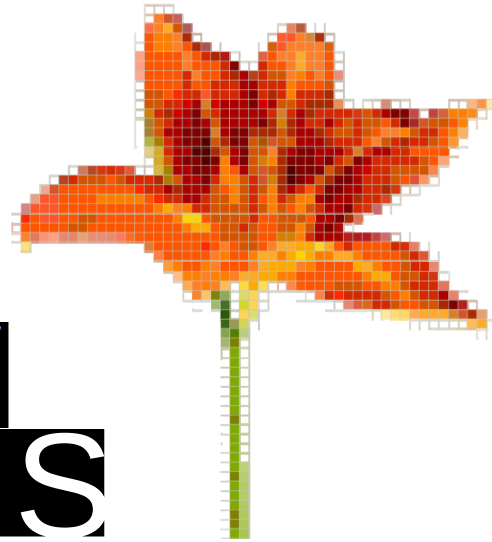
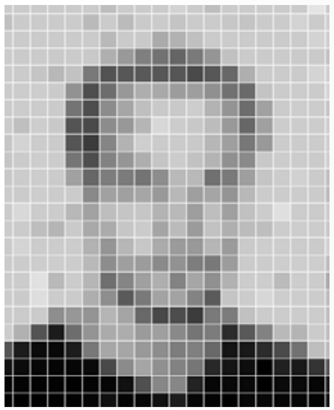
DATA-FLOW ANALYSIS



[HOW TO ANALYZE LANGUAGES AUTOMATICALLY]



Claus Brabrand

(((**brabrand@itu.dk**)))

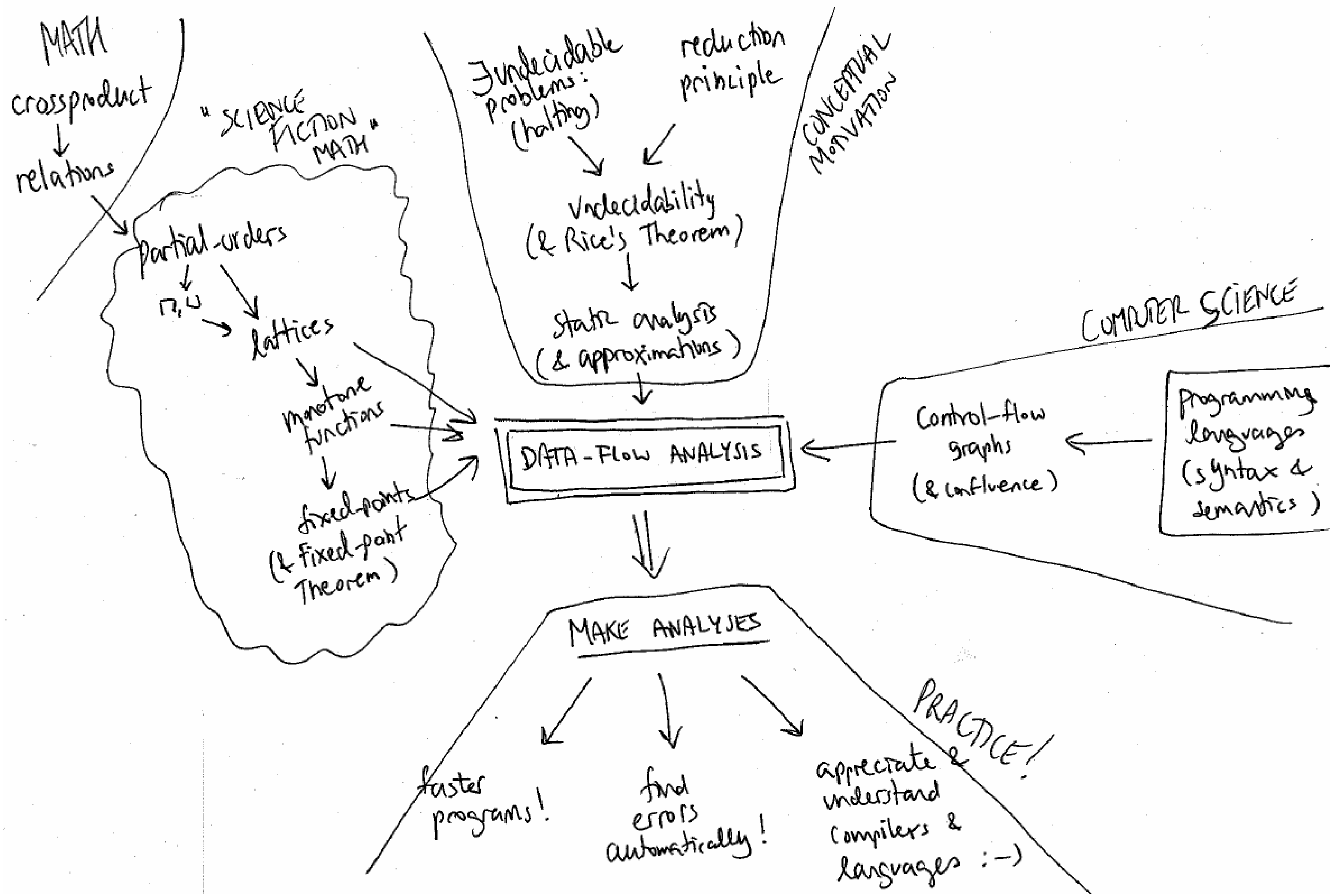
Associate Professor, Ph.D.

(((Programming, Logic, and Semantics)))

IT University of Copenhagen

⊢ Agenda

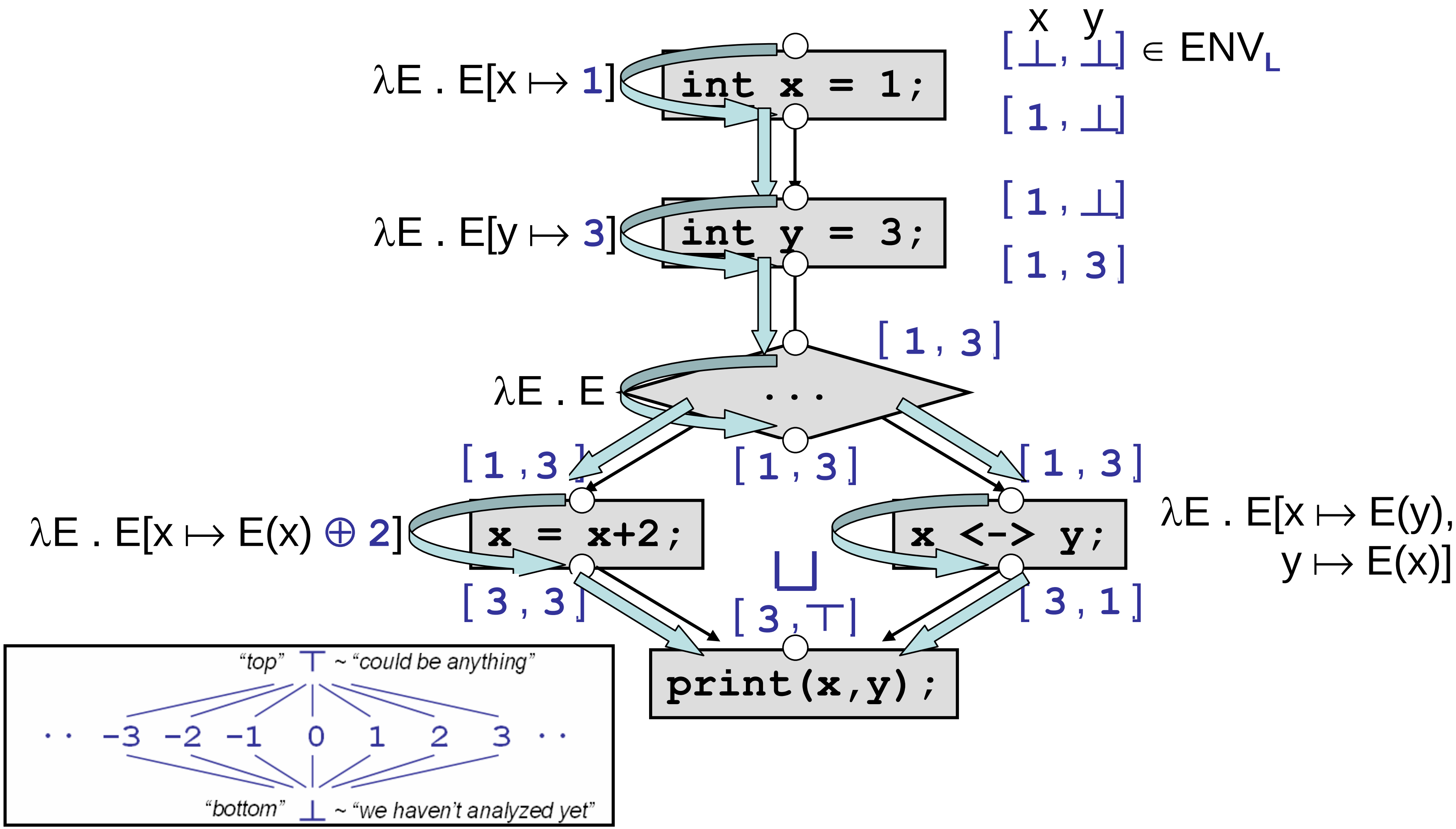
- Relations:
  - Crossproducts, powersets, and relations
- Lattices:
  - Partial-Orders, least-upper-bound, and lattices
- Monotone Functions:
  - Monotone Functions and Transfer Functions
- Fixed Points:
  - Fixed Points and Solving Recursive Equations
- Putting it all together…:
  - Example revisited


MATH
crossproduct
relations
"SCIENCE FICTION MATH"
partial-orders
⊓, ⊔
lattices
monotone functions
fixed-points
(& fixed-point Theorem)
∃ undecidable problems: (halting)
reduction principle
undecidability
(& Rice's Theorem)
static analysis
(& approximations)
CONCEPTUAL MOTIVATION
DATA-FLOW ANALYSIS
COMPUTER SCIENCE
Control-flow graphs
(& confluence)
Programming languages
(syntax & semantics)
MAKE ANALYSES
faster programs!
find errors automatically!
appreciate & understand compilers & languages :-)
PRACTICE!

