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?
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!
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?

Lattice has infinite ascending chains.
Infinite Ascending Chains

Think of an example of an infinite ascending chain.

No!
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

\((\mathbb{Z}_1, \leq)\) \hspace{1cm} \((\mathcal{P}(\mathbb{Z}), \subseteq)\)

\(\gamma : \mathbb{Z}_1 \rightarrow \mathcal{P}(\mathbb{Z})\)

\(\alpha : \mathcal{P}(\mathbb{Z}) \rightarrow \mathbb{Z}_1\)
An Example Abstract Domain for Values of Variables

\[(\mathbb{Z}^\perp, \leq)\]  \hspace{1cm}  \[(\mathcal{P}(\mathbb{Z}), \subseteq)\]

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

\[ \gamma(\top) \]
\[ \gamma(\bot) \]
\[ \gamma(x) \]
Relation between the Abstract and Concrete Domains

\[
\begin{align*}
\gamma(\top) & : = \mathbb{Z} \\
\gamma(\bot) & : = \emptyset \\
\gamma(x) & : = \{ x \}
\end{align*}
\]
Relation between the Abstract and Concrete Domains

\[
\begin{align*}
\gamma(\top) &:= \mathbb{Z} \\
\gamma(\bot) &:= \emptyset \\
\gamma(x) &:= \{x\}
\end{align*}
\]
Relation between the Abstract and Concrete Domains

\[\gamma(\top) := \mathbb{Z}\]
\[\gamma(\bot) := \emptyset\]
\[\gamma(x) := \{x\}\]

\[\alpha(A) := \begin{cases} 
\top : |A| \geq 2 \\
x : A = \{x\} \\
\bot : A = \emptyset
\end{cases}\]
Relation between the Abstract and Concrete Domains

\[
\begin{align*}
\gamma(\top) & := \mathbb{Z} \\
\gamma(\bot) & := \emptyset \\
\gamma(x) & := \{x\}
\end{align*}
\]

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

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 the concretize?
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

<table>
<thead>
<tr>
<th>#</th>
<th>T.</th>
<th>a</th>
<th>0</th>
<th>⊥</th>
</tr>
</thead>
<tbody>
<tr>
<td>T.</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>⊥</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

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

Correctness Condition:

Correct by construction (if concretization and abstraction have certain properties):
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice
How to Compute in the Abstract Domain Example: Multiplication on Flat Lattice

\[
\begin{array}{c}
\top \quad 0 \\
\gamma \quad \gamma \\
\mathbb{Z} \quad \{0\}
\end{array}
\]

\[
\begin{array}{c}
\gamma \\
\gamma \\
\mathbb{Z} \quad \{0\}
\end{array}
\]
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

\[
\begin{array}{ccc}
\top & 0 & \# \\
\gamma & \gamma & \ast \\
Z & \{0\} & \{0\}
\end{array}
\]
A partially-ordered set (\( \mathbb{N} \)) of natural numbers ordered by \( \leq \) is a lattice. We have a join (denoted \( \vee \)) of a directed acyclic graph \( D \), and for \( (\mathbb{Z}, \leq) \), we have an intersection (denoted \( \cap \)) of the total function space \( \mathbb{Z} \times \mathbb{F} \). By the ascending chain condition the sequence \( \langle S \rangle \) must eventually stabilize. So for some \( m \), we get \( S_m = S_{m+1} \). By monotonicity, we also get \( \gamma \leq S_{m+1} \). We show by induction over the number of elements that \( a \leq \gamma \) for some \( a \in S_{m+1} \). States and their descriptions with:

\[
\begin{align*}
Z & \rightarrow \{0\} \\
\gamma & \rightarrow \gamma \\
\gamma & \rightarrow \gamma \\
0 & \rightarrow 0
\end{align*}
\]

Proof. By the ascending chain condition the sequence \( \langle S \rangle \) is non-increasing and \( S \in \mathbb{Z} \). By monotonicity, we also get \( \gamma \leq S_{m+1} \).
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

\[ a \times b \]
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

\[
\begin{align*}
\gamma & \quad \{a\} \\
\gamma & \quad \{b\} \\
\end{align*}
\]
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

\[ a \rightarrow b \rightarrow \{ a \} \rightarrow \{ a * b \} \]

\[ a \rightarrow b \rightarrow \{ a \} \rightarrow \{ a * b \} \]
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

\[ \begin{array}{ccc}
  a & b & a \times b \\
 \gamma & \gamma & \# \\
 \{a\} & \{b\} & \{a \times b\}
\end{array} \]
How to Compute in the Abstract Domain: Correct by Construction

