CS 243 Review Session 1
Creating a Dataflow Analysis

Winter 2024 · Zhenbang You

Dataflow analysis

A **common skeleton** for a wide variety of program analysis tasks.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Dataflow analysis

A **common skeleton** for a wide variety of program analysis tasks.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

1) OUT[ENTRY] = v_{ENTRY};
2) **for** (each basic block B other than ENTRY) OUT[B] = $\top$;
3) **while** (changes to any OUT occur)
4) **for** (each basic block B other than ENTRY) {
5) IN[B] = $\bigwedge_{P \text{ a predecessor of } B}$ OUT[P];
6) OUT[B] = f_B(IN[B]);
}

(a) Iterative algorithm for a forward data-flow problem.

Dataflow analysis

A **common skeleton** for a wide variety of program analysis tasks.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

```
1)  OUT[ENTRY] = v_ENTRY;
2)  for (each basic block B other than ENTRY) OUT[B] = ⊤;
3)  while (changes to any OUT occur)
4)      for (each basic block B other than ENTRY) {
5)          IN[B] = ⋀_{P a predecessor of B} OUT[P];
6)          OUT[B] = f_B(IN[B]);
        }
```

(a) Iterative algorithm for a forward data-flow problem.

Topic of today

- How to create a dataflow algorithm
- Step 1: Choose a lattice
- Step 2: Find the transfer function
- Step 3: Determine the initial values

Creating a new dataflow algorithm

Goal: Detect all **possible** use of uninitialized variables.

Simplified instruction set:

- a = v
- a = v + w
- print(v)

a must be a variable.
v and w could be variables or constants.

Roadmap

Two steps:

1. Figure out what variables are defined at each instruction;
2. Warn if a variable used has not been defined yet.

Part 1 is the dataflow analysis → we need to fill out the table.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Step 1: Figure out the lattice

Recall from lecture that a semilattice can be seen in two different ways:

- A **set** equipped with a **partial order ≤**
- A **set** equipped with a **“meet” operator ∧**

These two definitions are interchangeable (i.e., you can derive ≤ from ∧, or the other way around).

> Given ≤, define $a \wedge b$ as the **greatest lower bound** of a and b.
> Given ∧, define $a \leq b$ iff $a \wedge b = a$.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Step 1a: Figure out the lattice domain

We first look at the lattice **set**.

What kind of data do we want to associate with each basic block / instruction?

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Step 1a: Figure out the lattice domain

What kind of data do we want to associate with each basic block / instruction?

Goal: detect when a variable is **undefined**.
So we can keep track of the set of **currently defined variables**.

Lattice elements = set of program variables
Meaning = currently defined variables

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Step 1b: Figure out the ordering / meet operator

The partial order and meet operator are essentially equivalent: figure out one, and you've automatically found the other!

Ordering (≤): what's the "safer" thing to do? Define ≤ such that

$$\text{safe} \leq \text{unsafe}$$

In our case, erring on the side of safety = assume a variable is NOT defined. So ≤ is the **subset relation**.

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Step 1b: Figure out the ordering / meet operator

A different way to look at the problem.

Meet operator (∧): What should we do when two different paths converge?

```
if …:     a = 1;    b = 2
else:     a = 42;   d = 43
# what can we assume here?
```

Assume a variable is defined only if it's defined in **both** branches → define ∧ to be **set intersection**

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 1b: Figure out the ordering / meet operator

Exercise: Check that defining

$\leq$ as $\subseteq$ and $\wedge$ as $\cap$

are equivalent.

> Given $\leq$, define $a \wedge b$ as the **greatest lower bound** of a and b.
>
> Given $\wedge$, define $a \leq b$ iff $a \wedge b = a$.

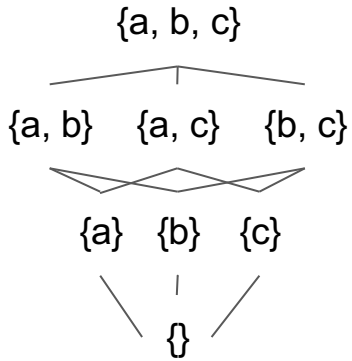
Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Side note: lattice diagrams

Diagrams are helpful in that they unify the two definitions of semilattices.

