Applied Static Analysis

Three Address Code

Software Technology Group
Department of Computer Science
Technische Universität Darmstadt
Dr. Michael Eichberg

Lattice Theory

Many static analyses are based on the mathematical theory of lattices.

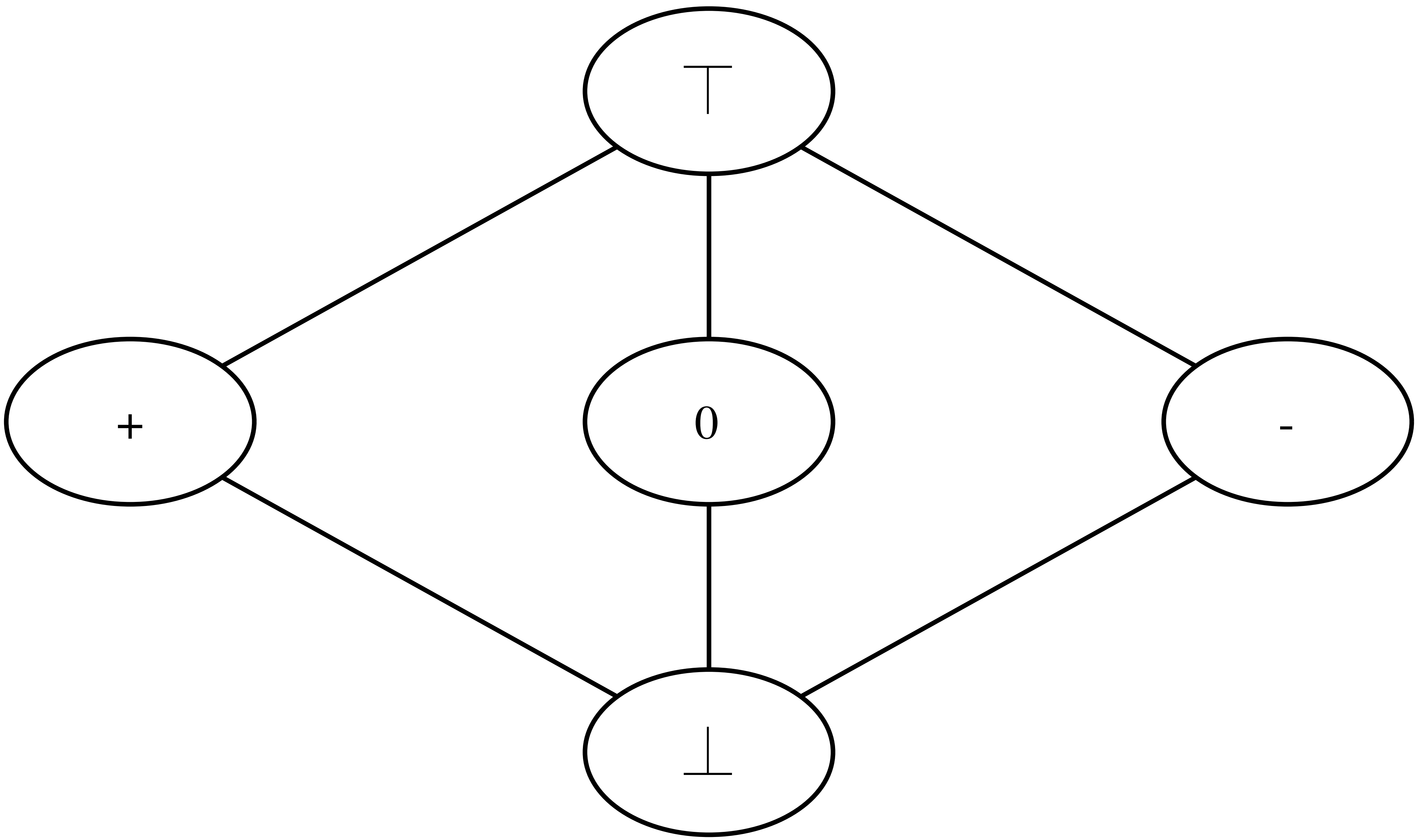
The lattice put the facts (often, but not always, sets) computed by an analysis in a well-defined partial order.

