Verification of Real-Time Systems
Foundations of Abstract Interpretation

Jan Reineke

Advanced Lecture, Summer 2015
Recap: Reachability Semantics

Can be captured as the least solution of:

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

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

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

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

\[ \text{Reach}(1) = \{0\} \cup \{v + 1 \mid v \in \text{Reach}(2)\} \]
\[ \text{Reach}(2) = \text{Reach}(1) \cap \{\ldots, 98, 99\} \]
\[ \text{Reach}(3) = \text{Reach}(1) \cap \{100, 101, \ldots\} \]
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**?
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 \(\bigcup A\)) and a greatest lower bound (denoted \(\bigcap A\)).

**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 \(a \leq x\).

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

\(x\) is the least upper bound of \(A\), denoted \(\bigcup A\), 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>(\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>

Which of these are complete lattices?
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):**

![Hasse diagram of a power-set lattice](image)
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 $f \leq g :\Leftrightarrow \forall s \in S : f(s) \leq g(s)$:

“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 $\text{Reach} : V \rightarrow \mathcal{P}(\text{States})$?
Generic Lattice Constructions: Flat Lattice

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 $f : D \to D$ 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:  

$$\{1, 2, 3\}$$
$$\{1, 2, 3, 4\}$$
$$\{1, 2, 3, 5\}$$
$$\{1, 2, 3, 4, 5\}$$

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 \(f : D \to D\) 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? ✓
Can be captured as the least fixed point of:

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

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

\[
\text{Reach}(1) = \{0\} \cup \{v + 1 \mid v \in \text{Reach}(2)\} \\
\text{Reach}(2) = \text{Reach}(1) \cap \{\ldots, 98, 99\} \\
\text{Reach}(3) = \text{Reach}(1) \cap \{100, 101, \ldots\}
\]
How to Compute the Least Fixed Point

Kleene Iteration:
\[
\bot \leq f(\bot) \leq f^2(\bot) \leq f^3(\bot) \leq \ldots
\]

Why is this increasing?  
Will this reach the fixed point?  
It will here:  
But in general?  

No!  
Lattice has infinite ascending chains.
Infinite Ascending Chains

Think of an example of an infinite ascending chain.

No!
Ascending Chain Condition

A partially-ordered set $S$ satisfies the *ascending chain condition* if every strictly ascending sequence of elements is finite.

**Theorem (Ascending Chain Condition):**

Let $(S, \leq)$ be a complete lattice set that satisfies the ascending chain condition, and let $f : S \rightarrow S$ be a monotone function. Then, there is an $n \in \mathbb{N}$, such that

$$\text{lfp } f = f^n(\bot).$$

⇒ Length of longest ascending chain determines worst-case complexity of Kleene Iteration.
Ascending Chain Condition: Examples

A partially-ordered set $S$ satisfies the **ascending chain condition** if every strictly ascending sequence of elements is finite.

→ Ascending chain condition does not imply finite partially-ordered set!

How about total function space lattice?
How about finite partially-ordered sets?
Recap: 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
Abstract Semantics

Similar to concrete semantics:
- A complete lattice \((L^#, \leq)\) as the domain for abstract elements
- A monotone function \(F^#\) corresponding to the concrete function \(F\)
- Then the abstract semantics is the least fixed point of \(F^#\), \(\text{lfp}\ F^#\)

If \(F^#\) “correctly approximates” \(F\),
then \(\text{lfp}\ F^#\) “correctly approximates” \(\text{lfp}\ F\).
An Example Abstract Domain for Values of Variables

How to relate the two?