{a} ≤ {a, b} because there's an edge {a} – {a, b}.

{a, b} ∧{a, c} = {a} because {a} is the **greatest** element that lower-bounds both {a, b} and {a, c}.



Step 2: Figure out the transfer function

Given an instruction / basic block, **how does it change the lattice element?** (We also need to decide the direction.)

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 2: Figure out the transfer function

Given an instruction / basic block, **how does it change the lattice element?** (We also need to decide the direction.)

Here: lattice element = set of defined variables

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 2: Figure out the transfer function

Given an instruction / basic block, **how does it change the lattice element?** (We also need to decide the direction.)

Here: lattice element = set of defined variables

IN[b1] = …

b1: x = …

OUT[b1] = IN[b1] ∪{x}

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 2: Figure out the transfer function

Given an instruction / basic block, **how does it change the lattice element?** (We also need to decide the direction.)

Here: lattice element = set of defined variables

IN[b1] = …

b1: x = …

OUT[b1] = IN[b1] ∪{x}

forward

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 2: Figure out the transfer function

Given an instruction / basic block, **how does it change the lattice element?** (We also need to decide the direction.)

Here: lattice element = set of defined variables

IN[b2] = …

b2: print(v)

OUT[b2] = IN[b2]

Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

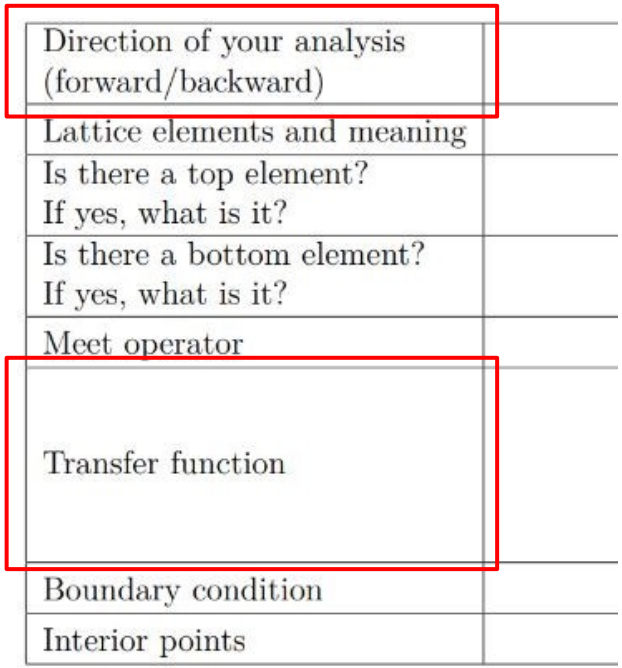
Step 2: Figure out the transfer function

Transfer function:

$$f_b(S) = \begin{cases} S \cup \{x\} & \text{if } b \text{ has instruction } x = \ldots \\ S & \text{otherwise} \end{cases}$$

Alternatively using the Def set used in live variables:

$$f_b(S) = S \cup \text{Def}[b]$$



Direction of your analysis (forward/backward)
Lattice elements and meaning
Is there a top element? If yes, what is it?
Is there a bottom element? If yes, what is it?
Meet operator
Transfer function
Boundary condition
Interior points

Step 3: Figure out initial values

Interior points should typically always be initialized to the top element. (Otherwise the iterative algorithm will never go anywhere.)

Boundary condition (OUT[ENTRY]) depends on our assumption of the surrounding environment (e.g., whether there are any variables defined outside / prior to the given control-flow graph).

Direction of your analysis (forward/backward)	
Lattice elements and meaning	
Is there a top element? If yes, what is it?	
Is there a bottom element? If yes, what is it?	
Meet operator	
Transfer function	
Boundary condition	
Interior points	

Filled-out table

Direction of your analysis (forward/backward)	forward
Lattice elements and meaning	sets of program variables, representing currently defined variables
Is there a top element? If yes, what is it?	yes; universal set
Is there a bottom element? If yes, what is it?	yes; empty set
Meet operator	set intersection
Transfer function	$f_b(S) = S \cup \{x\}$ if b has instruction x = … or S otherwise
Boundary condition	empty set
Interior points	universal set