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

```
  (                  )
  |                | op#
  |                |
  |                |
   |   α           |
  |                |
  |                |
  |                |
```

```
  (                  )
  |                | op
  |                |
  |                |
   |   γ           |
  |                |
  |                |
  |                |
```

**“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 \((L, \leq) \leftrightarrow_{\alpha} \gamma (M, \sqsubseteq)\) 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 \(m \in M\).
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 \((L, \leq) \leftarrow_{\alpha} \rightarrow_{\gamma} (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

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

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

Graphically: \((L, \leq) \leftarrow_{\alpha} \rightarrow_{\gamma} (M, \sqsubseteq)\)
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 \((L, \leq) \text{≤} \rightarrow (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

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

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

Graphically:
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 \((L, \leq) \xleftarrow{\alpha} \xrightarrow{\gamma} (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

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

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

**Graphically:**

\[
\begin{array}{c}
\alpha \quad \gamma \\
\downarrow \quad \downarrow \\
L \quad M \\
\gamma \quad \alpha
\end{array}
\]
**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 \((L, \leq) \xleftarrow{\gamma} \xrightarrow{\alpha} (M, \sqsubseteq)\) 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:**

![Diagram showing Galois connections](image)
Galois connections: Properties

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

2) a) Abstraction function induces concretization function
   b) Concretization function induces abstraction function
Galois connections: Properties

Graphically:

\[ \begin{array}{c}
(L, \leq) \\
\gamma \\
\alpha \\
\forall l \\
\end{array} \quad \begin{array}{c}
(M, \subseteq) \\
\|l \\
\gamma \\
\alpha \\
\end{array} \]

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

2) a) Abstraction function induces concretization function
   b) Concretization function induces abstraction function

Why?
How?
Concrete states are not just sets of values...

**Concrete states** consist of variables and memory:

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

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

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

\[
States = (Vars \rightarrow \text{int}) \times (\mathbb{N} \rightarrow \text{int})
\]

\[
States = (Vars \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})
\]
Abstracting Sets of Concrete States

Recap: Concrete States

Concrete states are not just sets of values...

**Concrete states** consist of variables and memory:

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

\[ \rho : Vars \rightarrow int \]

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

\[ States = (Vars \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z}) \]
Abstracting Sets of Concrete States

Recap: Concrete States

*Reachability semantics is defined on sets of states:*

\[
\begin{align*}
\llbracket \text{statement} \rrbracket & \subseteq \text{States} \times \text{States} \\
\llbracket \text{statement} \rrbracket & : \mathcal{P}(\text{States}) \to \mathcal{P}(\text{States}) \\
\llbracket \text{statement} \rrbracket (S) & := \{ s' \mid \exists s \in S : (s, s') \in \llbracket \text{statement} \rrbracket \}
\end{align*}
\]

\[
\mathcal{P}(\text{States}) = \mathcal{P}(\left( \text{Vars} \to \mathbb{Z} \right) \times (\mathbb{N} \to \mathbb{Z}))
\]
Relation between Concrete Domain and Abstract Domain

Concrete domain!

Abstract domain?

\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]
### Relation between Concrete Domain and Abstract Domain

<table>
<thead>
<tr>
<th>Concrete domain!</th>
<th>Abstract domain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}(\text{States}) =$</td>
<td>$\widehat{\text{States}} = \text{Vars} \rightarrow \mathbb{Z}^\perp$</td>
</tr>
<tr>
<td>$\mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z}))$</td>
<td></td>
</tr>
</tbody>
</table>
Relation between Concrete Domain and Abstract Domain

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

Abstract domain?
\[ \overline{States} = \text{Vars} \rightarrow \mathbb{Z}^\uparrow \]

Relation between the two?
\[ \rightarrow \text{For ease of understanding, introduce Intermediate domain:} \]
\[ \overline{\text{PowerSetStates}} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]
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:**
\[ \text{PowerSetStates} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]

**Abstraction:**

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

**Abstraction:**
\[ \alpha_{C,I} : \mathcal{P}(\text{States}) \rightarrow \text{PowerSetStates} \]
\[ \alpha_{C,I}(C) := \lambda x \in \text{Vars}.\{v(x) \in \mathbb{Z} \mid (v, m) \in C\} \]

**Concretization:**
Relation between Concrete Domain and Intermediate Domain

Concrete domain:
\[ \mathcal{P}(\text{States}) = \mathcal{P}(\text{(Vars} \to \mathbb{Z}) \times (\mathbb{N} \to \mathbb{Z})) \]

Intermediate domain:
\[ \overset{\text{PowerSetStates}}{\mathcal{P}(\text{States})} = \text{Vars} \to \mathcal{P}(\mathbb{Z}) \]