- Concretization function, specifying “meaning” of abstract values.
  \[ \gamma : \mathbb{Z}_\perp \rightarrow \mathcal{P}(\mathbb{Z}) \]

- Abstraction function: determines best representation concrete values.
  \[ \alpha : \mathcal{P}(\mathbb{Z}) \rightarrow \mathbb{Z}_\perp \]
Relation between the Abstract and Concrete Domains

\[ \begin{align*}
\gamma(\top) & := \mathbb{Z} \\
\gamma(\bot) & := \emptyset \\
\gamma(x) & := \{x\} \\
\alpha(A) & := \begin{cases} 
\top & : |A| \geq 2 \\
x & : A = \{x\} \\
\bot & : A = \emptyset 
\end{cases}
\end{align*} \]

1. Are these functions monotone?
2. Should they be?
3. What is the meaning of the partial order in the abstract domain?
4. What if we first abstract and then concretize?
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

<table>
<thead>
<tr>
<th></th>
<th>$\top$</th>
<th>$a$</th>
<th>$0$</th>
<th>$\perp$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\top$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\perp$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Denotes abstract version of operator.
How to Compute in the Abstract Domain: Correctness Conditions

**Correctness Condition:**
- Abstract Domain
  - \( \gamma \) \( \rightarrow \) \( \gamma \)
- Concrete Domain
  - \( \gamma \) \( \rightarrow \) \( \gamma \)

**Correct by construction**
*(if concretization and abstraction have certain properties)*:

- Abstract Domain
  - \( \gamma \) \( \rightarrow \) \( \gamma \)
- Concrete Domain
  - \( \gamma \) \( \rightarrow \) \( \gamma \)
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

\[
\begin{array}{cccc}
\{0\} & 0 & \rightarrow & 0 \\
\gamma & \uparrow & \# & \downarrow \\
\mathbb{Z} & \{0\} & \rightarrow & \{0\} \\
\end{array}
\]
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

\[ \begin{align*}
\{a\} \quad \{b\} \quad & \quad \rightarrow \quad \{a \ast b\} \\
\gamma \quad \gamma \quad & \quad \rightarrow \quad a \ast b
\end{align*} \]
How to Compute in the Abstract Domain: Correct by Construction

Correct by construction
(if concretization and abstraction have certain properties):

"Certain properties": Notion of Galois connections:

Let \((L, \leq)\) and \((M, \sqsubseteq)\) be partially ordered sets and \(\alpha \in L \rightarrow M, \gamma \in M \rightarrow L\). We call \((L, \leq) \rightleftharpoons (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

\[
\begin{align*}
\gamma(\alpha(l)) & \leq l \\
\alpha(\gamma(m)) & \sqsubseteq m
\end{align*}
\]

for all \(l \in L\) and \(m \in M\).
Galois connections

**Notion of Galois connections:**

Let $(L, \leq)$ and $(M, \sqsubseteq)$ be partially ordered sets and $\alpha \in L \rightarrow M$, $\gamma \in M \rightarrow L$. We call $(L, \leq) \leftrightarrow_{\alpha}^{\gamma} (M, \sqsubseteq)$ a Galois connection if $\alpha$ and $\gamma$ are monotone functions and

\[
\begin{align*}
    l & \leq \gamma(\alpha(l)) \\
    \alpha(\gamma(m)) & \sqsubseteq m
\end{align*}
\]

for all $l \in L$ and $m \in M$.

**Graphically:**

For soundness.

For precision.
Galois connections: Properties

Graphically:

Properties:
1) Can be used to systematically construct correct (and in fact the most precise) abstract operations: $op^\# = \alpha \circ op \circ \gamma$

2) a) Abstraction function induces concretization function
b) Concretization function induces abstraction function
Concrete states are not just sets of values...

Concrete states consist of variables and memory:

\[
\begin{align*}
  s &= (\rho, \mu) \in \text{States} \\
  \rho : \text{Vars} &\to \text{int} \\
  \mu : \mathbb{N} &\to \text{int}
\end{align*}
\]

\[
\text{States} = (\text{Vars} \to \mathbb{Z}) \times (\mathbb{N} \to \mathbb{Z})
\]
Abstracting Sets of Concrete States
Recap: Concrete States

Reachability semantics is defined on sets of states:

\[
\llbracket \text{statement} \rrbracket \subseteq \text{States} \times \text{States} \\
\llbracket \text{statement} \rrbracket : \mathcal{P}(\text{States}) \rightarrow \mathcal{P}(\text{States}) \\
\llbracket \text{statement} \rrbracket (S) := \{ s' \mid \exists s \in S : (s, s') \in \llbracket \text{statement} \rrbracket \} \\
\mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z}))
\]
Relation between Concrete Domain and Abstract Domain

Concrete domain!