Analysis are often <u>well-defined</u> functions over lattices and can then be combined a and reasoned about.

Example: Sign Analysis

- Let's assume that we want to compute the sign of an integer value. The analysis should only return the information is definite. I.e.,
- Instead of computing with concrete values, our analysis performs it computations using abstract values:
  - positive (+)
  - negative (-)
  - zero
- Additionally, we have to add an abstract value $\top$ that represents the fact that we don't know the sign of the value.
- Values that are not initialized are represented using $\perp$.

Example: Sign Analysis - the lattice



Example: Sign Analysis - example program

```
def select(c : Boolean): Int = {
    val a = 42
    val b = 333
    var x = 0;
    if (c)
        x = a + b;
    else
        x = a - b;
    x
}
```

Partial Orderings

- a partial ordering is a relation $\sqsubseteq : L \times L \rightarrow \{true, false\}$, which
  - is reflexiv: $\forall l : l \sqsubseteq l$
  - is transitive: $\forall l_1, l_2, l_3 : l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3$
  - is anti-symmetric: $\forall l_1, l_2 : l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_1 \Rightarrow l_1 = l_2$
- a partially ordered set $(L, \sqsubseteq)$ is a set L equipped with a partial ordering $\sqsubseteq$

Upper Bounds

- for $Y \subseteq L$ and $l \in L$
  - l is an upper bound of Y, if $\forall l' \in Y : l' \sqsubseteq l$
  - l is a least upper bound of Y, if $l \sqsubseteq l_0$ whenever l_0 is also an upper bound of Y
  - if a least upper bound exists, it is unique ($\sqsubseteq$ is anti-symmetric)
  - the least upper bound of Y is denoted $\bigsqcup Y$ we write: $l1 \sqcup l2$ for $\bigsqcup \{l1, l2\}$

