = \sigma_i[x \mapsto \mathcal{A}(a, \sigma_i)]$
 - $d_{in} = \alpha_{CP}(T, i) = \text{sign}(\sigma_i)$
 - $d_{out} = f_{CP}([x := a]^f, d_{in})$
= $d_{in}[x \mapsto SA(a, d_{in})]$
 - $\alpha_{CP}(T, i+1) = \text{sign}(\sigma_{i+1})$
= $\text{sign}(\sigma_i[x \mapsto \mathcal{A}(a, \sigma_i)])$
 - Lemma: $\forall a, \sigma \text{ sign}(\mathcal{A}(a, \sigma)) \sqsubseteq SA(a, \text{sign}(\sigma))$
 - Thus $\text{sign}(\sigma_i[x \mapsto \mathcal{A}(a, \sigma_i)]) \sqsubseteq \text{sign}(\sigma_i[x \mapsto SA(a, \text{sign}(\sigma_i))])$

2/8/2005

29

The Proof Succeeds Now

To show:

if $(S_i, \sigma_i) \rightarrow (S_{i+1}, \sigma_{i+1}) \in T$
and $d_{in} = \alpha_{CP}(T, i)$
and $d_{out} = f_{CP}(first(S_i), d_{in})$
then $\alpha_{CP}(T, i+1) \sqsubseteq d_{out}$

- Lemma: $\forall a, \sigma_i \text{ sign}(\mathcal{A}(a, \sigma_i)) \sqsubseteq SA(a, \text{sign}(\sigma_i))$
 - Proof by induction on structure of a
 - Case: $a = n$
 - $\text{sign}(\mathcal{A}(n, \sigma_i)) = \text{sign}(n)$
 - $SA(n, \text{sign}(\sigma_i)) = \text{sign}(n)$
 - $\text{sign}(n) \sqsubseteq \text{sign}(n)$ OK!
 - Case: $a = a_1 + a_2$
 - Assume ind. hyp. for a_1 and a_2
 - Therefore $\text{sign}(\mathcal{A}(a_k, \sigma_i)) = \text{sign}(a_k)$, $k \in 1, 2$
 - Subcase: $SA(a_k, \text{sign}(\sigma_i)) = -$
Then $\text{sign}(\mathcal{A}(a_k, \sigma_i)) = -$
Now $SA(a, \text{sign}(\sigma_i)) = -$
And clearly $\text{sign}(\mathcal{A}(a, \sigma_i)) = -$
So $\text{sign}(\mathcal{A}(a, \sigma_i)) \sqsubseteq SA(a, \text{sign}(\sigma_i))$
OK!
- Other cases are similar...

2/8/2005

32

What does Correctness Mean?

- Intuition

- At a fixed point, analysis results are a *conservative abstraction* of program execution

- Soundness condition

- When data flow analysis reaches a **fixed point** F , then for all traces T and all times t in each trace, $\alpha(T, t) \sqsubseteq F(pp(T, t))$

2/8/2005

33

Global Soundness

- Theorem (Global Soundness)
 - Assume that $\forall T \in \text{traces}(S) \alpha_{DF}(T, 0) \sqsubseteq \iota$ and that analysis DF is monotone and locally sound with respect to α_{DF}
 - Then for any fixed point DF_{fix} of DF on program S, $\forall T \in \text{traces}(S) \forall t \in \text{times}(T)$ we have $\alpha_{DF}(T, t) \sqsubseteq DF_{fix}(pp(T, t))$
- Proof: For each trace T we do induction on t
 - Induction hypothesis: $\alpha_{DF}(T, t) \sqsubseteq DF_{fix}(pp(T, t))$
 - Base case: t=0
 - By assumption $\alpha_{DF}(T, 0) \sqsubseteq \iota = DF_{fix}(pp(T, 0))$
 - Inductive case: time t and statement S_i
 - Simplifying assumption: straight-line control flow
 - By induction hypothesis we have $\alpha_{DF}(T, t-1) \sqsubseteq DF_{fix}(pp(T, t-1))$
 - By monotonicity of DF we have: $f_{DF}(S_i, \alpha_{DF}(T, t-1)) \sqsubseteq f_{DF}(S_i, DF_{fix}(pp(T, t-1)))$
 - By local soundness we have $\alpha_{DF}(T, t) \sqsubseteq f_{DF}(S_i, \alpha_{DF}(T, t-1))$
 - By transitivity we get $\alpha_{DF}(T, t) \sqsubseteq f_{DF}(S_i, DF_{fix}(pp(T, t-1)))$
 - But $f_{DF}(S_i, DF_{fix}(pp(T, t-1))) = DF_{fix}(pp(T, t))$ because it's a fixed point
 - So we have $\alpha_{DF}(T, t) \sqsubseteq DF_{fix}(pp(T, t))$

