Design and Analysis of Real-Time Systems
Foundations of Abstract Interpretation I

Jan Reineke

Advanced Lecture, Summer 2013
Recap: Value Analysis

Determines invariants on values of registers at different program points. Invariants are often in the form of enclosing intervals of all possible values.

Where is this information used?
- Microarchitectural Analysis
  - Pipeline Analysis
  - Cache Analysis
- Control-Flow Analysis
  - Detect infeasible paths
  - Derive loop bounds
Value Analysis
Intuition of Interval Analysis

Can be formalized as Abstract Interpretation. ➔ Yields soundness and termination guarantees.

R1 = [-infty, +infty]
R2 = [-infty, +infty]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [42, 43]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]

R1 = [0, 41]
R2 = [42, 42]

R1 = [0, 43]
R2 = [42, 42]

R1 = [0, 4]
R2 = [42, 42]

R1 = [0, 6]
R2 = [42, 42]
Abstract Interpretation

- **Semantics-based** approach to program analysis
- Framework to develop **provably correct** and **terminating** analyses

**Ingredients:**
- **Concrete semantics**: Formalizes meaning of a program
- **Abstract semantics**
- Both semantics defined as **fixpoints of monotone functions** over some **domain**
- Relation between the two semantics establishing correctness
1. INTRODUCTION

$$\text{WCET} \quad H(P) := \max_{i \in \text{Inputs}} \max_{h \in \text{States}(H)} \text{ET}(P, i, h)$$

subject to

$$Ax \leq b$$ and $$x \geq 0$$

What is a CFG?

$$\text{CFG} \quad G = (V, E, \text{start}, \text{labeling})$$

$\text{start} \in V$

$E \subseteq V \times V$

$\text{labeling} : E \rightarrow \text{Statement}$
Four Kinds of Statements

1. Assignment: \( R = e \)
2. Load: \( R = M[e] \)
3. Store: \( M[e_1] = e_2 \)
4. Test: \( \text{Pos}(e) \) or \( \text{Neg}(e) \)
Meaning of Statements

States consist of variables and memory:

\[ s = (\rho, \mu) \in States \]

- \( \rho : Vars \rightarrow \text{int} \) (Values of Variables)
- \( \mu : \mathbb{N} \rightarrow \text{int} \) (Contents of Memory)

Execution of a statement transforms states:

\[ \llbracket \text{statement} \rrbracket \subseteq States \times States \]
States consist of variables and memory:

\[ s = (\rho, \mu) \in \text{States} \]
\[ \rho : \text{Vars} \rightarrow \text{int} \] \text{Values of Variables}
\[ \mu : \mathbb{N} \rightarrow \text{int} \] \text{Contents of Memory}

Execution of a statement transforms states:

\[ \llbracket \text{statement} \rrbracket \subseteq \text{States} \times \text{States} \]
\[ \llbracket R = e \rrbracket := \{((\rho, \mu), (\rho[R \rightarrow [e]_\rho], \mu)) \mid (\rho, \mu) \in \text{States} \} \]
States consist of variables and memory:

\[ s = (\rho, \mu) \in \text{States} \]

\[ \rho : \text{Vars} \rightarrow \text{int} \]

\[ \mu : \mathbb{N} \rightarrow \text{int} \]

Execution of a statement transforms states:

\[ \llbracket \text{statement} \rrbracket \subseteq \text{States} \times \text{States} \]

\[ \llbracket R = e \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \llbracket e \rrbracket \rho], \mu)) \mid (\rho, \mu) \in \text{States}\} \]

\[ \llbracket R = M[e] \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \mu(\llbracket e \rrbracket \rho)], \mu)) \mid (\rho, \mu) \in \text{States}\} \]
Meaning of Statements

States consist of variables and memory:

\[ s = (\rho, \mu) \in \text{States} \]

\[ \rho : \text{Vars} \rightarrow \text{int} \]

\[ \mu : \mathbb{N} \rightarrow \text{int} \]