⊢ Quick recap:

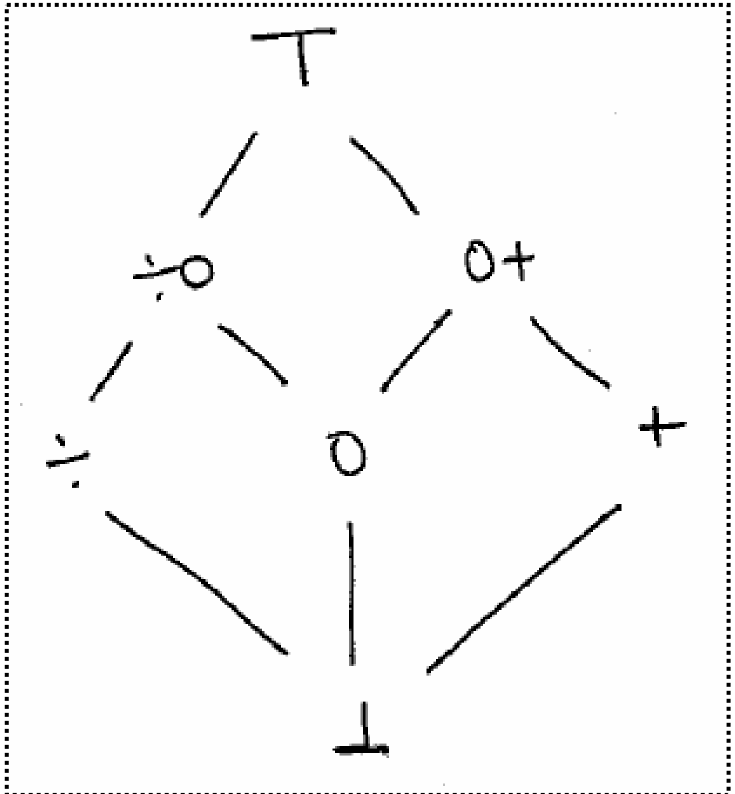
We (only) need 3 things:

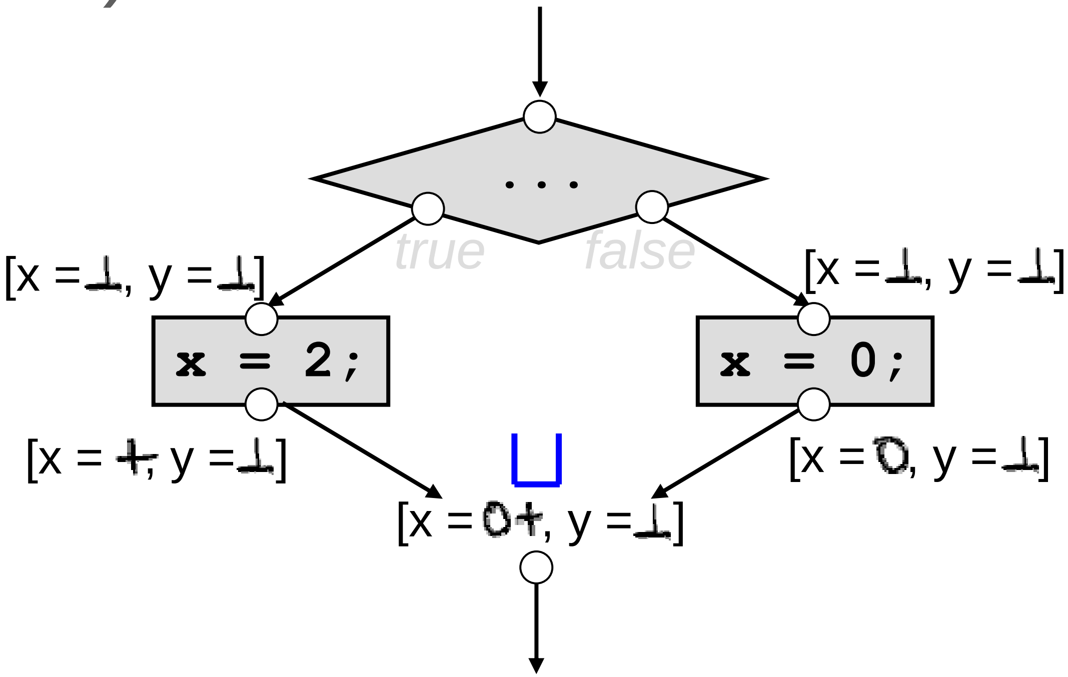
- A ***control-flow graph***
- A ***lattice***
- ***Transfer functions***



Example: Least upper bound

- Analyses use '⊔' to ***combine information***
  (at confluence points):





Lattice

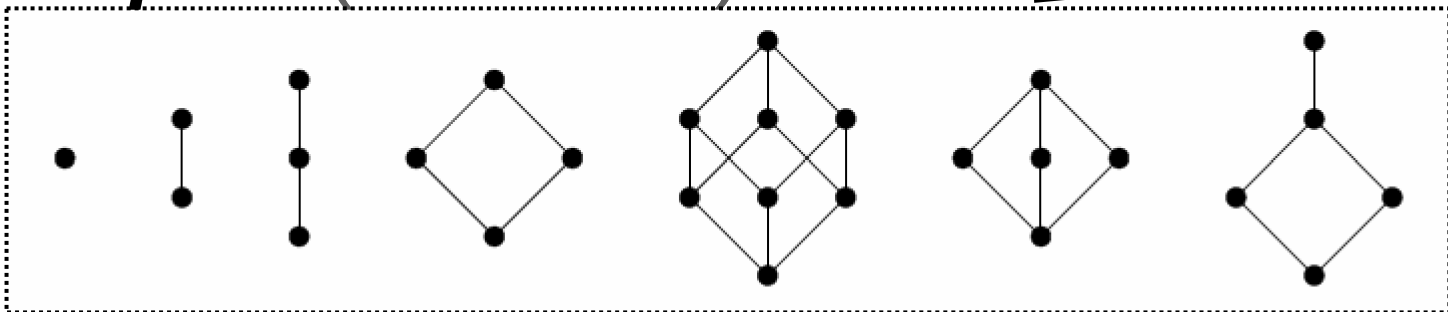
⊢ Lattice

Simplification: if we restrict to ***finite height lattices*** (which are the ones used in practice); then '$\sqcup$' only has to exist for each ***pair*** of elements (i.e., not for all subsets of elements)

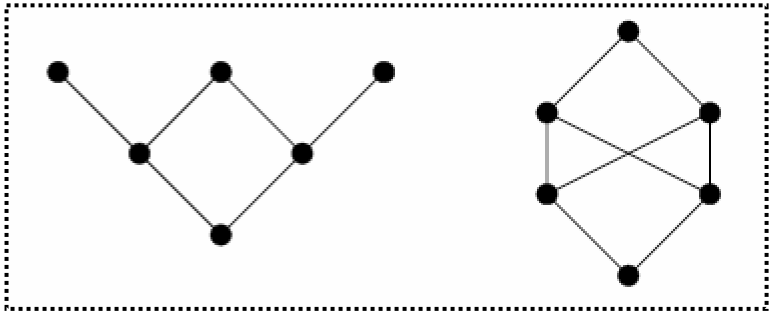
- A ***Lattice*** is a ***Partial Orde***
  - …where '$\sqcup S$' exists for all subsets $S \subseteq L$

"We must be able to combine information"

- ***Examples*** (of lattices):



- ***Non-Examples*** (of non-lattices):

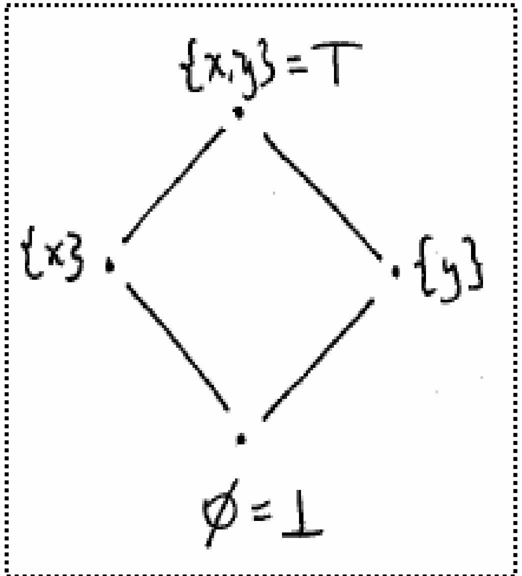


Why?