2/8/2005

43

Abstraction for Reaching Definitions

- $\alpha_{RD}(\langle p_1, m_1 \rangle \dots \langle p_n, m_n \rangle, t) =$
 $\{ (x, p_k) \mid x \in FV(S) \}$
 and $stm(p_k) = [x := a]$
 and $k < t$
 and $\forall j, k < j < t \text{ } stm(p_j) \neq [x := a']$
- Intuition
 - x was assigned at time k
 - k is before the current time t
 - x was not assigned at any time j between k and t

2/8/2005

46

Local Soundness for Reaching Definitions

- To show:
- Case: $S_i = [x := a]^t$
 - $d_{in} = \alpha_{RD}(T, i)$
 - $d_{out} = f_{RD}([x := a]^t, d_{in})$
 $= (\alpha_{RD}(T, i) \setminus \{(x, *)\}) \cup \{(x, t)\}$
 - Lemma: $\alpha_{RD}(T, i+1) =$
 $(\alpha_{RD}(T, i) \setminus \{(x, *)\}) \cup \{(x, t)\}$
 - So $\alpha_{RD}(T, i+1) = d_{out}$
 - Thus $\alpha_{RD}(T, i+1) \sqsubseteq d_{out}$

2/8/2005

47

Local Soundness for Reaching Definitions

- Lemma: if $S_i = [x := a]^t$ then
 $\alpha_{RD}(T, i+1) = (\alpha_{RD}(T, i) \setminus \{(x, *)\}) \cup \{(x, t)\}$
 - $\alpha_{RD}(T, i) = \{ (y, p_k) \mid y \in FV(S) \}$
 and $stm(p_k) = [y := a']$
 and $k < i$
 and $\forall j, k < j < i \text{ } stm(p_j) \neq [y := a'']$
 - $\alpha_{RD}(T, i+1) = \{ (y, p_k) \mid y \in FV(S) \}$
 and $stm(p_k) = [y := a']$
 and $k < i+1$
 and $\forall j, k < j < i+1 \text{ } stm(p_j) \neq [y := a'']$
 - $(x, t) \in \alpha_{RD}(T, i+1)$ by formula with $k = i, y = x$
 - $(x, t') \notin \alpha_{RD}(T, i+1)$ for $t' \neq t$ by formula with $j = i, y = x$
 - Other than substitutions of $k = i$ and $j = i$, the formulas are the same

2/8/2005

52

Abstraction for Live Variables

- $\alpha_{LV}(\langle p_1, m_1 \rangle \dots \langle p_n, m_n \rangle, t) =$
 $\{ x \mid x \in FV(stm(p_k)) \text{ where } k > t \}$
 and $\forall j, t < j < k \text{ } stm(p_j) \neq [x := a']$
- Intuition
 - x is used at time k
 - k comes after the current time t
 - No statement j executing between k and t assigns x

2/8/2005

55

Local Soundness for Live Variables

- To show:
- Case: $S_{i+1} = [x := a]^t$
 - $d_{in} = \alpha_{LV}(T, i+1)$
 - $d_{out} = f_{LV}([x := a]^t, d_{in})$
 $= (\alpha_{LV}(T, i+1) \setminus \{x\}) \cup FV(a)$
 - Lemma: $\alpha_{LV}(T, i) =$
 $(\alpha_{LV}(T, i+1) \setminus \{x\}) \cup FV(a)$
 - So $\alpha_{LV}(T, i) = d_{out}$
 - Thus $\alpha_{LV}(T, i) \sqsubseteq d_{out}$
- Note: i and i+1 are swapped due to reverse analysis

2/8/2005

56

Local Soundness for Live Variables

- Lemma: if $S_{i+1} = [x := a]^{\ell}$ then
 - $\alpha_{LV}(T, i) = (\alpha_{LV}(T, i+1) \setminus \{x\}) \cup FV(a)$
 - $\alpha_{LV}(\langle p_1, m_1 \rangle, \dots, \langle p_n, m_n \rangle, i) =$
 $\{y \mid y \in FV(stm(p_k)) \text{ where } k > i$
 $\text{and } \forall j, i < j < k \text{ } stm(p_j) \neq [y := a]^{\ell}\}$
 - $\alpha_{LV}(\langle p_1, m_1 \rangle, \dots, \langle p_n, m_n \rangle, i+1) =$
 $\{y \mid y \in FV(stm(p_k)) \text{ where } k > i+1$
 $\text{and } \forall j, i+1 < j < k \text{ } stm(p_j) \neq [y := a]^{\ell}\}$
 - $FV(a) \in \alpha_{RD}(T, i)$ by formula with $k = i+1, y = x$
 - $x \notin \alpha_{RD}(T, i)$ if $x \notin FV(a)$ by formula with $j = i+1, y = x$
 - Other than substitutions of $k = i+1$ and $j = i+1$, the formulas are the same