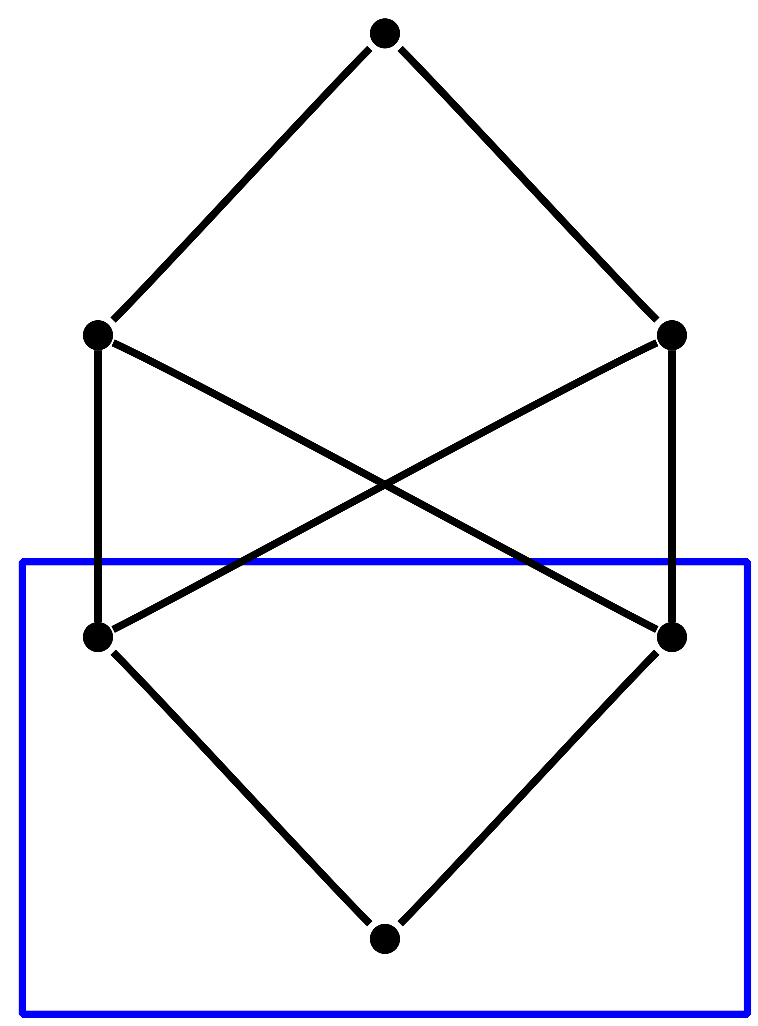
Lower Bounds

- for $Y \subseteq L$ and $l \in L$
  - l is a lower bound of Y, if $\forall l' \in Y : l \sqsubseteq l'$
  - l is a greatest lower bound of Y, if $l_0 \sqsubseteq l$ whenerver l_0 is also a lower bound of Y
  - if a greatest lower bound exists, it is unique ($\sqsubseteq$ is anti-symmetric)
  - the greatest lower bound of Y is denoted $\bigsqcap Y$

  we write: $l1 \sqcap l2$ for $\bigsqcap\{l1, l2\}$

Upper/Lower Bounds

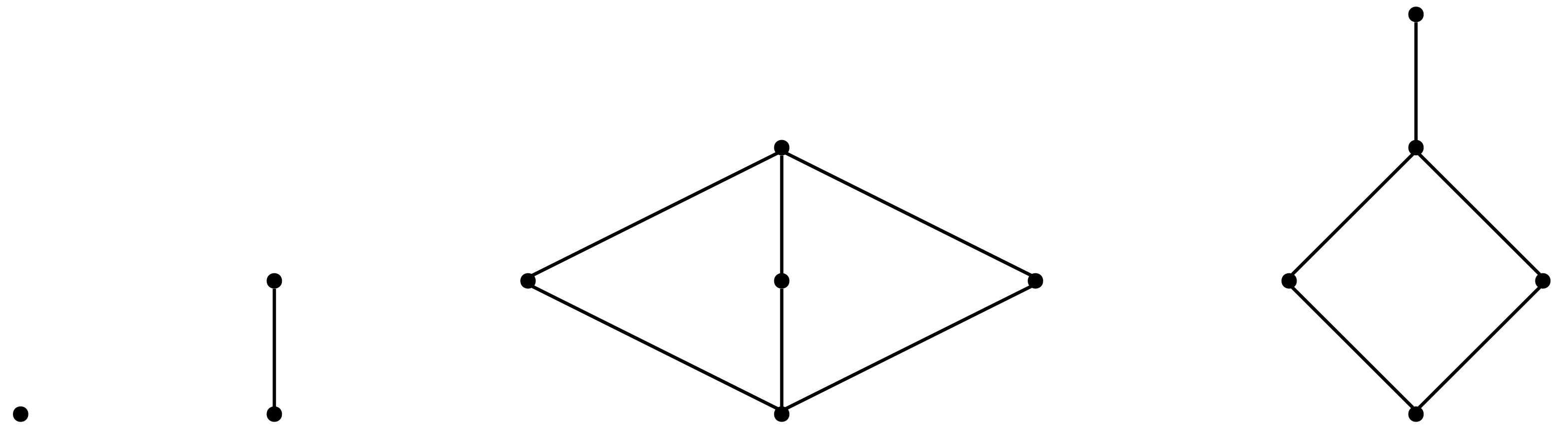
A subset Y of a partially ordered set L need not have least upper or greatest lower bounds.



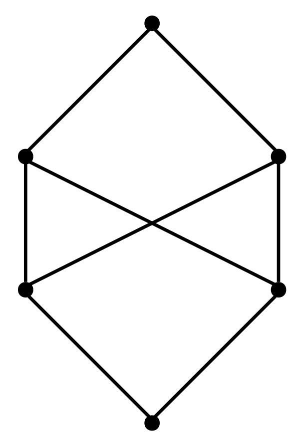
(complete) Lattice

- complete Lattice $L = (L, \sqsubseteq, \bigsqcap, \bigsqcup, \top, \bot)$
- is a partially ordered set $(L, \sqsubseteq)$ such that each subset Y has a greatest lower bound and a least upper bound.
  - $\bot = \bigsqcup \emptyset = \bigsqcap L$
  - $\top = \bigsqcap \emptyset = \bigsqcup L$

Valid lattices:

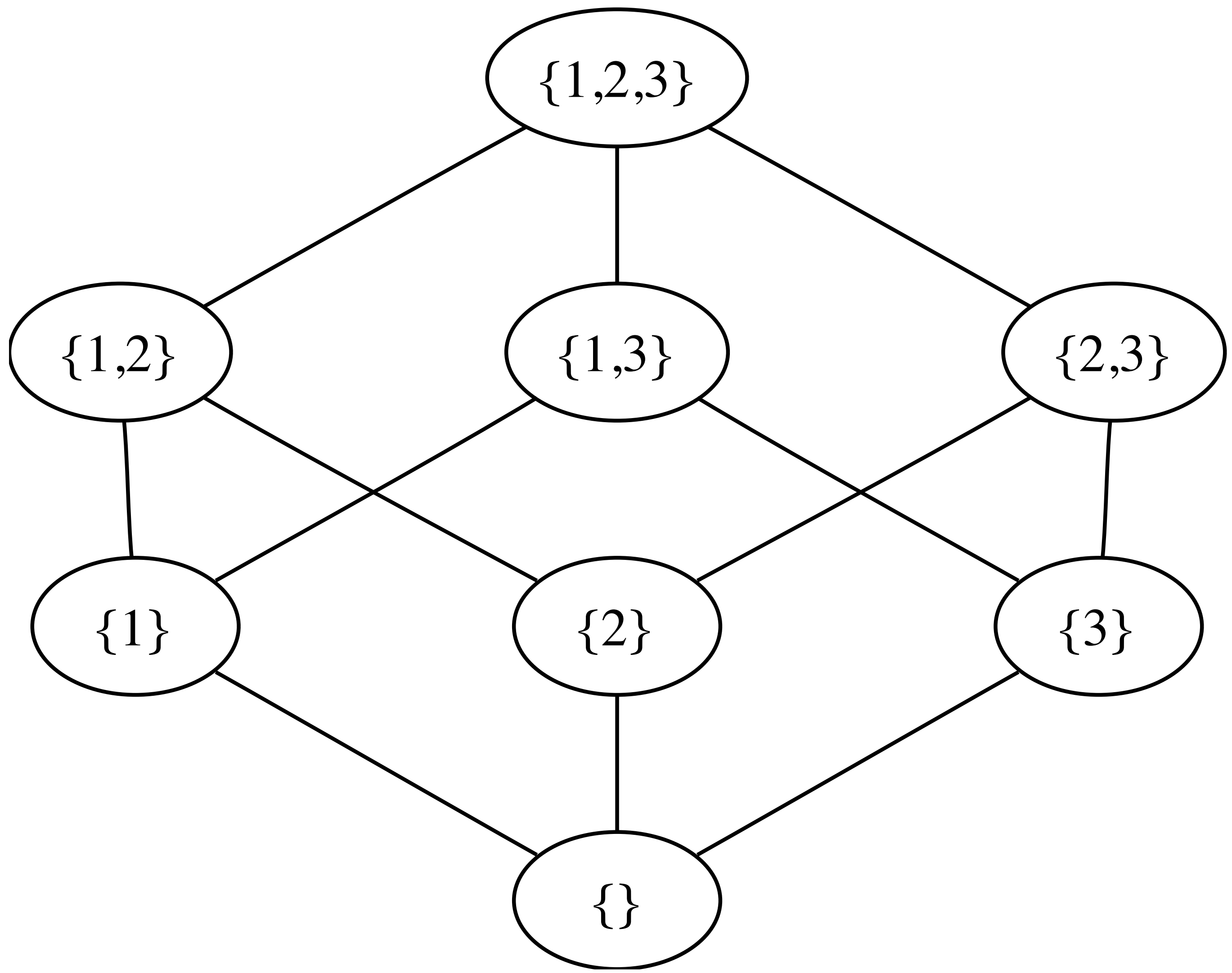


No lattice:



(complete) Lattice - example

Example $(\mathcal{P}(S), \subseteq)$, $S = \{1, 2, 3\}$



Height of a lattice

The length of the longest path from $\perp$ to $\top$.

In general, the powerset lattice has height $|S|$.