Abstraction:
\[ \alpha_{C,I} : \mathcal{P}(\text{States}) \to \overset{\text{PowerSetStates}}{\mathcal{P}(\text{States})} \]
\[ \alpha_{C,I}(C) := \lambda x \in \text{Vars}.\{v(x) \in \mathbb{Z} \mid (v, m) \in C\} \]

Concretization:
\[ \gamma_{I,C} : \overset{\text{PowerSetStates}}{\mathcal{P}(\text{States})} \to \mathcal{P}(\text{States}) \]
\[ \gamma_{I,C}(\widehat{C}) := \{(v, m) \in \text{States} \mid \forall x \in \text{Vars} : v(x) \in \widehat{c}(x)\} \]
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:**

**Concretization:**
Relation between Intermediate Domain and Abstract Domain

**Intermediate domain:**

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

**Abstract domain:**

\[ \overset{\text{States}}{\text{Vars}} \rightarrow \mathbb{Z}^\top \]

**Abstraction:**

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

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

**Concretization:**
Relation between Intermediate Domain and Abstract Domain

**Intermediate domain:**

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

**Abstract domain:**

\[ \hat{\text{States}} = \text{Vars} \to \mathbb{Z}^\top \]

**Abstraction:**

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

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

**Concretization:**

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

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

Intermediate domain: \[ \text{PowerSetStates} = \text{Vars} \to \mathcal{P} (\mathbb{Z}) \]
Abstract domain: \[ \text{States} = \text{Vars} \to \mathbb{Z}^\top \]

Abstraction:
\[ \alpha_{I,A} : \text{PowerSetStates} \to \text{States} \]
\[ \alpha(\hat{c}) := \lambda x \in \text{Vars}. \alpha(c(x)) \]

Concretization:
\[ \gamma_{A,I} : \text{States} \to \text{PowerSetStates} \]
\[ \gamma(\hat{a}) := \lambda x \in \text{Vars}. \gamma(a(x)) \]

Abstraction and Concretization functions from before!

Could plug in other abstractions for sets of values...
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:**

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

**Abstraction:**

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

\[ \alpha_{C,A} \equiv \alpha_{I,A} \circ \alpha_{C,I} \]

**Concretization:**

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

\[ \gamma_{A,C} \equiv \gamma_{I,C} \circ \gamma_{A,I} \]
Relation between Concrete Domain and Abstract Domain

Concrete domain:
\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \to \mathbb{Z}) \times (\mathbb{N} \to \mathbb{Z})) \]

Abstract domain:
\[ \overline{\text{States}} = \text{Vars} \to \mathbb{Z}^{\top} \]

Abstraction:
\[ \alpha_{C,A} : \mathcal{P}(\text{States}) \to \overline{\text{States}} \]
\[ \alpha_{C,A} := \alpha_{I,A} \circ \alpha_{C,I} \]

Concretization:
\[ \gamma_{A,C} : \overline{\text{States}} \to \mathcal{P}(\text{States}) \]
\[ \gamma_{A,C} := \gamma_{I,C} \circ \gamma_{A,I} \]

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

\[
\begin{align*}
[R = e]^\#(\hat{\alpha}) & := \hat{\alpha}[R \mapsto [e]^\#(\hat{\alpha})] \\
[R = M[e]]^\#(\hat{\alpha}) & := \hat{\alpha}[R \mapsto \top] \\
[M[e_1] = e_2]^\#(\hat{\alpha}) & := \hat{\alpha} \\
[Pos(e)]^\#(\hat{\alpha}) & := \hat{\alpha} \\
[Neg(e)]^\#(\hat{\alpha}) & := \hat{\alpha}
\end{align*}
\]
Meaning of Statements in the Abstract Domain

\[ [R = e]^\#(\hat{a}) := \hat{a}[R \mapsto [e]^\#(\hat{a})] \]

\[ [R = M[e]]^\#(\hat{a}) := \hat{a}[R \mapsto \top] \]

\[ [M[e_1] = e_2]^\#(\hat{a}) := \hat{a} \]

\[ [\text{Pos}(e)]^\#(\hat{a}) := \hat{a} \]

\[ [\text{Neg}(e)]^\#(\hat{a}) := \hat{a} \]

Can this be done better?
Meaning of Statements in the Abstract Domain

\[
\boxed{[R = e]^\#(\widehat{a}) := \widehat{a}[R \mapsto [e]^\#(\widehat{a})]}
\]

\[
\boxed{[R = M[e]]^\#(\widehat{a}) := \widehat{a}[R \mapsto \top]}
\]

\[
\boxed{[M[e_1] = e_2]^\#(\widehat{a}) := \widehat{a}}
\]

\[
\boxed{[Pos(e)]^\#(\widehat{a}) := \widehat{a}}
\]

\[
\boxed{[Neg(e)]^\#(\widehat{a}) := \widehat{a}}
\]