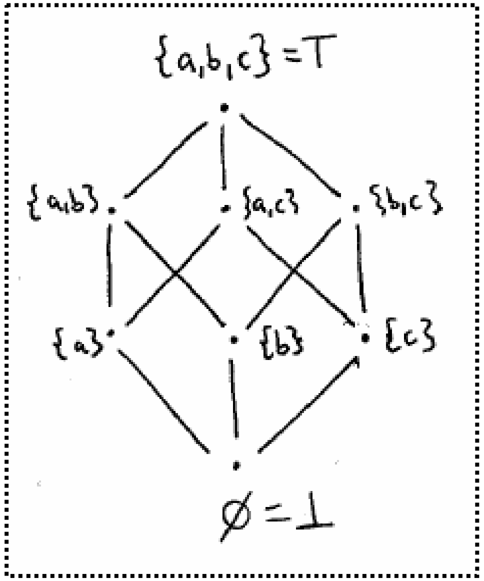
⊢ Power-Lattices

- Powerset Lattices (as Hasse Diagrams):
  - $\mathcal{P}$(**{x,y}**) = { Ø, **{x}**, **{y}**, **{x,y}** }
  - ...ordered under '$\sqsubseteq$' = '$\subseteq$' (i.e., *subset inclusion*):

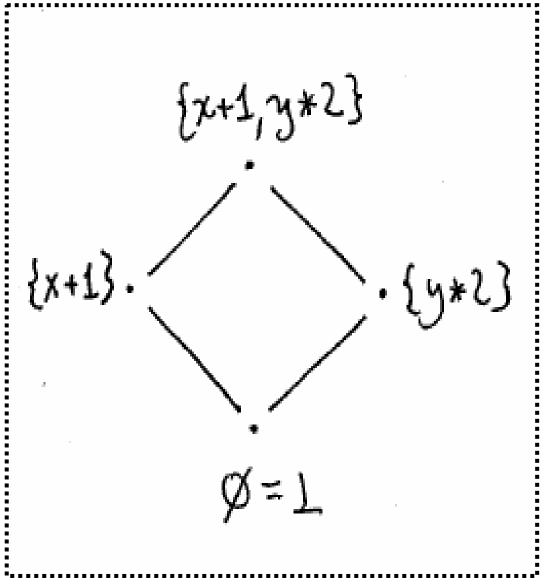
$\mathcal{P}$(**{x,y}**)



$\mathcal{P}$(**{a,b,c}**)



$\mathcal{P}$(**{x+1, y*2}**)



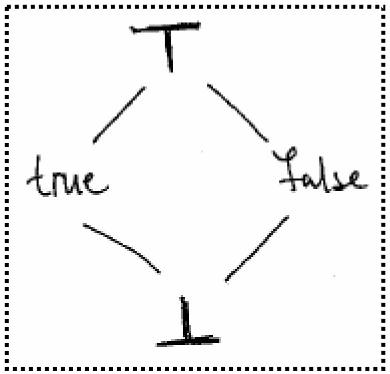
⊢ Agenda

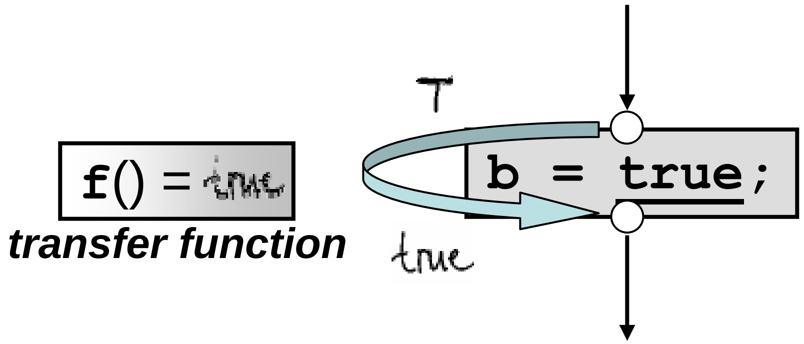
- Relations:
  - Crossproducts, powersets, and relations
- Lattices:
  - Partial-Orders, least-upper-bound, and lattices
- Monotone Functions:
  - Monotone Functions and Transfer Functions
- Fixed Points:
  - Fixed Points and Solving Recursive Equations
- Putting it all together…:
  - Example revisited

⊢ Transfer Functions

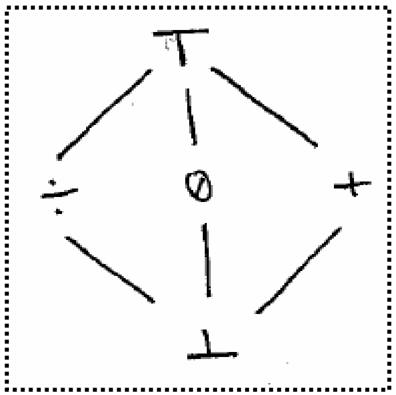
- ***Constant*** Transfer functions:

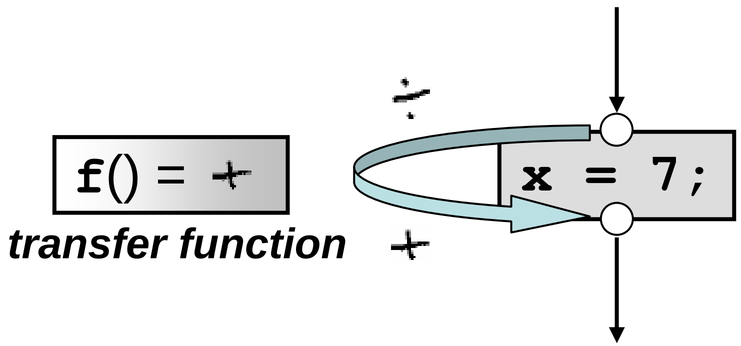
Analysing value-of-'b'





Analysing sign-of-'x'

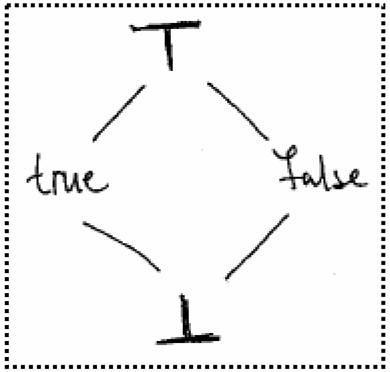




Transfer Functions

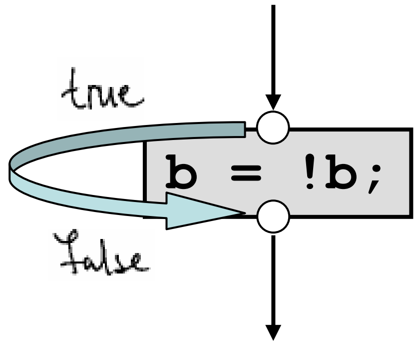
- ***Unary*** Transfer functions:

Analysing value-of-'b'

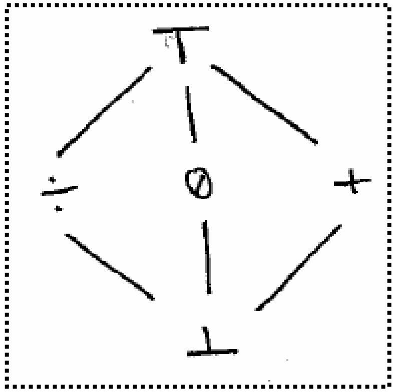


$$f_!(l) = \begin{cases} \top & \text{, if } l = \top \\ true & \text{, if } l = false \\ false & \text{, if } l = true \\ \bot & \text{, if } l = \bot \end{cases}$$

transfer function



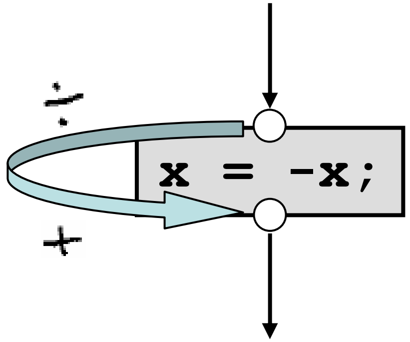
Analysing sign-of-'x'



$f(l) =$

<u>Exercise</u>:
what's this transfer function?