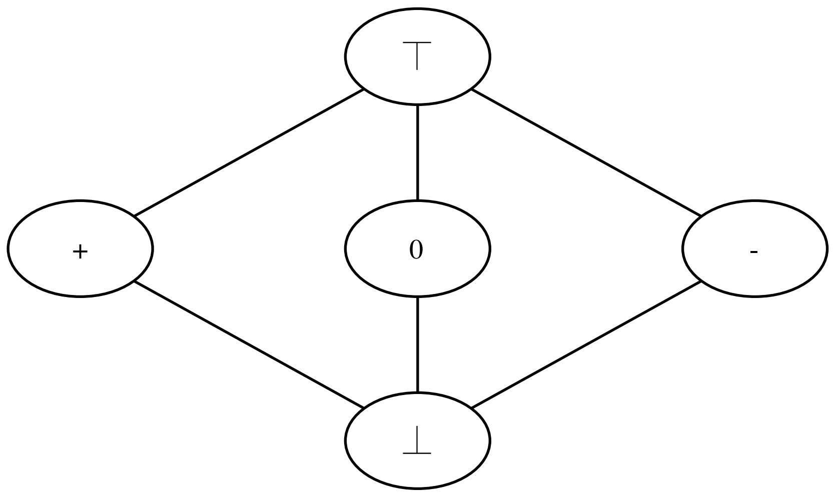
Closure Properties

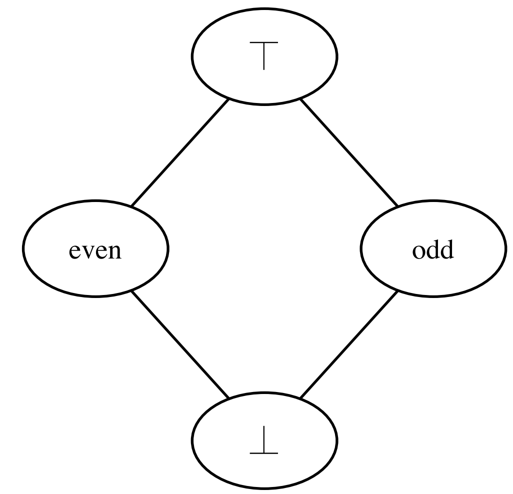
If $L_1, L_2, \ldots, L_n$ are lattices with finite height, then so is the (cartesian) product:

$$L_1 \times L_2 \times \cdots \times L_n = \{(x_1, x_2, \ldots, x_n) | X_i \in L_i\}$$

$$height(L_1 \times \cdots \times L_n) = height(L_1) + \cdots + height(L_n)$$

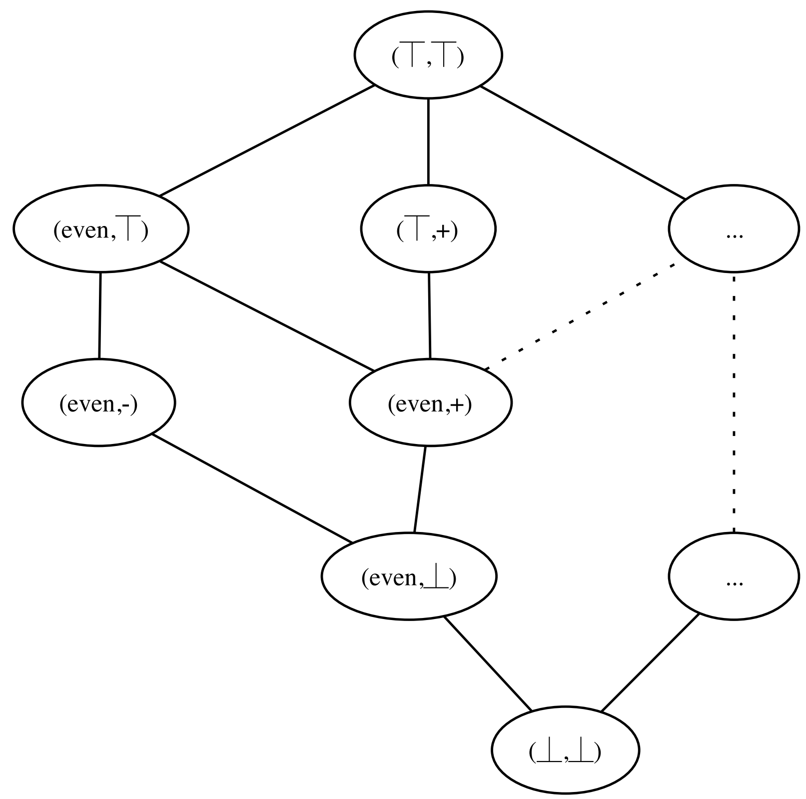
Two basic domains





Creating the cross-product

Creating the cross product of the sign and even-odd lattices.