**Execution of a statement** transforms states:

\[
\llbracket \text{statement} \rrbracket \subseteq \text{States} \times \text{States}
\]

\[
\llbracket R = e \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \llbracket e \rrbracket \rho], \mu)) \mid (\rho, \mu) \in \text{States}\}
\]

\[
\llbracket R = M[e] \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \mu(\llbracket e \rrbracket \rho)], \mu)) \mid (\rho, \mu) \in \text{States}\}
\]

\[
\llbracket M[e_1] = e_2 \rrbracket := \{((\rho, \mu), (\rho, \mu[\llbracket e_1 \rrbracket \rho \mapsto \llbracket e_2 \rrbracket \rho])) \mid (\rho, \mu) \in \text{States}\}
\]
**Meaning of Statements**

**States** consist of variables and memory:

\[ s = (\rho, \mu) \in States \]

\( \rho : Vars \rightarrow \text{int} \)

\( \mu : \mathbb{N} \rightarrow \text{int} \)

**Execution of a statement transforms states:**

\[ \llbracket \text{statement} \rrbracket \subseteq States \times States \]

\[ \llbracket R = e \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \llbracket e \rrbracket \rho], \mu)) \mid (\rho, \mu) \in States\} \]

\[ \llbracket R = M[e] \rrbracket := \{((\rho, \mu), (\rho[R \mapsto \mu(\llbracket e \rrbracket \rho)], \mu)) \mid (\rho, \mu) \in States\} \]

\[ \llbracket M[e_1] = e_2 \rrbracket := \{((\rho, \mu), (\rho, \mu[\llbracket e_1 \rrbracket \rho \mapsto \llbracket e_2 \rrbracket \rho])) \mid (\rho, \mu) \in States\} \]

\[ \llbracket Pos(e) \rrbracket := \{((\rho, \mu), (\rho, \mu)) \mid (\rho, \mu) \in States \land \llbracket e \rrbracket \rho \neq 0\} \]
**Meaning of Statements**

**States** consist of variables and memory:

\[ s = (\rho, \mu) \in States \]

\[ \rho : Vars \rightarrow \text{int} \]

\[ \mu : \mathbb{N} \rightarrow \text{int} \]

**Execution of a statement** transforms states:

\[ [\text{statement}] \subseteq States \times States \]

\[ [R = e] := \{((\rho, \mu), (\rho[R \mapsto [e]\rho], \mu)) | (\rho, \mu) \in States\} \]

\[ [R = M[e]] := \{((\rho, \mu), (\rho[R \mapsto \mu([e]\rho)], \mu)) | (\rho, \mu) \in States\} \]

\[ [M[e_1] = e_2] := \{((\rho, \mu), (\rho, \mu[[e_1]\rho \mapsto [e_2]\rho])) | (\rho, \mu) \in States\} \]

\[ [\text{Pos}(e)] := \{((\rho, \mu), (\rho, \mu)) | (\rho, \mu) \in States \land [e]\rho \neq 0\} \]

\[ [\text{Neg}(e)] := \{((\rho, \mu), (\rho, \mu)) | (\rho, \mu) \in States \land [e]\rho = 0\} \]
Meaning of Expressions

Evaluation of expressions is as expected:

\[
[a] \rho := \rho(a) \quad \text{if } a \in Vars
\]

\[
[e_1 \otimes e_2] \rho := [e_1] \rho \otimes [e_2] \rho
\]

\[
[a < b] \rho := \begin{cases} 
1 & : [a] \rho < [b] \rho \\
0 & : otherwise
\end{cases}
\]
Concrete Semantics

Different semantics are required for different properties:

- “Is there an execution in which the value of x alternates between 3 and 5?” ➔ Trace Semantics
- “Is the final value of x always the same as the initial value of x?” ➔ “Input/Output” Semantics
- “May x ever assume the value 45 at program point 7?” ➔ Reachability Semantics
Concrete Semantics

- **Trace Semantics**: Captures set of traces of states that the program may execute.
- **Input/Output Semantics**: Captures the pairs of initial and final states of execution traces.
  - Abstraction of Trace Semantics
- **Reachability Semantics**: Captures the set of reachable states at each program point
  - Abstraction of Trace Semantics
Reachability Semantics

Captures the set of reachable states at each program point. Formally: $\text{Reach} : V \rightarrow \mathcal{P}(\text{States})$

