RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

CS 516 Compilers and Programming Languages II

Approximation 2

Road Ahead

New homework is posted on data flow analysis frameworks / approximation

First homework submission extension until Friday 21, 11:59pm

Will post first papers for in-class discussion soon,

Classification of Approximations

Underlying assumption: There is **a ground truth** consisting of the **perfect/correct/optimal answer**. This answer may not be effectively computable. An **approximate answer** comes "close" to the ground truth, where "closeness" is an application-specific metric.

Model approximation (solving a real-world problem through an algorithm):

- use a simpler view of the world (ground truth) that still works for a particular application
  (example: assume the world is flat for a car-navigation application)
- even optimal solution of the model is approximation to ground truth (e.g.: simplified physical interactions)

Data approximations :

- less precise data (e.g.: single vs. double)
- use a subset of data (e.g.: statistical representation, subset probing)
- probabilistic data values

Computation approximation:

- solve a model non-optimally (e.g.: use a heuristic)
  + relaxed convergence criteria
  + imprecise solution strategies (e.g.: simpler stencil for PDE solvers)
  + probabilistic solution strategies

Example: Data flow analysis

How to represent information?

Answer: Represent information as "elements" in a semi-lattice

A meet semi-lattice (Info, $\wedge$)
- domain of values Info
- **meet** operator $\wedge$: Info x Info -> Info

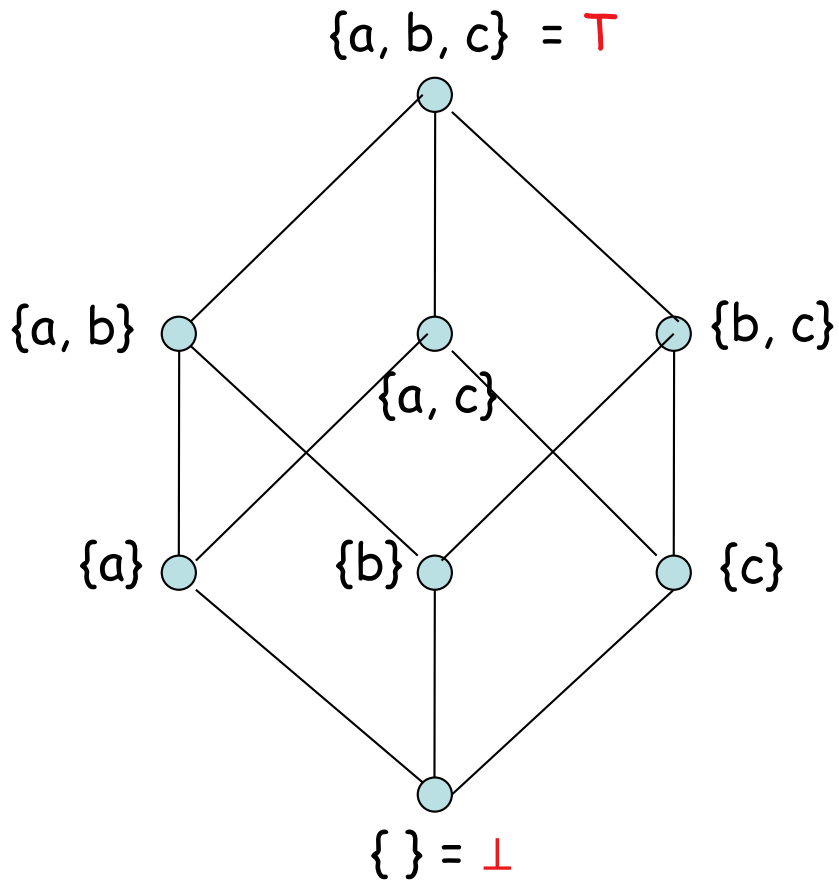
With binary operator $\wedge$ is
- commutative: $a \wedge b = b \wedge a$
- associative: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
- idempotent: $a \wedge a = a$

The meet operator $\wedge$ can be used to define a partial order over pieces of information:

$$a \wedge b = a \quad \text{IFF} \quad a \leq b$$

Lattice

Example lattice



A lattice (Info, ∧, ∨) has two binary operators
- domain of values Info
- **meet** operator ∧: Info x Info -> Info
- **join** operator : ∨: Info x Info -> Info

with binary operators are basically "duals" of each other that model information "intersection" (meet) and "union" (join). Both (Info, ∧) and (Info, ∨) are semi-lattices.

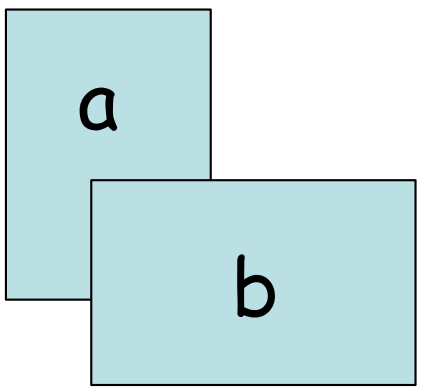
The operators ∧ **and** ∨ define a partial order over Info (pieces of information):

$$a \vee b = b \quad \text{IFF} \quad a \wedge b = a \quad \text{IFF} \quad a \leq b$$

Exercise

You are given a rectangular, 2-dimensional array / data regions.