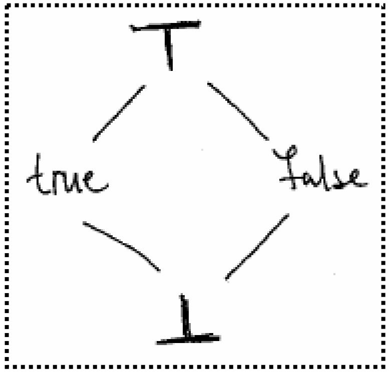
transfer function



⊢ Transfer Functions

- ***Binary*** Transfer functions:

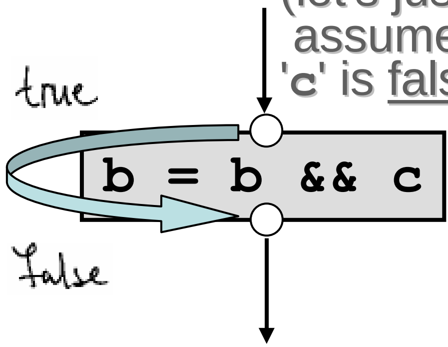
Analysing value-of-'b'



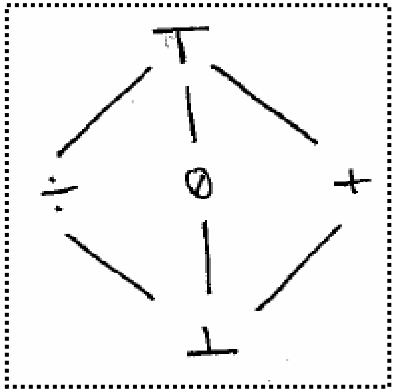
$f_{\&\&}$	⊥	true	false	⊤
⊥	⊥	⊥	⊥	⊥
true	⊥	true	false	⊤
false	⊥	false	false	⊤
⊤	⊥	⊤	⊤	⊤

transfer function

(let's just assume 'c' is false)



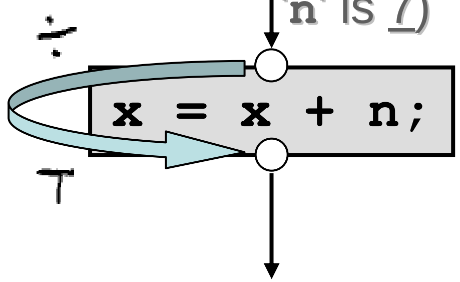
Analysing sign-of-'x'



Exercise: what's this transfer function?

transfer function

(let's just assume 'n' is 7)



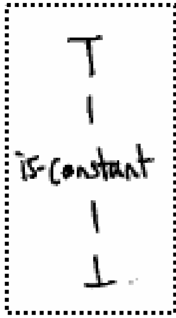
Environments

Runtime: $Var \to Val$

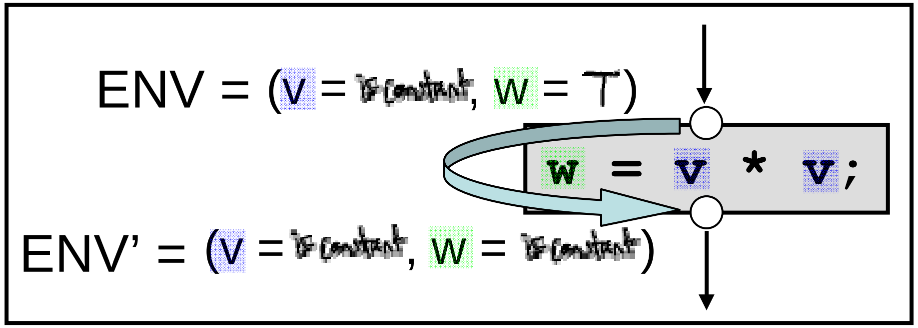
Analysis: $Var \to \mathcal{L}$ *(i.e., abstract values)*

We need Transfer Functions on *Environments*!

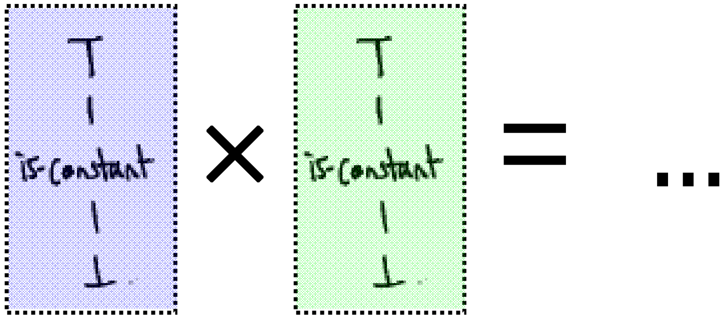
- Say lattice ***L*** analyses ”constantness” of ***one single value***:



- We need ***environments*** for analyzing:
  - (tracking both ’v’ and ’w’):



  - i.e. we need lattice ’***L*** × ***L***’**:**
    - Note:

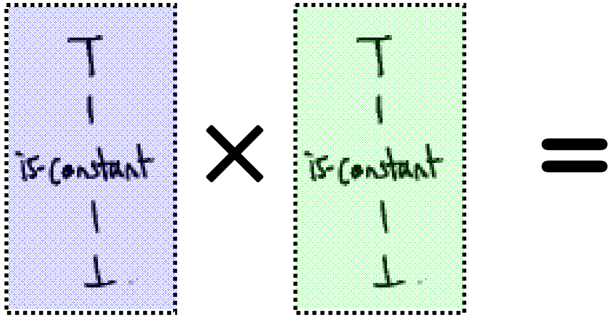


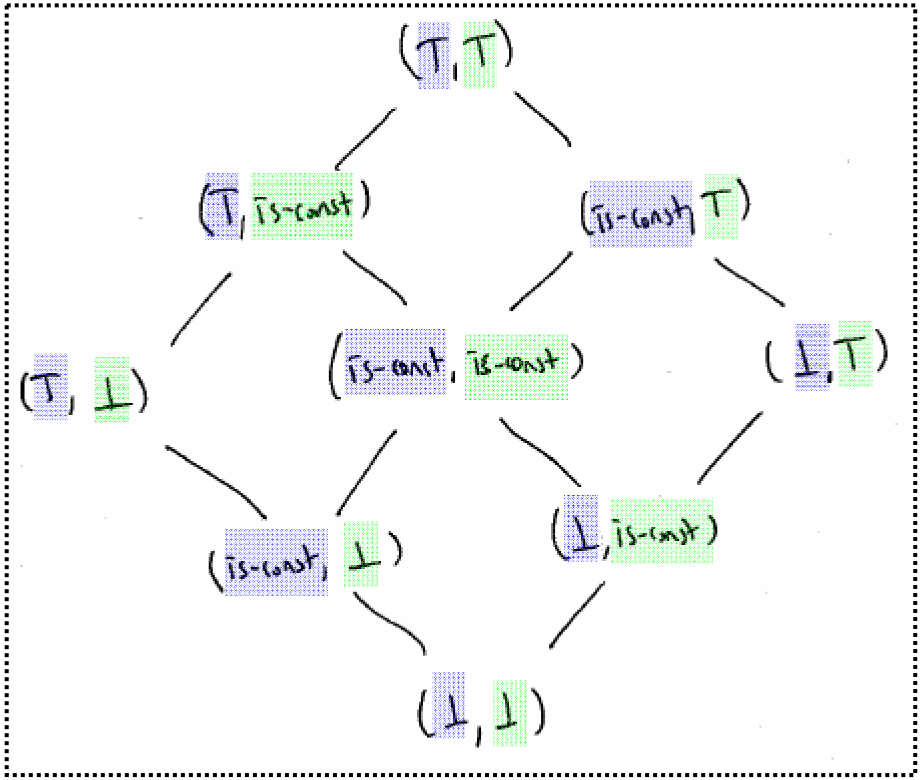
$\boldsymbol{L} \times \boldsymbol{L} \;\approx\; \{x,y\} \to \boldsymbol{L}$