Example:
Reachability Semantics

Captures the set of reachable states at each program point. Formally: \( \text{Reach} : V \rightarrow \mathcal{P}(\text{States}) \)

Example:

\[ x \in \{\ldots, -2, -1, 0, 1, 2, \ldots\} \]

\[ x = 0 \]

\[ 1 \]

\[ \text{Pos}(x < 100) \]

\[ x = x + 1 \]

\[ 2 \]

\[ 3 \]

\[ \text{Neg}(x < 100) \]
Reachability Semantics

Captures the set of reachable states at each program point. Formally: \( \text{Reach} : V \rightarrow \mathcal{P}(\text{States}) \)

Example:

```
Example:
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
Reachability Semantics

Captures the set of reachable states at each program point. Formally: \( \text{Reach} : V \rightarrow \mathcal{P}(\text{States}) \)

Example:

```
start
x \in \{\ldots, -2, -1, 0, 1, 2, \ldots\}

x = 0
x \in \{\ldots, -2, -1, 0, 1, 2, \ldots\}

1
x = 0
x \in \{0, \ldots, 100\}

2
Pos(x < 100)

3
x = x+1

Neg(x < 100)

x \in \{100\}
```

Evaluation of Expressions:

```latex
[a] \prec \bullet := \prec (a) \text{ if } a \in \text{Vars}
[e_1 \implies e_2] \prec := [e_1] \prec \implies [e_2] \prec
[a < b] \prec := \begin{cases} 1: [a] \prec < [b] \prec & 0: \text{otherwise} \end{cases}
```
Reachability Semantics

Can be captured as the least solution of:

\[ \text{Reach}(\text{start}) = \text{States} \]

\[ \forall v' \in V \setminus \{\text{start}\} : \text{Reach}(v') = \bigcup_{v \in V, (v, v') \in E} [\text{labeling}(v, v')]\text{(Reach}(v)) \]

\[ \text{Reach}(1) = \left[\text{labeling}(\text{start}, 1)\right]\text{(Reach}(\text{start})) \cup \left[\text{labeling}(2, 1)\right]\text{(Reach}(2)) \]
\[ \text{Reach}(2) = \left[\text{labeling}(1, 2)\right]\text{(Reach}(1)) \]
\[ \text{Reach}(3) = \left[\text{labeling}(1, 3)\right]\text{(Reach}(1)) \]
Reachability Semantics

Can be captured as the **least solution** of:

\[
Reach(\text{start}) = \text{States} \\
\forall v' \in V \setminus \{\text{start}\} : Reach(v') = \bigcup_{v \in V, (v, v') \in E} \llbracket \text{labeling}(v, v') \rrbracket (Reach(v))
\]

\[
Reach(1) = \llbracket \text{labeling}(\text{start}, 1) \rrbracket (Reach(\text{start})) \cup \llbracket \text{labeling}(2, 1) \rrbracket (Reach(2)) \\
Reach(2) = \llbracket \text{labeling}(1, 2) \rrbracket (Reach(1)) \\
Reach(3) = \llbracket \text{labeling}(1, 2) \rrbracket (Reach(1))
\]

Reach(1) = \llbracket x = 0 \rrbracket (Reach(\text{start})) \cup \llbracket x = x + 1 \rrbracket (Reach(2))

Reach(2) = \llbracket Pos(x < 100) \rrbracket (Reach(1))

Reach(3) = \llbracket Neg(x < 100) \rrbracket (Reach(1))
Reachability Semantics

Can be captured as the least solution of:

\[
Reach(start) = States \\
\forall v' \in V \setminus \{start\} : Reach(v') = \bigcup_{v \in V, (v, v') \in E} \langle labeling(v, v') \rangle(Reach(v))
\]

\begin{align*}
Reach(1) &= [labeling(start, 1)](Reach(start)) \cup [labeling(2, 1)](Reach(2)) \\
Reach(2) &= [labeling(1, 2)](Reach(1)) \\
 Reach(3) &= [labeling(1, 3)](Reach(1))
\end{align*}

\begin{align*}
Reach(1) &= [x = 0](Reach(start)) \cup [x = x + 1](Reach(2)) \\
Reach(2) &= [Pos(x < 100)](Reach(1)) \\
Reach(3) &= [Neg(x < 100)](Reach(1))
\end{align*}

\begin{align*}
Reach(1) &= \{0\} \cup \{v + 1 \mid v \in Reach(2)\} \\
Reach(2) &= Reach(1) \cap \{\ldots, 98, 99\} \\
Reach(3) &= Reach(1) \cap \{100, 101, \ldots\}
\end{align*}
Questions

- Why the least solution?
- Is there more than one solution?
- Is there a unique least solution?
- Can we systematically compute it?
Answers

- Is there more than one solution? Yes!
- Is there a unique least solution? Yes!
- Can we systematically compute it? Yes and No.