Build a lattice that models the "intersection" and the "union" of two rectangular regions "a" and "b".
Note: the regions may or may not overlap.



a ∧ b =

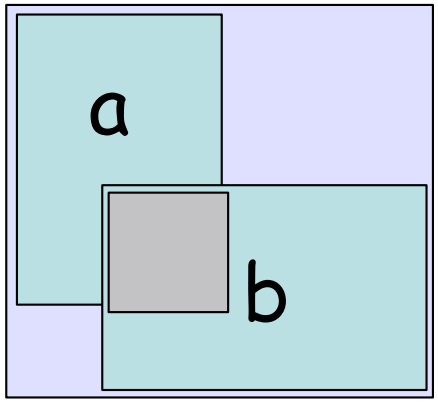
a ∨ b =

Do we have to use approximation here?

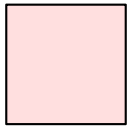
Exercise

You are given a rectangular, 2-dimensional array / data regions.

Build a lattice that models the "intersection" and the "union" of two rectangular regions "a" and "b".
Note: the regions may or may not overlap.

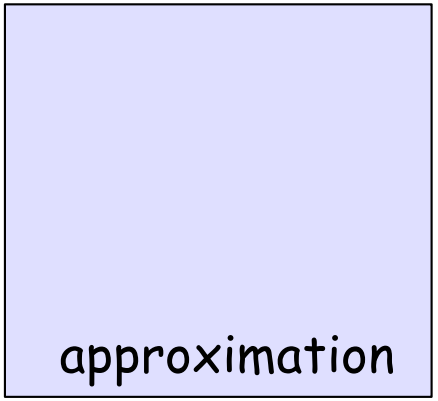


a ∧ b =



precise

a ∨ b =

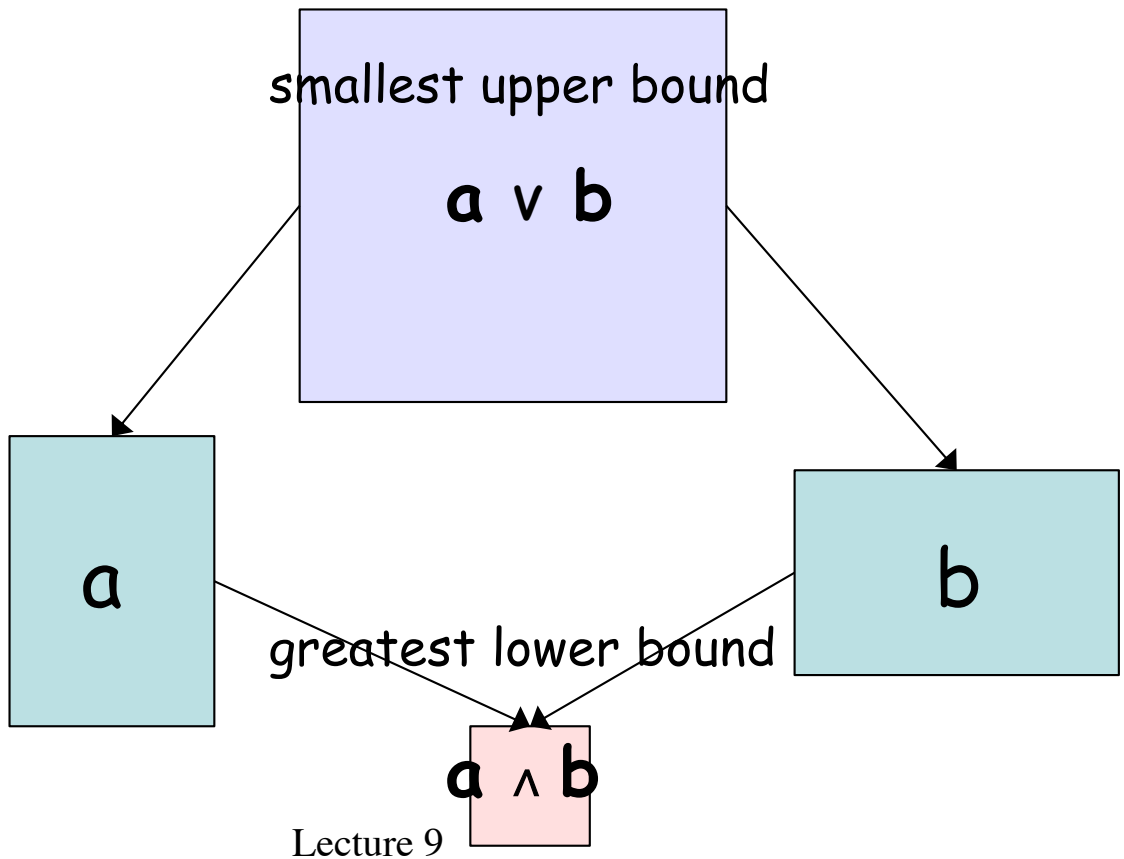


Do we have to use approximation here?

Exercise

You are given a rectangular, 2-dimensional array / data regions.