$\boldsymbol{L}^{|VAR|} \;\approx\; \boldsymbol{VAR} \to \boldsymbol{L}$

Crossproduct Lattice

- ***Environment Lattice:***
  - ***'L × L':***

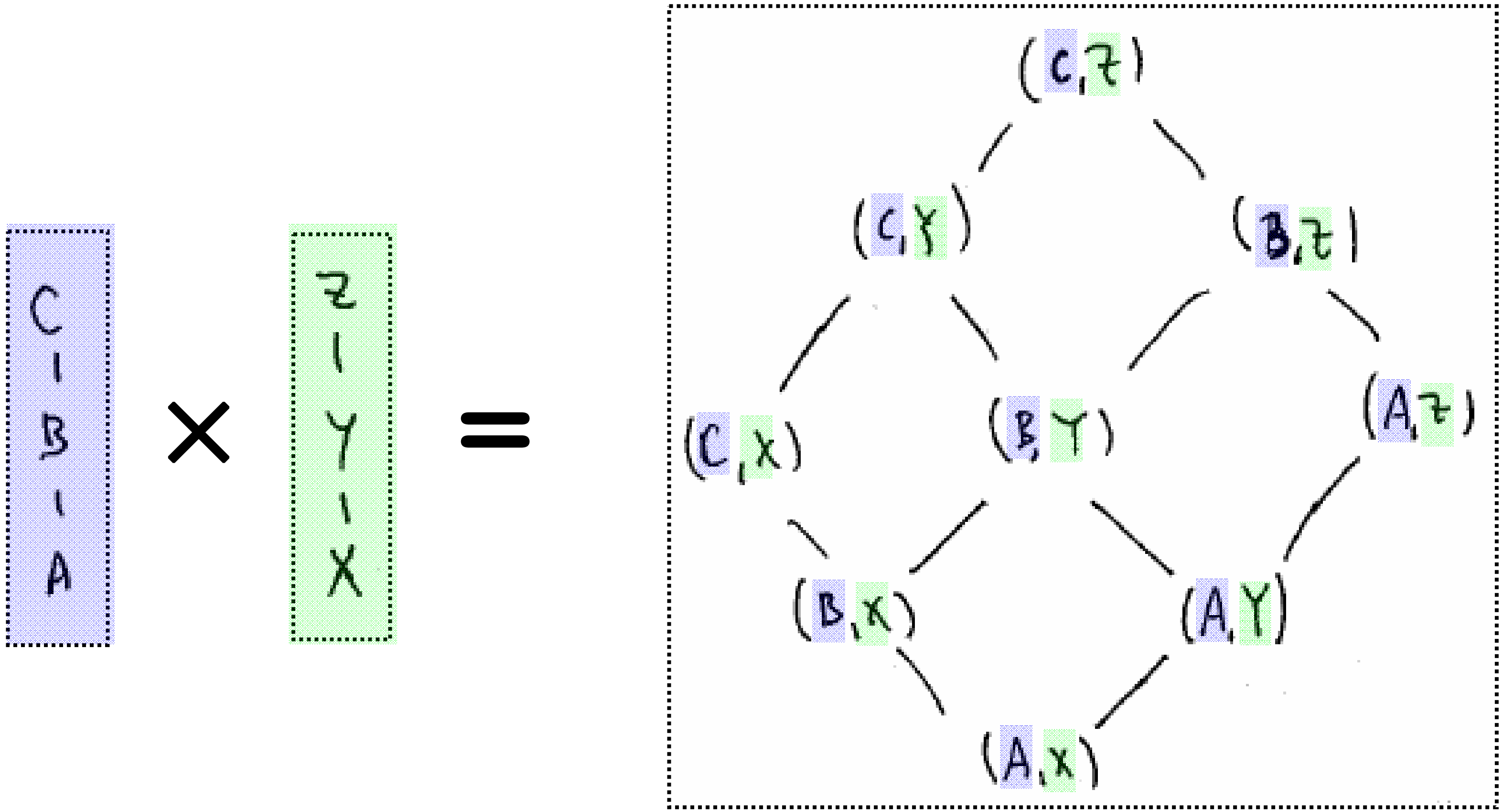




NB: if ***L*** is a lattice, then ***L*** × ***L*** is always a lattice

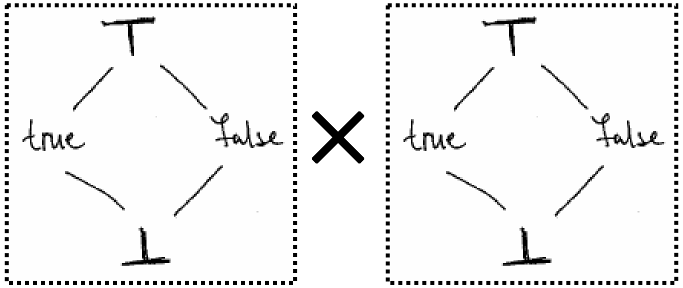
Exercise

- ***Calculate crossproduct lattice:***



Transfer Functions
on Environment Lattices

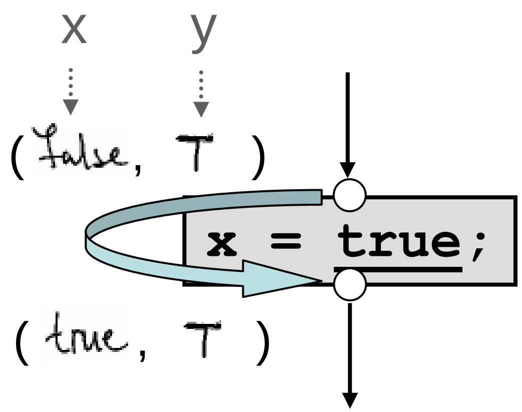
- ***Constant*** Transfer functions:

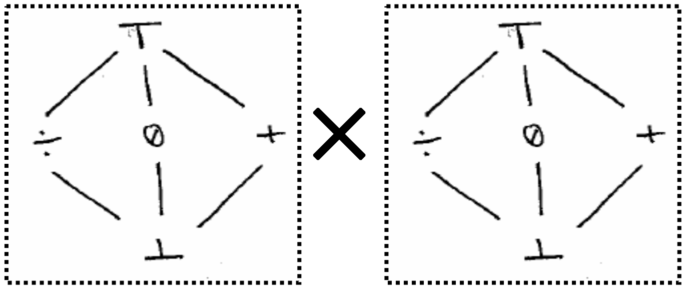


for 'x' for 'y'

$\lambda E\ .\ E[x = \text{true}]$

ENV transfer function

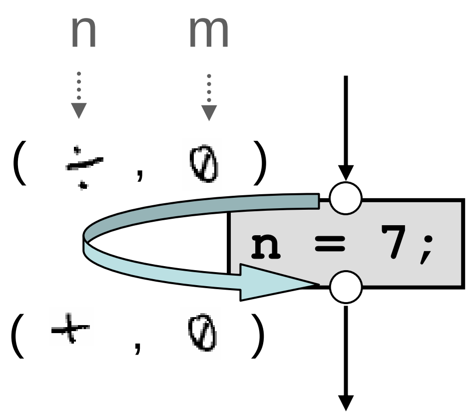




for 'n' for 'm'

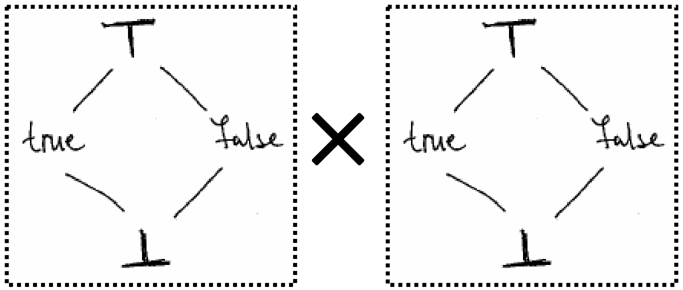
$\lambda E\ .\ E[n = +]$

ENV transfer function

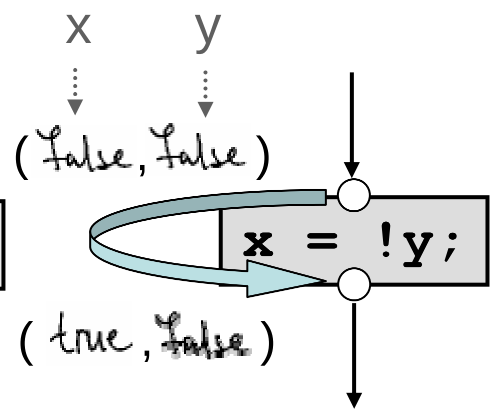


Transfer Functions *on Environment Lattices*

- ***Unary*** Transfer functions:

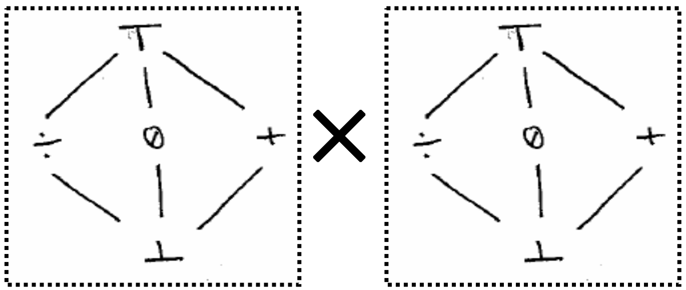


for 'x' for 'y'



λE . E[x = **f**$_{\textbf{not}}$(E(y))]

ENV transfer function

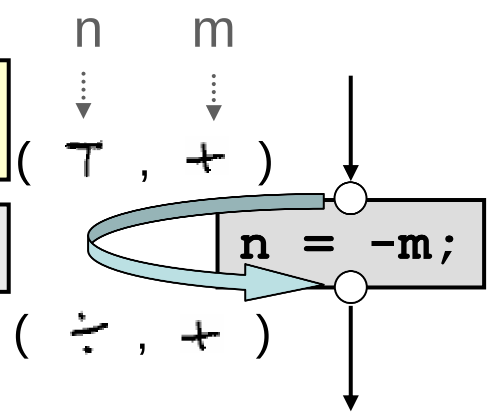


for 'n' for 'm'

<u>EXERCISE</u>:
(what's the transfer func?)