```
Pos(x < 100) -> 2
x = x + 1
```

```
1 -> Neg(x < 100) -> 3
```

```
start
x = 0
```
Why? Knaster-Tarski Fixpoint Theorem!

**Theorem 1** (Knaster-Tarski, 1955). Assume \((D, \leq)\) is a complete lattice. Then every monotonic function \(f : D \rightarrow D\) has a least fixed point \(d_0 \in D\).

Raises more questions:
- What is a complete lattice?
- What is a monotonic function?
- What is a fixed point?
Monotone Functions

Let \((D, \leq)\) be partially-ordered set.
For example: \(D = \mathbb{N}\) and \(\leq\) the order on natural numbers.

**Function** \(f : D \rightarrow D\) is monotone (order-preserving) iff
for all \(d_1, d_2 \in D\): \(d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2)\).
Monotone Functions

Let \((D, \leq)\) be partially-ordered set. For example: \(D = \mathbb{N}\) and \(\leq\) the order on natural numbers.

Function \(f : D \rightarrow D\) is monotone (order-preserving) iff for all \(d_1, d_2 \in D\): \(d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2)\).

Examples:
\[
\begin{align*}
    f(x) &= x \\
    g(x) &= -x \\
    h(x) &= x - 1 \\
    F(X) &= \{ f(x) \mid x \in X \} \\
    G(X) &= \{ y \mid x \in X \land (x, y) \in R \}
\end{align*}
\]

Which of these are monotone?
Monotone Functions

Let \((D, \leq)\) be partially-ordered set. For example: \(D = \mathbb{N}\) and \(\leq\) the order on natural numbers.

Function \(f : D \rightarrow D\) is monotone (order-preserving) iff
for all \(d_1, d_2 \in D: d_1 \leq d_2 \Rightarrow f(d_1) \leq f(d_2)\).

Examples:

\[
\begin{align*}
f(x) & = x \\
g(x) & = -x \\
h(x) & = x - 1
\end{align*}
\]

Which of these are monotone?

\[
\begin{align*}
F(X) & = \{ f(x) \mid x \in X \} \\
G(X) & = \{ y \mid x \in X \land (x, y) \in R \}
\end{align*}
\]

Need to know what the order is.
Partial Orders

A binary relation $\leq : D \times D$ is a partial order, iff for all $a, b, c \in D$, we have that:

- $a \leq a$ (reflexivity),
- if $a \leq b$ and $b \leq a$ then $a = b$ (antisymmetry),
- if $a \leq b$ and $b \leq c$ then $a \leq c$ (transitivity).

A set with a partial order is called a partially-ordered set.
Partial Orders: Examples I

- The natural numbers ordered by the standard less-than-or-equal relation: \((\mathbb{N}, \leq)\).
- The set of subsets of a given set (its powerset) ordered by the subset relation: \((\mathcal{P}(A), \subseteq)\).
- The set of subsets of a given set (its powerset) ordered by the superset relation: \((\mathcal{P}(A), \supseteq)\).
- The natural numbers ordered by divisibility: \((\mathbb{N}, |)\).
Partial Orders: Examples II

The vertex set $V$ of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

The vertex set $V$ of an arbitrary graph $G = (V, E)$ ordered by reachability.

For a set $X$ and a partially-ordered set $P$, the function space $F : X \rightarrow P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all $x$ in $X$. 
Partial Orders: Examples II

The vertex set $V$ of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

The vertex set $V$ of an arbitrary graph $G = (V, E)$ ordered by reachability.

For a set $X$ and a partially-ordered set $P$, the function space $F : X \rightarrow P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all $x$ in $X$. 
Partial Orders: Examples II

The vertex set $V$ of a directed acyclic graph $G = (V, E)$ ordered by reachability (reflexive, transitive closure of edge relation).