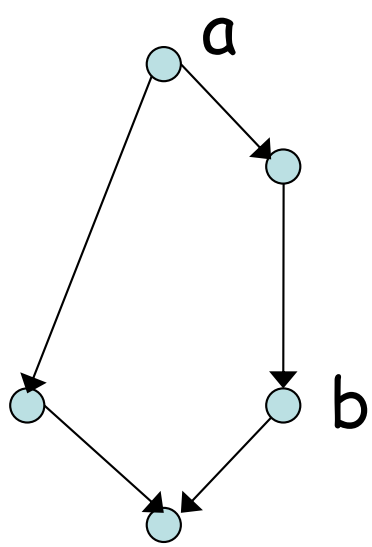
Build a lattice that models the "intersection" and the "union" of two rectangular regions "a" and "b".
Note: the regions may or may not overlap.



Formal: Drawing semi-lattice diagram

- $(x < y) \equiv (x \leq y) \wedge (x \neq y)$

- **A semi-lattice diagram:**
  - Set of nodes: set of values/information units
  - Set of edges $\{(y, x): x < y \text{ and } \neg\exists z \text{ s.t. } (x<z) \wedge (z<y)\}$



a < b, but no edge from a to b

Avoid transitive edges

Descending chain property

Definition

The **height** of a semi-lattice is the largest number of > relations that will fit in a descending chain:

x0 > x1 >...

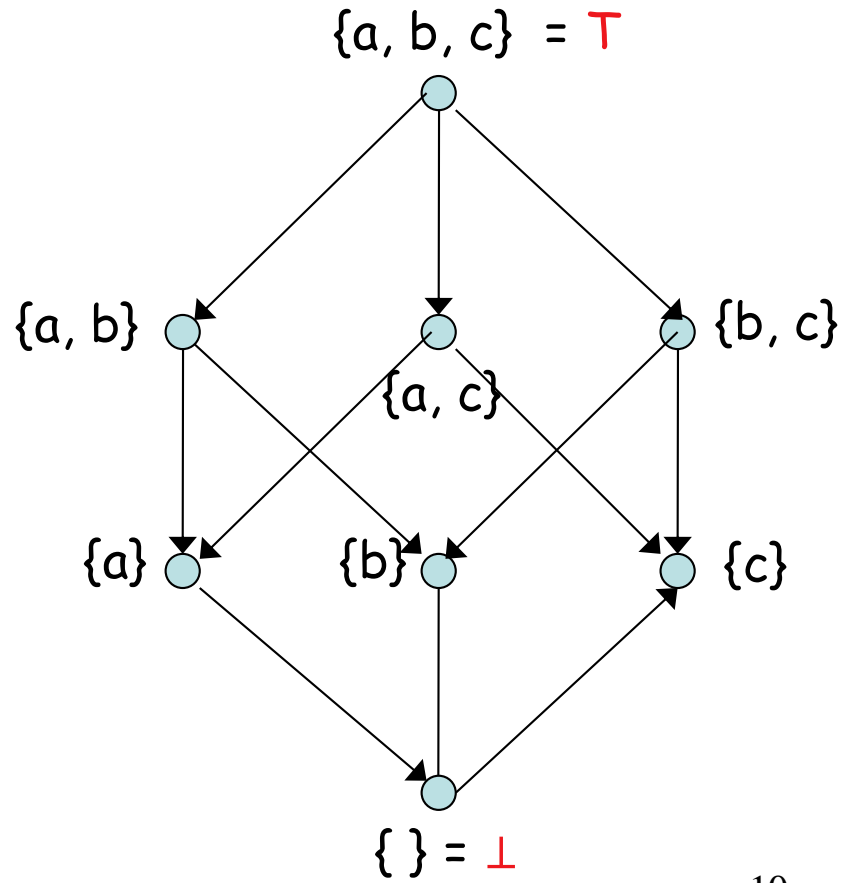
Descending chain property:

All chains are of finite height

Example:

What is the maximal height of a chain for the ({a, b, c}, ∩} semi-lattice?

What about the ({a, b, c}, ∪ } semi-lattice?



Example: Data flow analysis

One classical application for semi-lattices

Data-flow frameworks (F, Info, $\wedge$) are defined by

- A semi-lattice
  - domain of values Info
  - operator $\wedge$: Info x Info -> Info
- A family of transfer/propagation functions F : Info -> Info

Basic Properties f : Info -> Info

- There is an identity function
  - $\exists f \in F$ such that $f(x) = x$ for all x.
- Closed under composition
  - if $f1, f2 \in F$, $f1 \cdot f2 \in F$