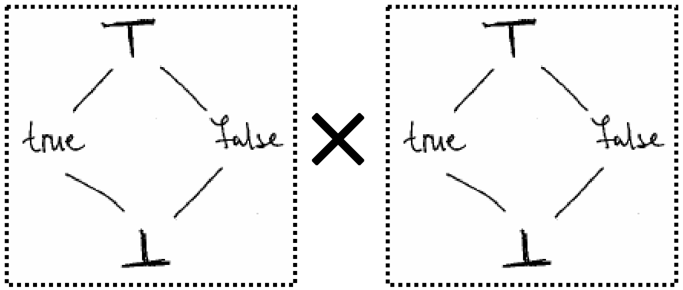
λE . E[n = **f**$_{\textbf{-}}$(E(m))]

ENV transfer function



Transfer Functions *on Environment Lattices*

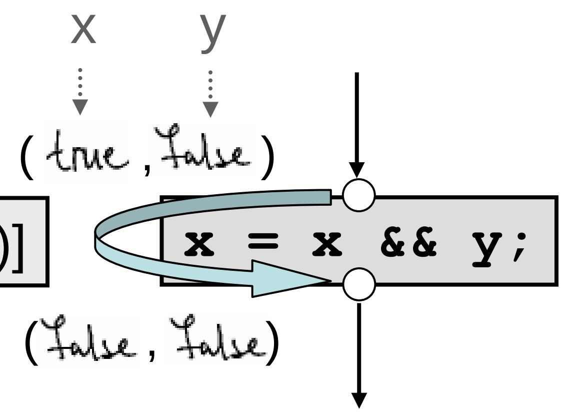
- ***Binary*** Transfer functions:

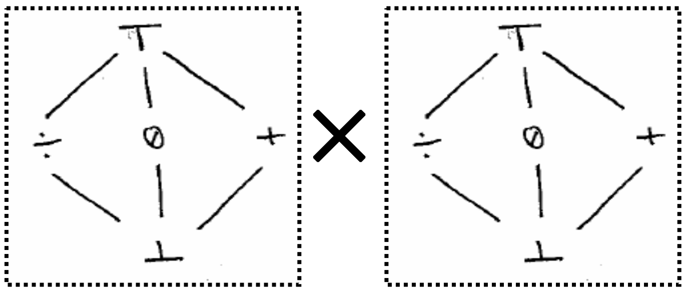


for ’x’ for ’y’

$\lambda E\ .\ E[x = E(x) \ \&\&_L\ E(y)]$

ENV transfer function

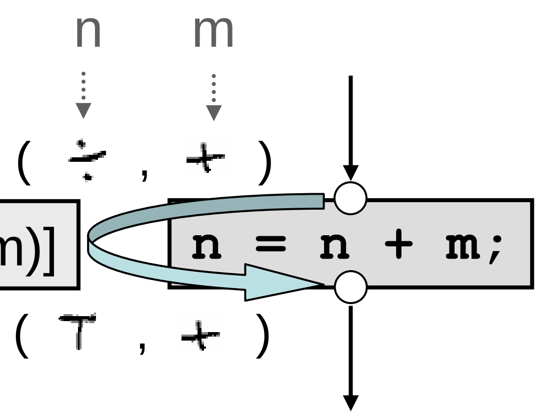




for ’n’ for ’m’

$\lambda E\ .\ E[x = E(n) \oplus_L E(m)]$

ENV transfer function



Monotonicity

Monotone *Transfer Functions*

⊢ Monotone Functions

- ***Monotone*** function $\boxed{f : L \rightarrow L}$ (on a lattice L):
  - $\boxed{\forall x,y \in L: \quad x \sqsubseteq y \;\Rightarrow\; f(x) \sqsubseteq f(y)}$

- **Note:**
  - *this is **not** saying that '`f`' is **ascending**:* $\boxed{\cancel{\forall x \in L: \quad x \sqsubseteq f(x)}}$

All the transfer functions you have seen were monotone!   :-)

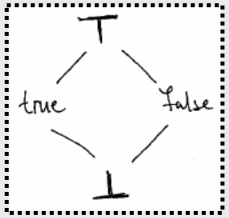
- *Examples:*

$\boxed{f() = true}$

$$f_1(l) = \begin{cases} \top & \text{, if } l = \top \\ true & \text{, if } l = false \\ false & \text{, if } l = true \\ \bot & \text{, if } l = \bot \end{cases}$$

f_{and}	$\bot$	true	false	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
true	$\bot$	true	false	$\top$
false	$\bot$	false	false	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$

…on lattice:



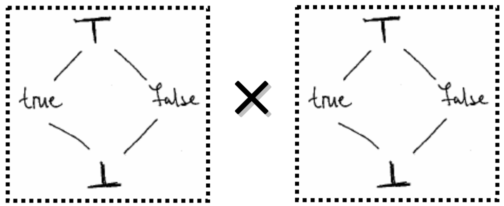
⊢ Monotone Func's (cont'd)

- More Examples:
  - $\lambda E \,.\, E[x = \textit{true}]$
  - $\lambda E \,.\, E[x = \mathbf{f}_{\mathbf{not}}(E(y))]$
  - $\lambda E \,.\, E[x = E(x) \;\&\&_L\; E(y)]$

Monotonicity:

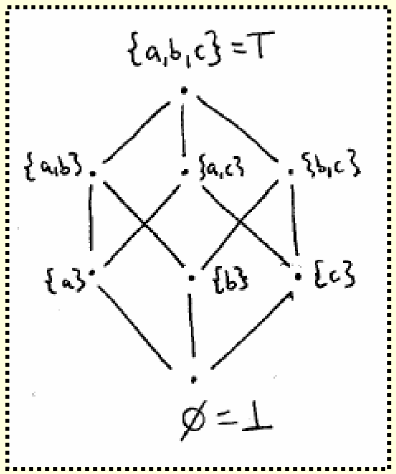
$$\forall x,y \in \boldsymbol{L}: \quad x \sqsubseteq y \;\Rightarrow\; f(x) \sqsubseteq f(y)$$

...on lattice:



- Exercise:
  - Check monotonicity of '`f`' below:
    - `1) f(X) = {a,b}`
    - `2) f(X) = X ∪ {a}`
    - `3) f(X) = {a,b} \ X`
    - 4) `f(X) = X`^{C}

...on lattice:
$\mathcal{P}(\{a,b,c\})$
$= 2^{\{a,b,c\}}$



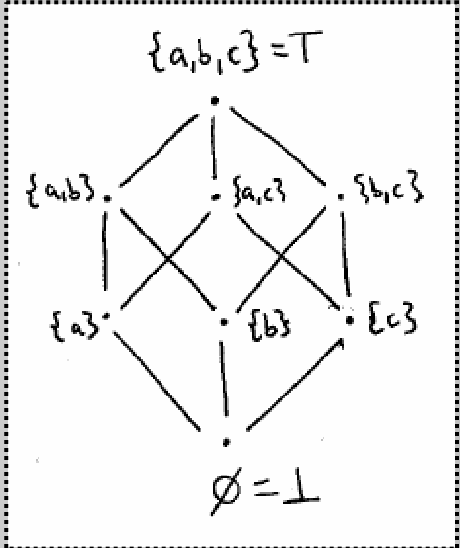
⊢ Agenda

- Relations:
  - Crossproducts, powersets, and relations
- Lattices:
  - Partial-Orders, least-upper-bound, and lattices
- Monotone Functions:
  - Monotone Functions and Transfer Functions
- Fixed Points:
  - Fixed Points and Solving Recursive Equations
- Putting it all together…:
  - Example revisited

⊢ Fixed-Points

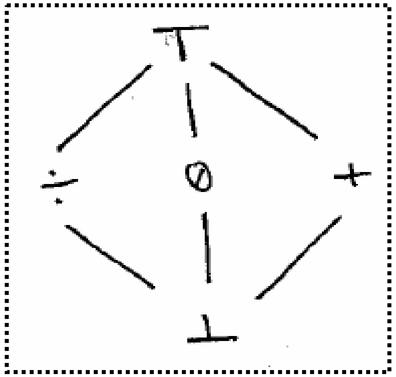
- A ***fixed-point*** for a function $\boxed{f : L \rightarrow L}$
- …is ***an element*** $\boxed{\ell \in L}$
  - such that: $\boxed{\ell = f(\ell)}$

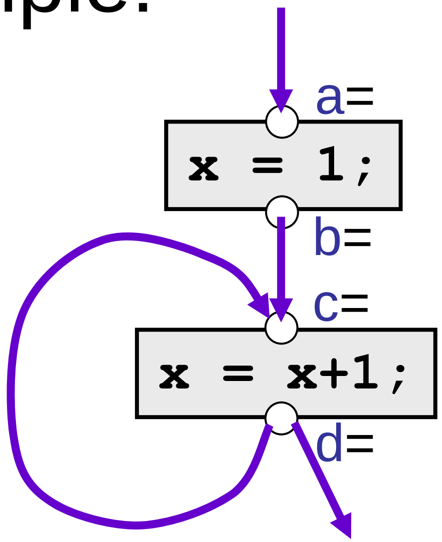
- **Example:**
  - Function: $\boxed{f(X) = X \cup \{c\}}$
    - …over $\mathcal{P}(\{a,b,c\})$
  - '**f**' has many fixed-points:
    - {c} ← ***LEAST*** ***fixed point***
    - {a,c}
    - …
    - {a,b,c} ***In fact, anything that includes 'c'***