\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]

Abstract domain?

\[ \widehat{\text{States}} = \text{Vars} \rightarrow \mathbb{Z}_\perp \]

Relation between the two?

→ For ease of understanding, introduce Intermediate domain:

\[ \overset{\text{PowerSetStates}}{\text{States}} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]
Relation between Concrete Domain and Intermediate Domain

**Concrete domain:**

\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]

**Intermediate domain:**

\[ \overline{\text{PowerSetStates}} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]

**Abstraction:**

\[ \alpha_{C,I} : \mathcal{P}(\text{States}) \rightarrow \overline{\text{PowerSetStates}} \]

\[ \alpha_{C,I}(C) := \lambda x \in \text{Vars}.\{v(x) \in \mathbb{Z} \mid (v,m) \in C\} \]

**Concretization:**

\[ \gamma_{I,C} : \overline{\text{PowerSetStates}} \rightarrow \mathcal{P}(\text{States}) \]

\[ \gamma_{I,C}(\widehat{C}) := \{(v,m) \in \text{States} \mid \forall x \in \text{Vars} : v(x) \in \widehat{c}(x)\} \]
Relation between Intermediate Domain and Abstract Domain

**Intermediate domain:**

\[ \text{PowerSetStates} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]

**Abstract domain:**

\[ \text{States} = \text{Vars} \rightarrow \mathbb{Z}^{\perp} \]

**Abstraction:**

\[ \alpha_{I,A} : \text{PowerSetStates} \rightarrow \text{States} \]

\[ \alpha(\widehat{c}) := \lambda x \in \text{Vars}.\alpha(c(x)) \]

**Concretization:**

\[ \gamma_{A,I} : \text{States} \rightarrow \text{PowerSetStates} \]

\[ \gamma(\widehat{a}) := \lambda x \in \text{Vars}.\gamma(\hat{a}(x)) \]
Relation between Concrete Domain and Abstract Domain

**Concrete domain:**

\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]

**Abstract domain:**

\[ \overline{\text{States}} = \overline{\text{Vars}} \rightarrow \mathbb{Z}^\perp \]

**Abstraction:**

\[ \alpha_{C,A} : \mathcal{P}(\text{States}) \rightarrow \overline{\text{States}} \]

\[ \alpha_{C,A} := \alpha_{I,A} \circ \alpha_{C,I} \]

**Concretization:**

\[ \gamma_{A,C} : \overline{\text{States}} \rightarrow \mathcal{P}(\text{States}) \]

\[ \gamma_{A,C} := \gamma_{I,C} \circ \gamma_{A,I} \]

Galois connections can be composed to obtain new Galois connections.
Meaning of Statements in the Abstract Domain

\[
\begin{align*}
[R = e]^\# (\hat{\alpha}) & := \hat{\alpha}[R \mapsto [e]^\# (\hat{\alpha})] \\
[R = M[e]]^\# (\hat{\alpha}) & := \hat{\alpha}[R \mapsto \top] \\
[M[e_1] = e_2]^\# (\hat{\alpha}) & := \hat{\alpha} \\
[Pos(e)]^\# (\hat{\alpha}) & := \hat{\alpha} \\
[Neg(e)]^\# (\hat{\alpha}) & := \hat{\alpha}
\end{align*}
\]

Can this be done better?