Monotone data flow framework

A framework (F, Info, $\wedge$) is monotone iff

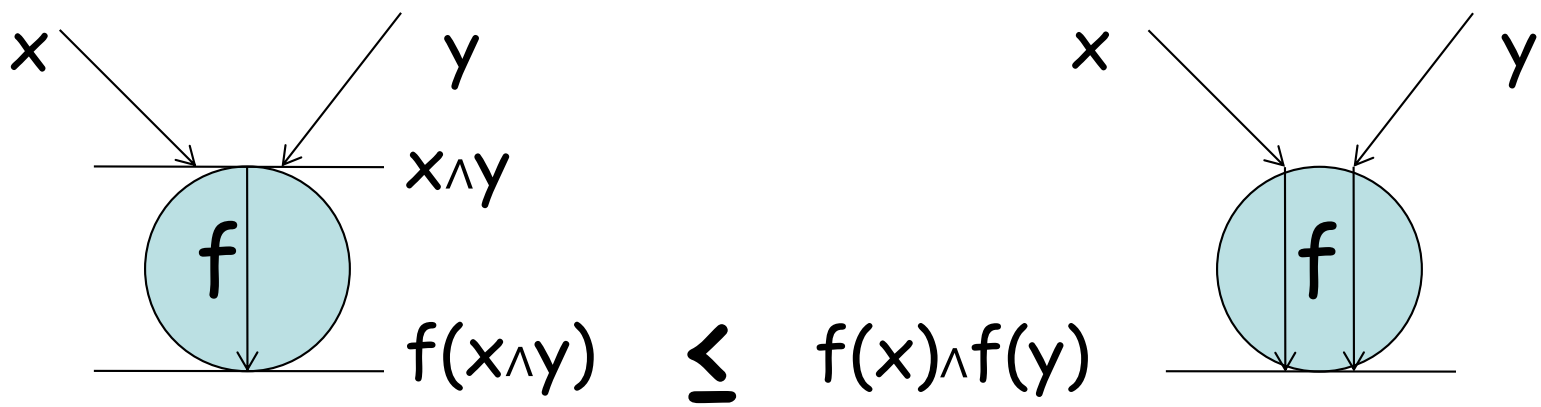
- $x \leq y$ implies $f(x) \leq f(y)$

Equivalently,

a framework (F, Info, $\wedge$) is monotone iff

- $f(x \wedge y) \leq f(x) \wedge f(y)$,
- meet inputs, then apply f is $\leq$ to apply f individually to inputs, then meet results

forward framework:



Lesson: Merging information "early" may lead to information loss, and summarizing "paths" may lead to information loss

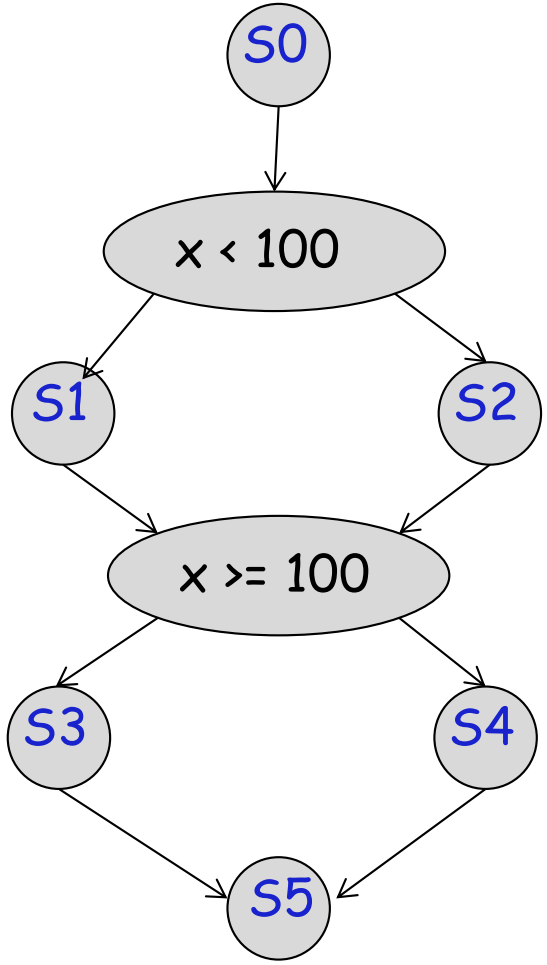
Control Flow Graph (CFG)

Directed graph representing a single program procedures.

Nodes: Basic blocks in a program
Edges: Possible control flow between basic blocks

Example:

```
S0
if ( x < 100) then
    S1
else
    S2
end
if (x >= 100) then
    S3
else
    S4
end
S5
```

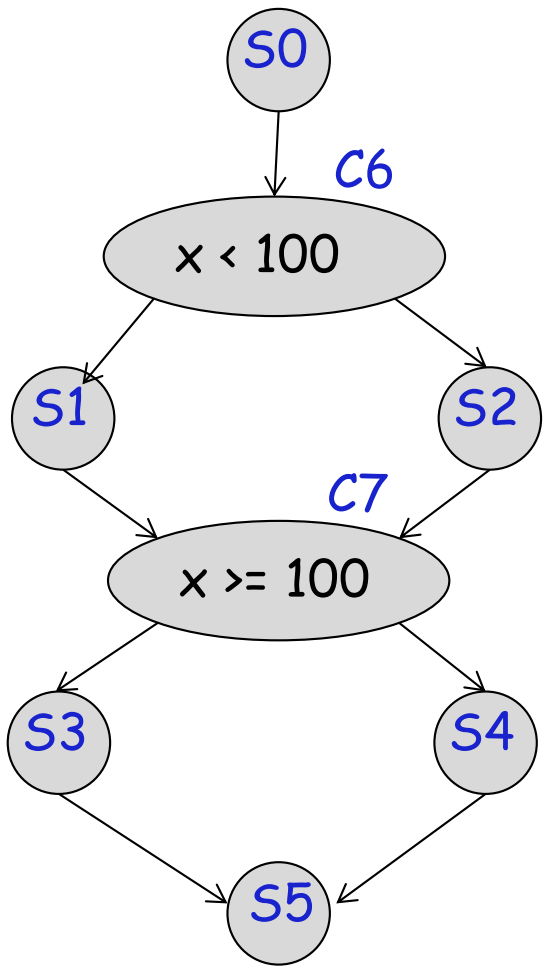


Control Flow Graph (CFG)

Homework 2: Directed graph representing a single program procedures.