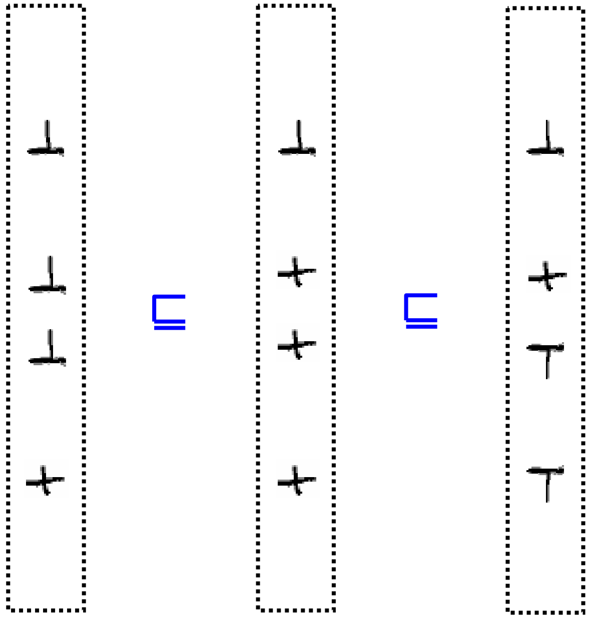


⊢ Another Example

- Another Example:







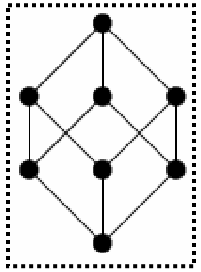
not a fixed-point! ***LEAST fixed-point!*** ***fixed-point!***

Recursive equations:

$$
\begin{aligned}
a &= \bot \\
b &= f_{x=1}(a) \\
c &= b \sqcup d \\
d &= f_{x=x+1}(c)
\end{aligned}
$$

⊢ Fixed-Point Theorem!

- If:
  - *'**f**'* is a ***monotone function:*** $f : L \rightarrow L$
  - *…over a **lattice L** (with finite height)**:***



- Then:
  - *'**f**'* is ***guaranteed*** to have a ***unique least fixed-point***
  - …which is ***computable*** as:
    - $$\texttt{fix(f)} = \bigsqcup_{i \geq 0} f^i(\bot)$$

Proof is quite simple
[cf. Notes, p.13 top]

- Intuition: $\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq$ ...until equality

⊢ Now you can...

- Now you can…
  - Solve ***ANY*** recursive equations (involving monotone functions over lattices):
    - `x = f(x,y,z)`
    - `y = g(x,y,z)`
    - `z = h(x,y,z)`

- Which means that you can solve ***any recursive data-flow analysis equation!*** **:-)**

⊢ Agenda

- Relations:
  - Crossproducts and relations
- Lattices:
  - Partial-Orders, least-upper-bound, and lattices
- Monotone Functions:
  - Monotone Functions and Transfer Functions
- Fixed Points:
  - Fixed Points and Solving Recursive Equations
- Putting it all together…:
  - Example revisited

⊢ All you need is...:

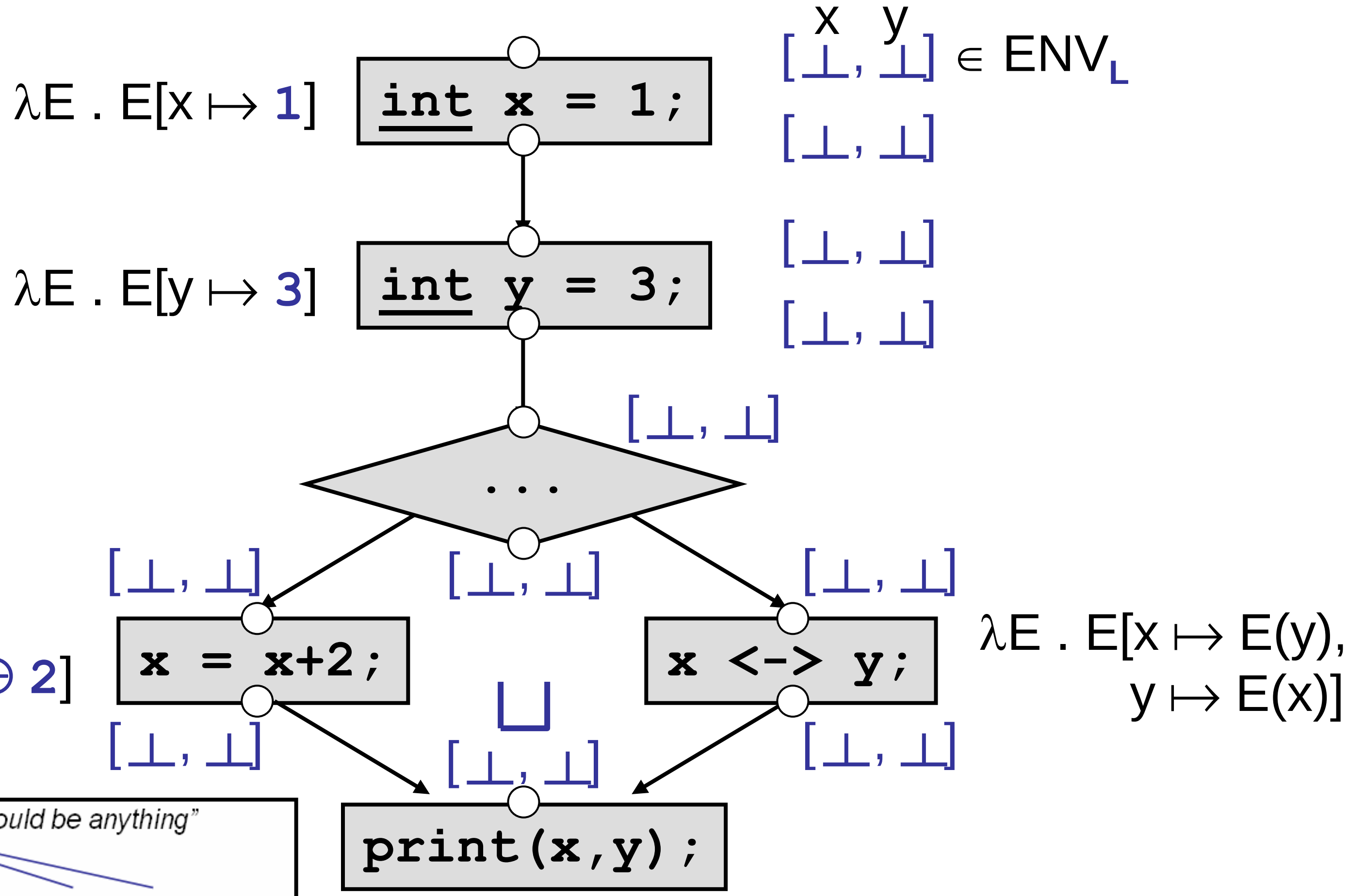
We (only) need 3 things:
- A ***control-flow graph***
- A ***lattice***
- ***Transfer functions***

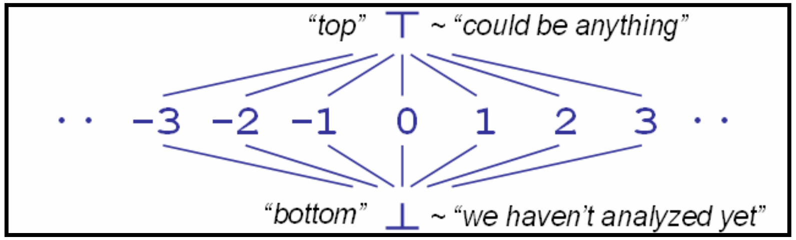
Given program:

```
int x = 1;
int y = 3;

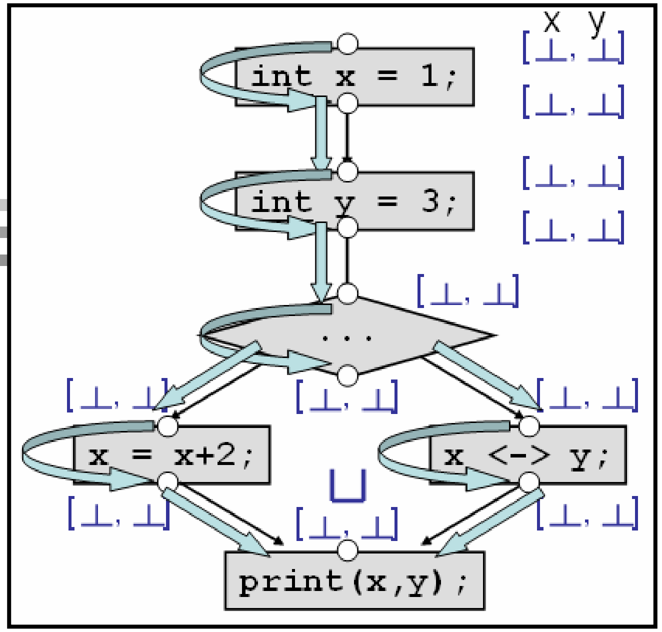
if (...) {
  x = x+2;
} else {
  x <-> y;
}
print(x,y);
```

$\lambda E\,.$
$E[x \mapsto E(x) \oplus 2]$

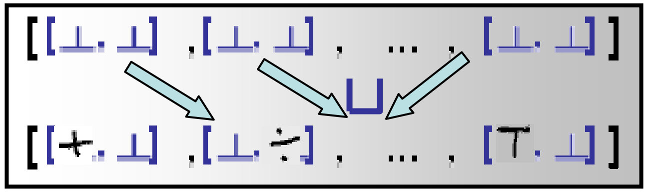




⊢ Solve Equations :-)