Again:

For Correctness:

For the best possible precision:
Evaluation of expressions is as expected:

\[
[x]^\#(\hat{\alpha}) := \hat{\alpha}(x) \quad \text{if } x \in Vars
\]

\[
[e_1 \ op \ e_2]^\#(\hat{\alpha}) := [e_1]^\#(\hat{\alpha}) \ op^\# \ [e_2]^\#(\hat{\alpha})
\]
Putting it all together: The Abstract Reachability Semantics

Abstract Reachability Semantics captured as least fixed point of:

\[
\begin{align*}
\widehat{\text{Reach}} : V & \to \text{States} \\
\widehat{\text{Reach}}(\text{start}) & = \top \\
\forall v' \in V \setminus \{\text{start}\} : \widehat{\text{Reach}}(v') & = \bigsqcup_{v \in V, (v, v') \in E} \llbracket \text{labeling}(v, v') \rrbracket \# (\widehat{\text{Reach}}(v))
\end{align*}
\]

\[
\begin{align*}
\widehat{\text{Reach}}(1) & = \llbracket \text{labeling}(\text{start}, 1) \rrbracket \# (\widehat{\text{Reach}}(\text{start})) \sqcup \llbracket \text{labeling}(2, 1) \rrbracket (\widehat{\text{Reach}}(2)) \\
\widehat{\text{Reach}}(2) & = \llbracket \text{labeling}(1, 2) \rrbracket \# (\widehat{\text{Reach}}(1)) \\
\widehat{\text{Reach}}(3) & = \llbracket \text{labeling}(1, 3) \rrbracket \# (\widehat{\text{Reach}}(1)) \\
\end{align*}
\]

\[
\begin{align*}
\widehat{\text{Reach}}(1) & = \llbracket x = 0 \rrbracket \# (\widehat{\text{Reach}}(\text{start})) \sqcup \llbracket x = x + 1 \rrbracket \# (\widehat{\text{Reach}}(2)) \\
\widehat{\text{Reach}}(2) & = \llbracket \text{Pos}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1)) \\
\widehat{\text{Reach}}(3) & = \llbracket \text{Neg}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1))
\end{align*}
\]

\[
\begin{align*}
\widehat{\text{Reach}}(1) & = \llbracket x = 0 \rrbracket \# (\widehat{\text{Reach}}(\text{start})) \sqcup \llbracket x = x + 1 \rrbracket \# (\widehat{\text{Reach}}(2)) \\
\widehat{\text{Reach}}(2) & = \llbracket \text{Pos}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1)) \\
\widehat{\text{Reach}}(3) & = \llbracket \text{Neg}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1))
\end{align*}
\]
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[
\begin{align*}
 x & \mapsto \perp \\
 x & \mapsto \perp
\end{align*}
\]

\[
\begin{align*}
 x & = 0 \\
 Pos(x < 100) & \quad 1 \\
 Neg(x < 100) & \quad 3 \\
 x & = x+1 \\
\end{align*}
\]
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[
\begin{align*}
x & \mapsto \top \\
x & \mapsto 0 \\
x & \mapsto \bot
\end{align*}
\]

\[
\begin{align*}
x & \mapsto \top \\
x & \mapsto 0 \\
x & \mapsto \bot
\end{align*}
\]

\[
\begin{align*}
x & \mapsto \top \\
x & \mapsto 0 \\
x & \mapsto \bot
\end{align*}
\]
Example II: Kleene Iteration to Compute Abstract Reachability Semantics

```c
y = 0;
x = 1;
z = 3;
while (x > 0) {
    if (x == 1) {
        y = 7;
    }
    else {
        y = z + 4;
    }
    x = 3;
    print y;
}
```

```c
start
1
2
3
4
5
6
7

x = 1
y = 0
z = 3

Neg(x > 0)
Pos(x > 0)

Neg(x == 1)
Pos(x == 1)

y = 7
y = z + 4
x = 3
print y
```
The Abstract Transformer $F#$

Local Correctness Condition:

Correct by construction
(if concretization and abstraction have certain properties):
From Local to Global Correctness: Kleene Iteration
Fixpoint Transfer Theorem

Let \((L, \leq)\) and \((L^\#, \leq^\#)\) be two lattices, \(\gamma : L^\# \to L\) a monotone function, and \(F : L \to L\) and \(F^\# : F^\# \to F^\#\) two monotone functions, with

\[
\forall l^\# \in L^\# : \gamma(F^#(l^#)) \geq F(\gamma(l^#)).
\]

Then:

\[
lfp F \leq \gamma(lfp F^#).
\]
Outlook: Other Abstractions

- Signs
- Parity
- Intervals
- Octagons
- Congruence