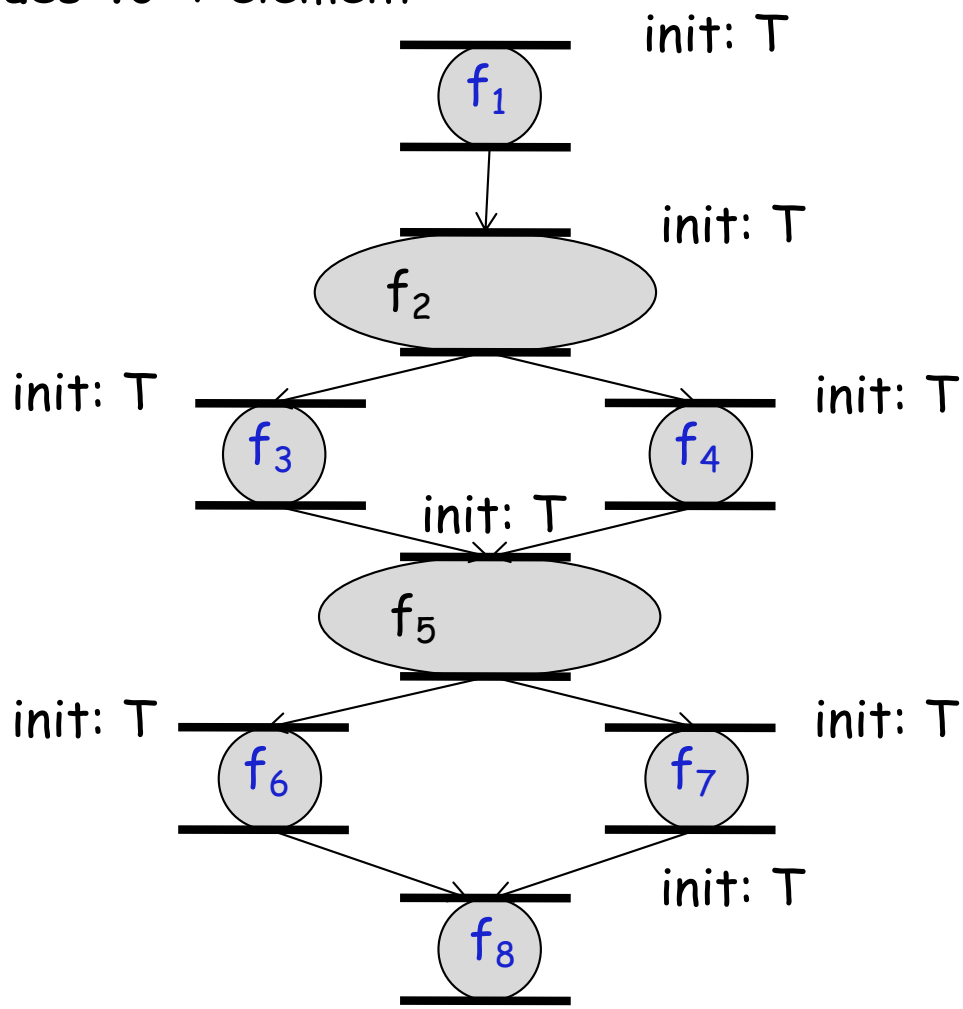
Nodes: Basic blocks in a program
Edges: Possible control flow between basic blocks

```
procedure foo( )
  S0: a = 1
      b = 2
      read(x)
  C6:    if ( x < 100) then
  S1:       c = 5
      else
  S2:       a = 2
            b = 10
      end
  C7:   if (x >= 100) then
  S3:       a = 3
            b = b + 1
      else
  S4:       b = c + 2
      end
  S5: print(a + b + c)
```



Instance of a data flow problem

- (Forward) monotone data-flow framework (F, Info, $\wedge$)
- CFG G = (N, E)
- Assignment of transfer functions in F to nodes: N -> F
- Initialization of all node values to T element

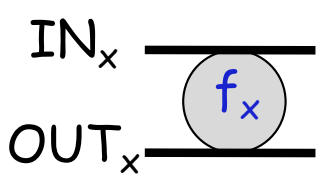


Solution: All equations

$$IN_x = \bigwedge_{p \in PRED(x)} f_p(IN_p)$$

$$f_x(IN_x) = OUT_x$$

are simultaneously satisfied.