- One ***big*** lattice:
  - E.g., $(L^{|VAR|})^{|PP|}$
- 1 ***big*** abstract value vector:
  - $[\ [\perp, \perp]\ , [\perp, \perp]\ ,\ \ldots\ ,\ [\perp, \perp]\ ] \in (L^{|VAR|})^{|PP|}$
- 1 ***big*** transfer function:
  - $\mathbf{F} : (L^{|VAR|})^{|PP|} \rightarrow (L^{|VAR|})^{|PP|}$

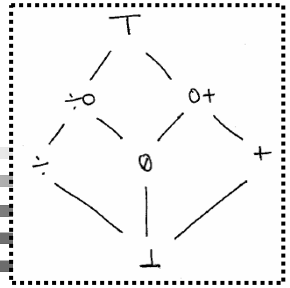


- Compute fixed-point (simply):
  - Start with bottom value vector ($\perp_{(L^{|VAR|})^{|PP|}}$)
  - Iterate transfer function '**F**' (until nothing changes)
  - Done; print out (or use) solution…! **:-)**

⊢ Agenda

- Relations:
  - Crossproducts, powersets, and relations
- Lattices:
  - Partial-Orders, least-upper-bound, and lattices
- Monotone Functions:
  - Monotone Functions and Transfer Functions
- Fixed Points:
  - Fixed Points and Solving Recursive Equations
- Putting it all together…:
  - Example revisited

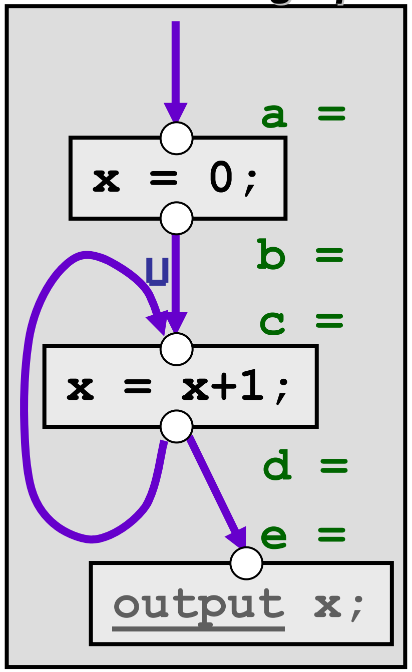
⊢ The Entire Process :-)



Program:

```
x = 0;
do {
  x = x+1;
} while (…);
output x;
```

1. Control-flow graph:



2. Transfer functions:

$f_{x=0}(\ell) = \emptyset$

$f_{x=x+1}(\ell) = \ell \oplus_L +$

3. Recursive equations:

$a = \perp$

$b = f_{x=0}(a)$

$c = b \sqcup d$

$d = f_{x=x+1}(c)$

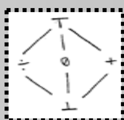
$e = d$

≀≀

4. one "big" transfer function:

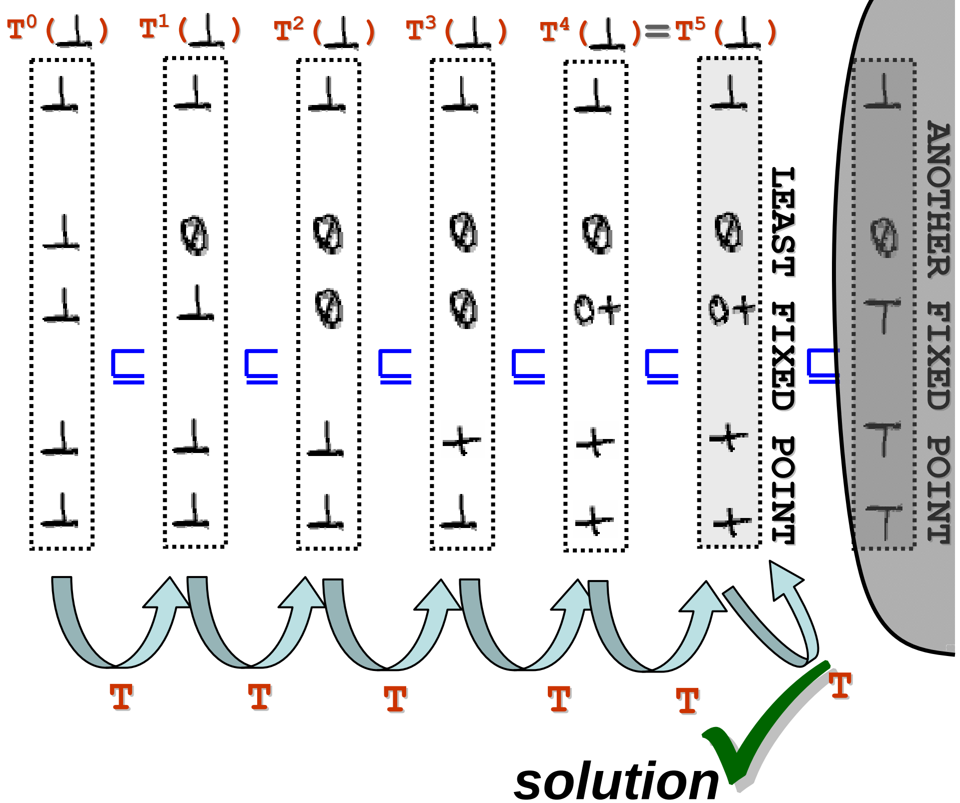
$T((a,b,c,d,e)) = (\perp, f_{x=0}(a), b \sqcup d, f_{x=x+1}(c), d)$

…over a "big" power-lattice:



|VAR||PP| = 1*5 = 5*

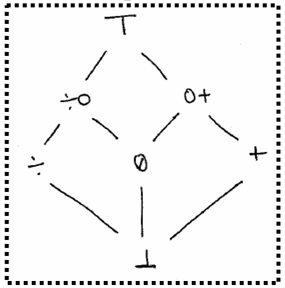
5. Solve rec. equations…:



Exercise:

- Repeat this process for program (of two vars):
  - ```
    x = 1;
    y = 0;
    while (v>w) {
      x <-> y;
    }
    y = y+1;
    ```

…using lattice:

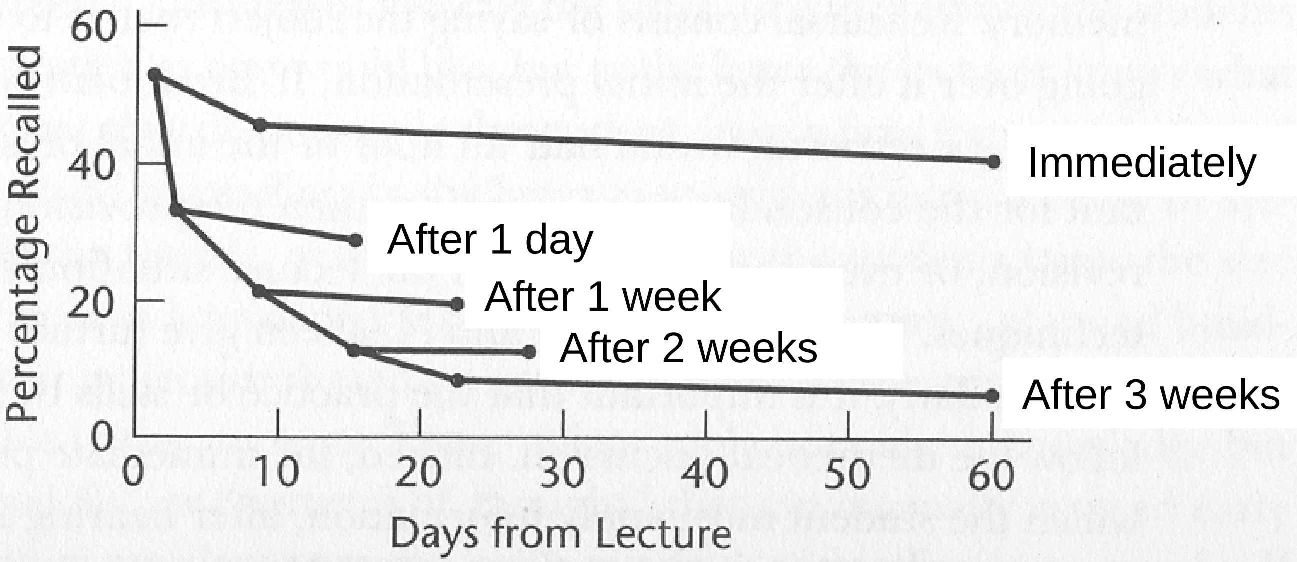


- i.e., *determine*…:
  - 1) ***Control-flow graph***
  - 2) ***Transfer functions***
  - 3) ***Recursive equations***
  - 4) ***One "big" transfer function***
  - 5) ***Solve recursive equations* :-)**
    ```

Now, please: 3’ recap

- Please spend 3' on thinking about and writing down the main ideas and points from the lecture – ***now!:***

FIGURE 2.5. THE VALUE OF REHEARSAL FOLLOWING A LECTURE.



Source: Adapted from Bassey (1968).

C