2/8/2005

61

Iterative Worklist Algorithm

Reading: NNH 2.4

17-654/17-765

Analysis of Software Artifacts

Jonathan Aldrich

Worklist Algorithm

```
worklist = new Stack();
forall l in labels(S) do
  analysis[l] = ⊥;
forall l in E do
  analysis[l] = ⊥;
  worklist.addAll({(l, l2) | (l, l2) ∈ F});

while (!worklist.isEmpty()) do
  (l1, l2) = worklist.pop();
  if f_{l1}(Analysis[l1]) ≠ Analysis[l2] then
    Analysis[l2] = Analysis[l1] ∪ f_{l1}(Analysis[l1])
    forall l3 where (l2, l3) ∈ F do
      worklist.add((l2, l3));
```

2/8/2005

67

Example of Worklist

	Iter	Edge	Worklist	a _o	b _o
[a := 1] ¹	0	-	1→2	T	T
[b := 2] ²	1	1→2	2→3	1	T
while [a < 2] ³ do	2	2→3	3→4, 3→6	1	2
	3	3→4	4→5, 3→6	1	2
[b := b * 1] ⁴ ;	4	4→5	5→3, 3→6	1	2
	5	5→3	3→4, 3→6	1	2
[a := a + 1] ⁵ ;	6	3→4	4→5, 3→6	T	2
	7	4→5	5→3, 3→6	T	2
[a := b + 1] ⁶ ;	8	5→3	3→6	T	2
	9	3→6	-	T	2

2/8/2005

68

Worklist: Properties

- Correctness
 - Implements chaotic iteration, therefore correct
- Performance
 - Visits each node only when data changes
 - up to $h = \text{height of lattice}$
 - Propagates data along control flow edges
 - up to $e = \text{max outbound edges per node}$
 - Assume lattice operation cost is o
 - Overall, $O(h * e * o)$

2/8/2005

69

Worklist Algorithm

```
worklist = new Stack();
forall l in labels(S) do
  analysis[l] = ⊥;
forall l in E do
  analysis[l] = ⊥;
  worklist.addAll({(l, l2) | (l, l2) ∈ F});

while (!worklist.isEmpty()) do
  (l1, l2) = worklist.pop();
  if f_{l1}(Analysis[l1]) ≠ Analysis[l2] then
    Analysis[l2] = Analysis[l1] ∪ f_{l1}(Analysis[l1])
    forall l3 where (l2, l3) ∈ F do
      worklist.add((l2, l3));
```

h: height of lattice and max times any node can change
n: number of nodes
e: number of edges
o: cost of data flow operations

*may execute O(h*e) times cost O(h*e*o)*
*may execute O(h*n) times cost O(h*n*o)*
*may execute O(h*e) times cost O(h*e)*

2/8/2005

70

Performance

	h	o	$O(h * e * o)$
Reaching Definitions	n	n	$n^2 * e$
Live Variables	v	v	$v^2 * e$
Constant Propagation (sets)	∞	∞	∞ (May not terminate)
Constant Propagation (lattice)	2^*v	2^*v	$v^2 * e$
Sign Analysis	2^*v	2^*v	$v^2 * e$

2/8/2005

72

Nonterminating Analysis

(Moral: make your lattices finite height!)

```
[x := 0]1
while [x < y]2 do
  [x := x + 1]3;
[x := 0]4;
```

iter	Position	x	y
0	--	\emptyset	\emptyset
1	entry(1)	Z	Z
2	exit(1)	{0}	Z
3	entry(2)	{0}	Z
4	exit(2)	{0}	Z
5	entry(3)	{0}	Z
6	exit(3)	{1}	Z
7	entry(2)	{0,1}	Z
8	exit(2)	{0,1}	Z
9	entry(3)	{0,1}	Z
10	exit(3)	{1,2}	Z
11	entry(2)	{0,1,2}	Z
12	exit(2)	{0,1,2}	Z
13	entry(3)	{0,1,2}	Z
14	exit(3)	{1,2,3}	Z
15	entry(2)	{0,1,2,4}	Z
...			

2/8/2005

73