Instance of a data flow problem

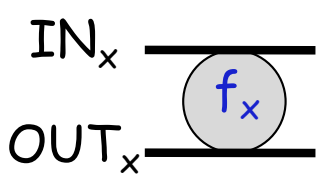
- Can compute a solution using an iterative algorithm
- Algorithm is guaranteed to terminate with MFP (maximal fixed-point) solution if
  - Info has descending chain property
  - Monotone data flow framework

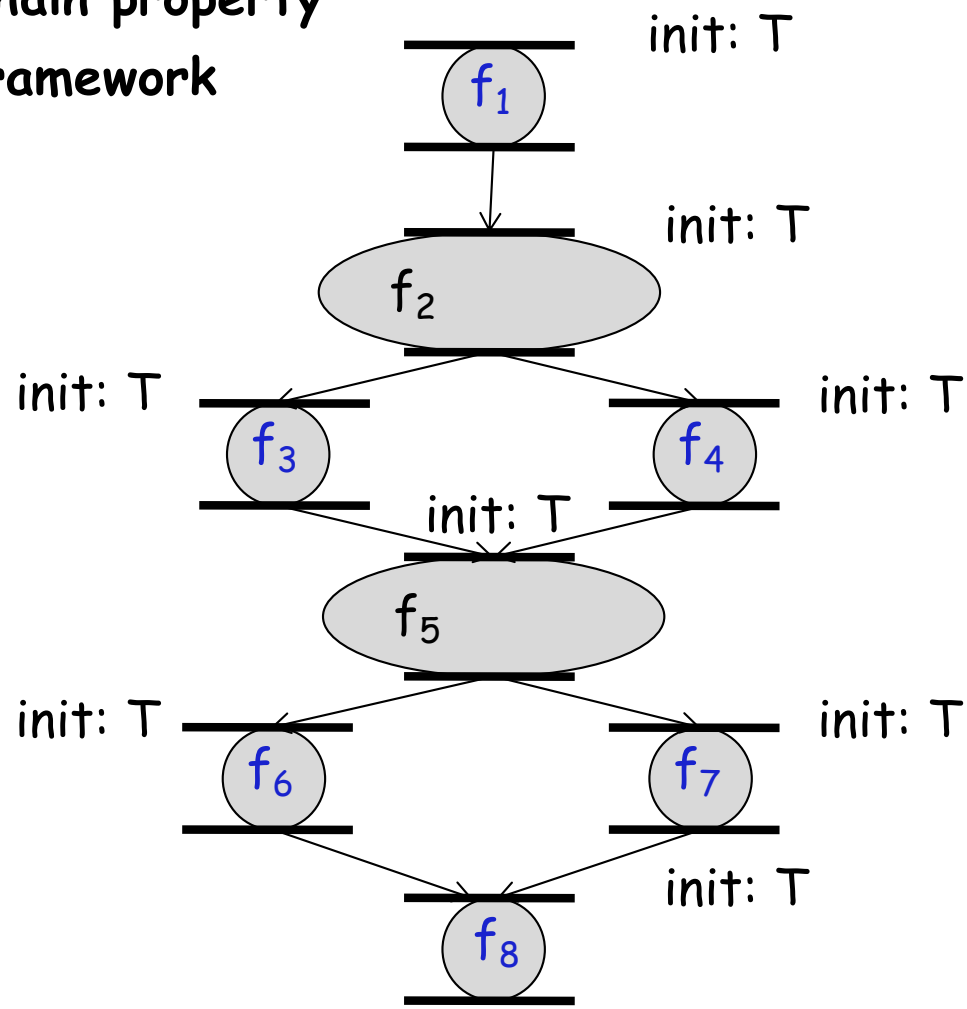
Solution: All equations

$$IN_x = \bigwedge_{p \in PRED(x)} f_p(IN_p)$$

$$f_x(IN_x) = OUT_x$$

are simultaneously satisfied.

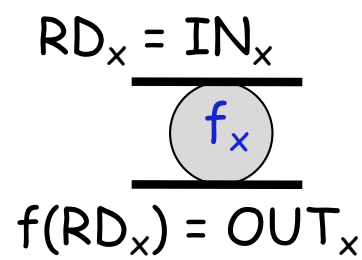




Example data flow problem

Reaching Definitions (RD)

$$RD_x = \bigcup_{p \in PRED(x)} f_p(RD_p)$$



Each transfer/propagation function f can be described by two sets, namely Gen and Kill:

f(x) = Gen ∪ (x - Kill),

Gen = {set of assignments to variables in node}
Kill = {set of assignments to same variables in other nodes}

Semi-lattice based on union ∧ = ∪

top element T is the empty set

Example data flow problem

Is RD a monoton data flow framework?

$$f(x \cup y) \subseteq f(x) \cup f(y)$$

Transfer/propagation function:

$$f(x) = Gen \cup (x - Kill)$$

Proof:

$$\begin{aligned} f(x_1 \cup x_2) &= Gen \cup ((x_1 \cup x_2) - Kill) = \\ &Gen \cup ((x_1 - Kill) \cup (x_2 - Kill)) = \\ &Gen \cup (x_1 - Kill) \cup Gen \cup (x_2 - Kill) = \\ &f(x_1) \cup f(x_2) \end{aligned}$$

□

In fact, RD is not only monotone, but also distributive:

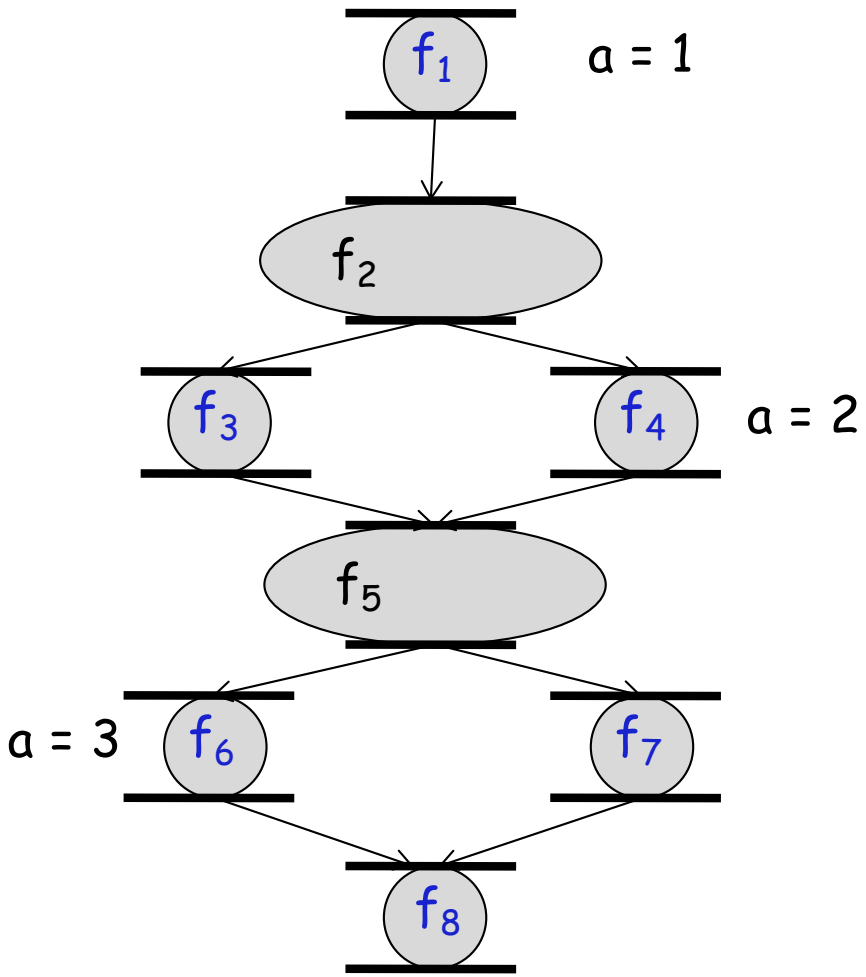
$$f(x \cup y) = f(x) \cup f(y)$$ -- no information loss

Worklist Algorithm for forward problems, e.g. RD

Input: CFG (N, E)
forward data flow framework with f(x) = Gen ∪ (x - Kill)
Output: Maximal fixed point solution represented as IN(n),

```
for n ∈ N
  IN(n) = ∅  //  init to T
  OUT(n) = GEN(n)
endfor
worklist ← n ∈ N
while ( worklist ≠ ∅)
   pick and remove a node n from worklist
   oldout(n) = OUT(n)
   IN(n) = ∪ (OUT(p)), p ∈ PRED(n)
   OUT(n) = GEN(n) ∪ (IN(n) - KILL(n))
   if oldout(n) ≠ OUT(n) then
        worklist ← worklist ∪ SUCC(n)
endwhile
```

Example data flow problem



Approximation: CFG

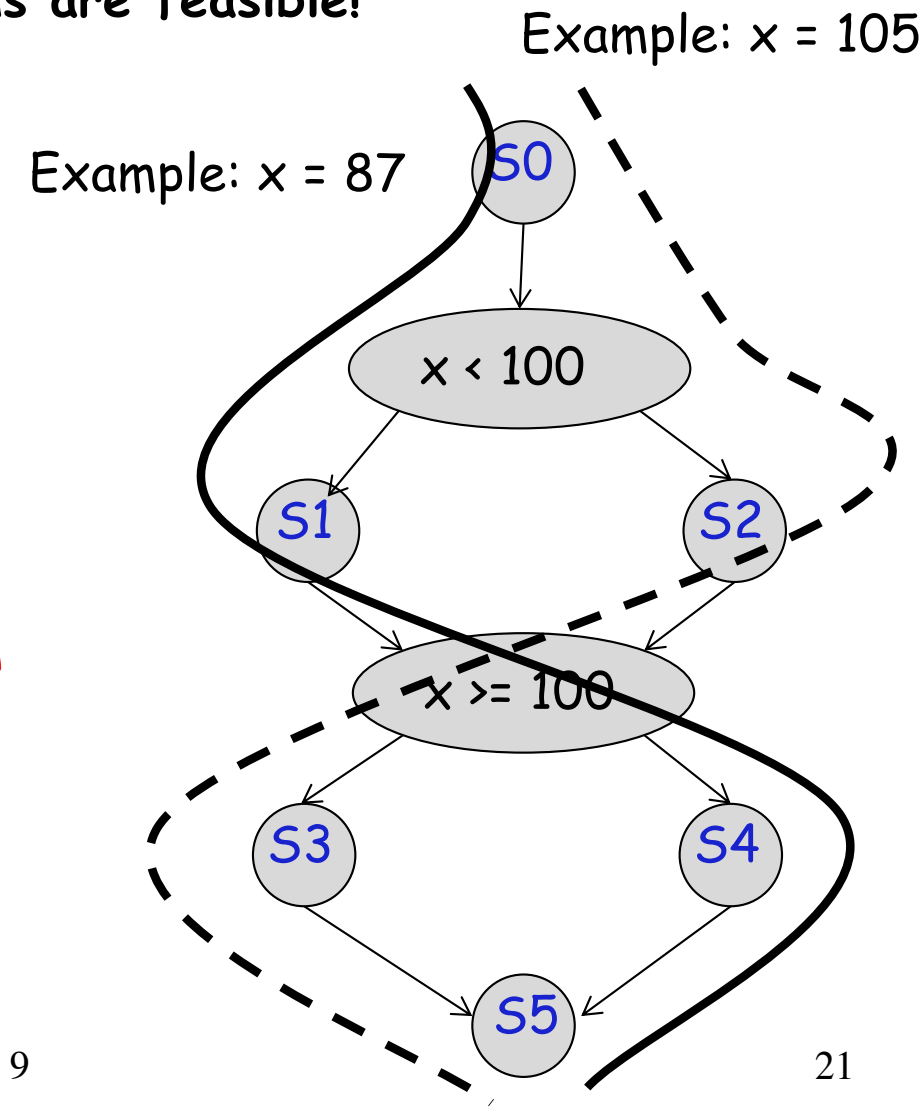
Assumption: No assignment to x in the two branches.