Properties of Functions

A function $f : L_1 \rightarrow L_2$ between partially ordered sets is <u>monotone</u> if:

$$\forall l, l' \in L_1 : l \sqsubseteq_1 l' \Rightarrow f(l) \sqsubseteq_2 f(l')$$

The function f is <u>distributiv</u> if:

$$\forall l_1, l_2 \in L_1 : f(l_1 \sqcup l_2) = f(l_1) \sqcup f(l_2)$$

Chains

A subset $Y \subseteq L$ of a partially ordered set $L = (L, \sqsubseteq)$ is a chain if
$\forall l_1, l_2 \in Y : (l_1 \sqsubseteq l_2) \vee (l_2 \sqsubseteq l_1)$

The chain is finite if Y is a finite subset of L.

A sequence $(l_n)_{n \in N}$ of elements in L is an ascending chain if
$n \leq m \Rightarrow l_n \sqsubseteq l_m$

A sequence $(l_n)_n$ eventually stabilizes iff
$\exists n_0 \in N : \forall n \in N : n \geq n_0 \Rightarrow l_n = l_{n_0}$

Ascending/Descending Chain Condition

- A partially ordered set L satisfies the Ascending Chain Condition if and only if all ascending chains eventually stabilize.
- A partially ordered set L satisfies the Descending Chain Condition if and only if all descending chains eventually stabilize.

Fixed Point

- $l \in L$ is a fixed point for f if $f(l) = l$
- A least fixed point $l_1 \in L$ for f is a fixed point for f where $l_1 \sqsubseteq l_2$ for every fixed point $l_2 \in L$ for f.
- In a lattice L with finite height, every monotone function f has a unique least fixed point.

Data-flow analysis: Available Expressions

Determine for each program point, which expressions must have already been computed and not later modified on all paths to the program point.

Available Expressions - Example

```
          def m(initialA: Int, b: Int): Int = {
/*pc 0*/   var a = initialA // a has to be variable
/*pc 1*/   var x = a + b;
/*pc 2*/   val y = a * b;
/*pc 3*/   while (y > a + b) {
/*pc 4*/     a = a + 1
/*pc 5*/     x = a + b
           }
/*pc 6*/   a + x
          }
```

Available Expressions - gen/kill functions

- An <u>expression is killed</u> in a block if any of the variables used in the (arithmetic) expression are modified in the block. The function $\boldsymbol{kill : Block \to \mathcal{P}(ArithExp)}$ produces the set of killed arithmetic expressions.
- A <u>generated expression</u> is a non-trivial (arithmetic) expression that is evaluated in the block and where none of the variables used in the expression are later modified in the block. The function $\boldsymbol{gen : Block \to \mathcal{P}(ArithExp)}$ produces the set of generated expressions.

Available Expressions - data flow equations

Let $\boldsymbol{S}$ be our program and $\boldsymbol{flow}$ be a flow in the program between two statements $(\boldsymbol{pc_i}, \boldsymbol{pc_j})$.

$$AE_{entry}(pc_i) = \begin{cases} \emptyset & \text{if } i = 0 \\ \bigcap\{AE_{exit}(pc_h) | (pc_h, pc_i) \in flow(S)\} & otherwise \end{cases}$$

$$AE_{exit}(pc_i) = (AE_{entry}(pc_i) \backslash kill(block(pc_i)) \cup gen(block(pc_i)))$$

Available Expressions - Example continued

$$
\begin{aligned}
AE_{entry}(pc_1) &= \emptyset \\
AE_{entry}(pc_2) &= AE_{exit}(pc_1) \\
AE_{entry}(pc_3) &= AE_{exit}(pc_2) \cap AE_{exit}(pc_5) \\
AE_{entry}(pc_4) &= \emptyset \\
AE_{entry}(pc_5) &= \emptyset
\end{aligned}
$$