The vertex set $V$ of an arbitrary graph $G = (V, E)$ ordered by reachability.

For a set $X$ and a partially-ordered set $P$, the function space $F : X \rightarrow P$, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all $x$ in $X$.

*What about $\text{Reach} : V \rightarrow \mathcal{P}(\text{States})$?*
Complete Lattices

A partially-ordered set \((L, \leq)\) is a \emph{complete lattice} if every subset \(A\) of \(L\) has both a \emph{least upper bound} (denoted \(\bigvee A\)) and a \emph{greatest lower bound} (denoted \(\bigwedge A\)).
Complete Lattices

A partially-ordered set \((L, \leq)\) is a complete lattice if every subset \(A\) of \(L\) has both a least upper bound (denoted \(\bigvee A\)) and a greatest lower bound (denoted \(\bigwedge A\)).

**What is an upper bound of a set \(A\)?**

An element \(x\) is an upper bound of a set \(A\) if \(x\) if for every element \(a\) of \(A\), we have \(a \leq x\).
A partially-ordered set \((L, \leq)\) is a complete lattice if every subset \(A\) of \(L\) has both a least upper bound (denoted \(\sqcup A\)) and a greatest lower bound (denoted \(\sqcap A\)).

**What is an upper bound of a set \(A\)?**

An element \(x\) is an upper bound of a set \(A\) if for every element \(a\) of \(A\), we have \(a \leq x\).

**What is the least upper bound (also: join, supremum) of a set \(A\)?**

\(x\) is the least upper bound of \(A\), denoted \(\sqcup A\), if

1. \(x\) is an upper bound of \(A\),
2. for every upper bound \(y\) of \(A\), we have \(x \leq y\).
Least Upper Bounds: Examples I

<table>
<thead>
<tr>
<th>Partially-ordered set ((D, \leq))</th>
<th>(A \subseteq D)</th>
<th>(\bigvee A)</th>
<th>(\bigwedge A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\mathbb{N}, \leq))</td>
<td>{1, 2, 3}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{R}, \leq))</td>
<td>({x \in \mathbb{R} \mid x &lt; 1})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{R}, \leq))</td>
<td>({x \in \mathbb{R} \mid x \leq 1})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{Q}, \leq))</td>
<td>({x \in \mathbb{Q} \mid x^2 \leq 2})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{N}, \leq))</td>
<td>({x \in \mathbb{N} \mid x \text{ is odd}})</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Least Upper Bounds: Examples I

Which of these are complete lattices?

<table>
<thead>
<tr>
<th>Partially-ordered set ((D, \leq))</th>
<th>(A \subseteq D)</th>
<th>(\bigcup A)</th>
<th>(\bigcap A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\mathbb{N}, \leq))</td>
<td>({1, 2, 3})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{R}, \leq))</td>
<td>({x \in \mathbb{R} \mid x &lt; 1})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{R}, \leq))</td>
<td>({x \in \mathbb{R} \mid x \leq 1})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{Q}, \leq))</td>
<td>({x \in \mathbb{Q} \mid x^2 \leq 2})</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{N}, \leq))</td>
<td>({x \in \mathbb{N} \mid x \text{ is odd}})</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## Least Upper Bounds: Examples II

### Examples:

- The natural numbers ordered by the standard less-than-or-equal relation: \((\mathbb{N}, \leq)\).
- The set of subsets of a given set (its powerset) ordered by the subset relation: \((\mathcal{P}(A), \subseteq)\).