There are only two possible (feasible) execution path in this program. CFG assumes that all paths are feasible!

Other examples of approximations:

- computed GOTOs
- "inter-procedural" CFGs

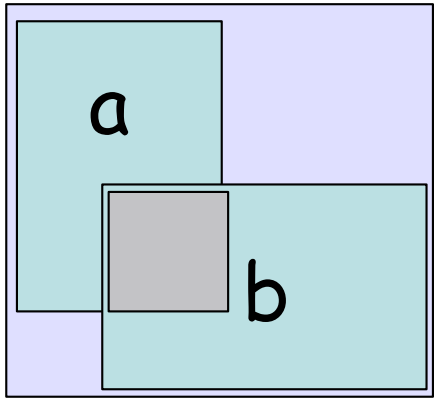
Model Approximation



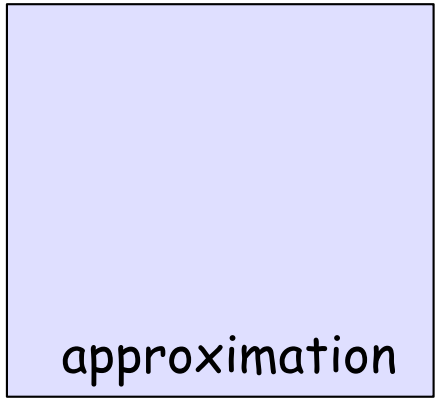
Approximation: semi-lattice

Our example join (v) semi-lattice.

Key issue: how much information to encode in every element in the "info" set of the semi-lattice.



a v b =



Data Approximation

Approximation: Fixed-point computation

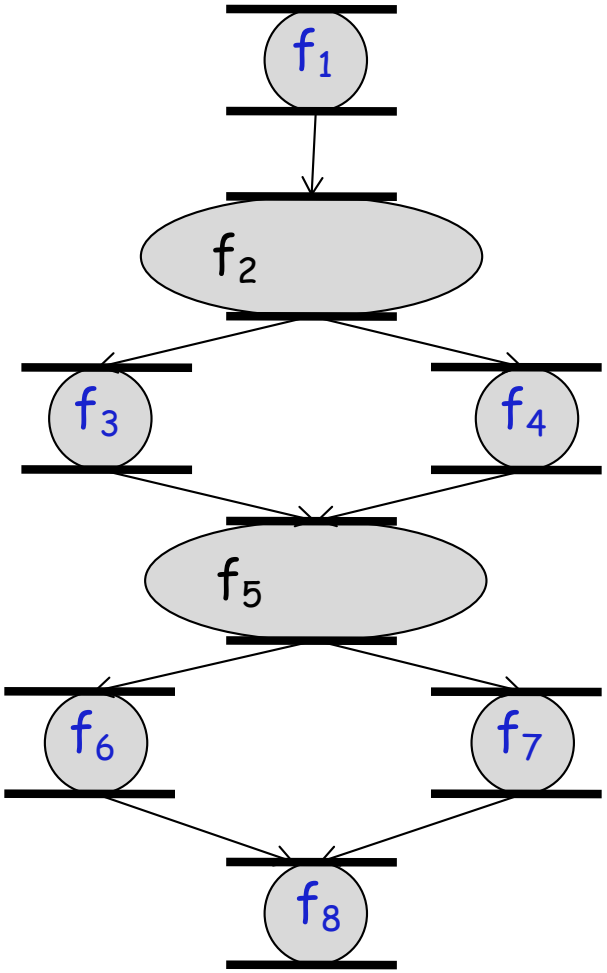
In a monotone data flow problem, information may be lost due to merging information (join or meet).

MOP (meet over all path solution)
enumerate all possible execution paths and compute information for these paths without merging. This is the best you can do!

$$MOP(n) = \bigwedge_{p \in PATH(n)} f_p(T)$$

MFP (maximal fixed-point solution)
initialize information with top element, and then propagate and merge information.

FP (fixed-point solution)
any solution



$$FP \leq MFP \leq MOP$$

Computation Approximation