- The set of subsets of a given set (its powerset) ordered by the subset relation: \((\mathcal{P}(A), \supseteq)\).
- The natural numbers ordered by divisibility: \((\mathbb{N}, |)\).
- The vertex set \(V\) of a directed acyclic graph \(G = (V, E)\) ordered by reachability (reflexive, transitive closure of edge relation).
- The vertex set \(V\) of an arbitrary graph \(G = (V, E)\) ordered by reachability.
- For a set \(X\) and a partially-ordered set \(P\), the function space \(F: X \to P\), where \(f \leq g\) if and only if \(f(x) \leq g(x)\) for all \(x\) in \(X\).

### Complete lattices:

A partially-ordered set \((L, \leq)\) is a complete lattice if every subset \(A\) of \(L\) has both a least upper bound (denoted \(\bigvee A\)) and a greatest lower bound (denoted \(\bigwedge A\)).

### Upper bound:

An element \(x\) is an upper bound of a set \(A\) if \(x \geq a\) for every element \(a\) of \(A\).

### Least upper bound:

\(x\) is the least upper bound of \(A\), denoted \(\bigvee A\), if

1. \(x\) is an upper bound of \(A\),
2. for every upper bound \(y\) of \(A\), we have \(x \leq y\).

### Examples least upper bounds:

<table>
<thead>
<tr>
<th>Partially-ordered set ((D, \leq))</th>
<th>(A \subseteq D)</th>
<th>(\bigvee A)</th>
<th>(\bigwedge A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\mathcal{P}(\mathbb{N}), \subseteq))</td>
<td>{{{1, 2}, {2, 4, 5}}}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathcal{P}(\mathbb{N}), \supseteq))</td>
<td>{{{1, 2}, {2, 4, 5}}}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{N},</td>
<td>))</td>
<td>{3, 4, 5}</td>
<td>?</td>
</tr>
<tr>
<td>((A \rightarrow \mathbb{N}, \leq))</td>
<td>{f, g, h}</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

### References
Least Upper Bounds: Examples II

<table>
<thead>
<tr>
<th>Partially-ordered set ((D, \leq))</th>
<th>(A \subseteq D)</th>
<th>(\bigcup A)</th>
<th>(\bigcap A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\mathcal{P}(\mathbb{N}), \subseteq))</td>
<td>\{{1, 2}, {2, 4, 5}}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathcal{P}(\mathbb{N}), \supseteq))</td>
<td>\{{1, 2}, {2, 4, 5}}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>((\mathbb{N},</td>
<td>))</td>
<td>{3, 4, 5}</td>
<td>?</td>
</tr>
<tr>
<td>((A \rightarrow \mathbb{N}, \leq))</td>
<td>{f, g, h}</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Which of these are complete lattices?
Properties of Complete Lattices

Every complete lattice \((D, \leq)\) has

- a least element (bottom element): \(\bot = \bigcup \emptyset\), and
- a greatest element (top element): \(\top = \bigcup D\).
Generic Lattice Constructions:
Power-set Lattice

For any set $S$, its power set $(\mathcal{P}(S), \subseteq)$ with set inclusion is a lattice:

- **“join”:** $\bigcup A = \bigcup A$
- **“meet”:** $\bigcap A = \bigcap A$
- **“top”:** $\top = S$
- **“bottom”:** $\bot = \emptyset$

**Graphical representation (Hasse diagram):**

![Diagram showing the Hasse diagram of a power-set lattice. The elements are nodes labeled with sets and the edges represent the inclusion relationship.]
Generic Lattice Constructions: Total Function Space

For any set \( S \) and complete lattice \((L, \leq_L)\), the total function space \((S \rightarrow L, \leq)\) is a complete lattice, with \( f \leq g :\iff \forall s \in S : f(s) \leq g(s)\):

- **“join”:** \( \bigvee A = \lambda s. \bigvee_{f \in A} f(s) \)
- **“meet”:** \( \bigwedge A = \lambda s. \bigwedge_{f \in A} f(s) \)
- **“top”:** \( \top = \lambda s. \top_L \)
- **“bottom”:** \( \bot = \lambda s. \bot_L \)
Generic Lattice Constructions:
Total Function Space

For any set $S$ and complete lattice $(L, \leq_L)$, the total function space $(S \rightarrow L, \leq)$ is a complete lattice, with $f \leq g :\iff \forall s \in S : f(s) \leq g(s)$:

"join": $\bigcup A = \lambda s. \bigcup_{f \in A} f(s)$

"meet": $\bigcap A = \lambda s. \bigcap_{f \in A} f(s)$

"top": $\top = \lambda s. \top_L$

"bottom": $\bot = \lambda s. \bot_L$

What about $\text{Reach} : V \rightarrow \mathcal{P}(\text{States})$?
For any set $S$ the flat lattice $(S \cup \{\bot, \top\}, \leq)$ is a complete lattice, with $a \leq b :\iff a = b \lor a = \bot \lor b = \top$.

Graphical representation (Hasse diagram) with $S = \mathbb{Z}$:
Fixed Points

A fixed point of a function \( f : D \rightarrow D \) is an element \( x \in D \) with \( x = f(x) \).

*Has multiple fixed points:*

\[
\begin{align*}
&\{1, 2, 3\} \\
&\{1, 2, 3, 4\} \\
&\{1, 2, 3, 5\} \\
&\{1, 2, 3, 4, 5\}
\end{align*}
\]
Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

**Example:**

$$f : \mathcal{P} \left( \{1, 2, 3, 4, 5\} \right) \to \mathcal{P} \left( \{1, 2, 3, 4, 5\} \right)$$

$$f(X) = \{1, 2, 3\} \cup X$$

**Has multiple fixed points:**

- $\{1, 2, 3\}$
- $\{1, 2, 3, 4\}$
- $\{1, 2, 3, 5\}$
- $\{1, 2, 3, 4, 5\}$
Fixed Points

A fixed point of a function $f : D \rightarrow D$ is an element $x \in D$ with $x = f(x)$.

**Example:**

$$f : \mathcal{P}(\{1, 2, 3, 4, 5\}) \rightarrow \mathcal{P}(\{1, 2, 3, 4, 5\})$$

$$f(X) = \{1, 2, 3\} \cup X$$

*Has multiple fixed points:*  
\{1, 2, 3\}  
\{1, 2, 3, 4\}  
\{1, 2, 3, 5\}  
\{1, 2, 3, 4, 5\}

*But a unique least fixed point.*  
\{1, 2, 3\}
Fixed Points

A fixed point of a function $f : D \to D$ is an element $x \in D$ with $x = f(x)$.

Example:

$$f : \mathcal{P}([1, 2, 3, 4, 5]) \to \mathcal{P}([1, 2, 3, 4, 5])$$

$$f(X) = \{1, 2, 3\} \cup X$$

Has multiple fixed points: $\{1, 2, 3\}$, $\{1, 2, 3, 4\}$, $\{1, 2, 3, 5\}$, $\{1, 2, 3\}$, $\{1, 2, 3, 4, 5\}$

But a unique least fixed point: $\{1, 2, 3\}$

The least fixed point $l$, denoted $\text{lfp } f$, of a function $f : D \to D$ over a lattice $(D, \leq)$, is a fixed point of $f$, such that for every fixed point $x$ of $f$: $l \leq x$. 
Knaster-Tarski Fixpoint Theorem

**Theorem 1** (Knaster-Tarski, 1955).

Assume $(D, \leq)$ is a complete lattice. Then every monotonic function $f : D \rightarrow D$ has a least fixed point $d_0 \in D$.

 Raises more questions:

- What is a complete lattice? ✓
- What is a monotonic function? ✓
- What is a fixed point? ✓
Back to the Reachability Semantics

Can be captured as the least fixed point of:

\[ \text{Reach}(\text{start}) = \text{States} \]

\[ \forall v' \in V \setminus \{\text{start}\} : \text{Reach}(v') = \bigcup_{v \in V, (v,v') \in E} \text{labeling}(v, v')(\text{Reach}(v)) \]

\[ \text{Reach}(1) = [x = 0](\text{Reach}(\text{start})) \cup [x = x + 1](\text{Reach}(2)) \]
\[ \text{Reach}(2) = [\text{Pos}(x < 100)](\text{Reach}(1)) \]
\[ \text{Reach}(3) = [\text{Neg}(x < 100)](\text{Reach}(1)) \]

\[ \text{Reach}(1) = \{0\} \cup \{v + 1 \mid v \in \text{Reach}(2)\} \]
\[ \text{Reach}(2) = \text{Reach}(1) \cap \{\ldots, 98, 99\} \]
\[ \text{Reach}(3) = \text{Reach}(1) \cap \{100, 101, \ldots\} \]