Again:

For Correctness:  
\[
\gamma \vdash [statement]^\# \rightarrow \gamma
\]

For the best possible precision:

\[
\gamma \vdash [statement]^\# \rightarrow \alpha
\]
Meaning of Expressions

Evaluation of expressions is as expected:

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

\[
[e_1 \ op \ e_2]^\#(\hat{a}) := [e_1]^\#(\hat{a}) \ op^# \ [e_2]^\#(\hat{a})
\]
Evaluation of expressions is as expected:

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

\[ [e_1 \ op \ e_2]^\#(\widehat{a}) := [e_1]^\#(\widehat{a}) \ op^\# \ [e_2]^\#(\widehat{a}) \]

As we have seen earlier!
Putting it all together: The Abstract Reachability Semantics

Abstract Reachability Semantics captured as least fixed point of:

\[
\widehat{\text{Reach}} : V \rightarrow \text{States}
\]

\[
\widehat{\text{Reach}}(\text{start}) = \top \\
\forall v' \in V \setminus \{\text{start}\} : \widehat{\text{Reach}}(v') = \bigsqcup_{v \in V, (v,v') \in E} \text{[[labeling}(v, v')]]^\#(\widehat{\text{Reach}}(v))
\]

\[
\widehat{\text{Reach}}(1) = \text{[[labeling}(\text{start}, 1)]^\#(\widehat{\text{Reach}}(\text{start})) \sqcup \text{[[labeling}(2, 1)]^\#(\widehat{\text{Reach}}(2))
\widehat{\text{Reach}}(2) = \text{[[labeling}(1, 2)]^\#(\widehat{\text{Reach}}(1))
\widehat{\text{Reach}}(3) = \text{[[labeling}(1, 3)]^\#(\widehat{\text{Reach}}(1))
\]
\[
\widehat{\text{Reach}}(1) = \text{[[x = 0]}^\#(\widehat{\text{Reach}}(\text{start})) \sqcup \text{[[x = x + 1]}^\#(\widehat{\text{Reach}}(2))
\widehat{\text{Reach}}(2) = \text{[[Pos}(x < 100)]^\#(\widehat{\text{Reach}}(1))
\widehat{\text{Reach}}(3) = \text{[[Neg}(x < 100)]^\#(\widehat{\text{Reach}}(1))
\]
Example: Kleene Iteration to Compute Abstract Reachability Semantics
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[ x \rightarrow \top \quad x \rightarrow \bot \]

\[ x \rightarrow 0 \quad x \rightarrow \bot \]

1. \( x = 0 \)
2. \( \text{Pos}(x < 100) \)
3. \( \text{Neg}(x < 100) \)

\[ x \rightarrow \bot \]

\[ x = x+1 \]
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[ x \rightarrow \top \]
\[ x \rightarrow \bot \]
\[ x \rightarrow 0 \]
\[ x \rightarrow \bot \]
\[ x \rightarrow 0 \]
\[ x \rightarrow \bot \]
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[x \mapsto T\]

\[x \mapsto 0\]

\[x \mapsto \perp\]

Reachability semantics: 

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

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

\[x = x + 1\]

\[x = 0\]
Example: Kleene Iteration to Compute Abstract Reachability Semantics

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

```plaintext
y = 0;
x = 1;
z = 3;
while (x > 0) {
    if (x == 1) {
        y = 7;
    }
    else {
        y = z+4;
    }
    x = 3;
    print y;
}
```
The Abstract Transformer $F^#$

**Local Correctness Condition:**

![Diagram of Local Correctness Condition](image)
The Abstract Transformer $F^\#$

**Local Correctness Condition:**

![Diagram](image)

**Correct by construction**

*(if concretization and abstraction have certain properties):*

![Diagram](image)
From Local to Global Correctness: Kleene Iteration

Abstract Domain

Concrete Domain

\[ F \xrightarrow{\gamma} F^\# \xrightarrow{\gamma} F^\# \xrightarrow{\gamma} \ldots \]

\[ F^\# \xrightarrow{\gamma} \gamma \xrightarrow{\gamma} \gamma \xrightarrow{\gamma} \ldots \]

\[ \gamma \xrightarrow{\gamma} \gamma \xrightarrow{\gamma} \gamma \xrightarrow{\gamma} \ldots \]

"Idea for Correctness: Abstract Interpretation Cousot, Cousot 1977

Establish a description relation \( \Delta \) between the concrete values and their descriptions with:

\[ x \Delta a^1 \land a^1 \sqsubseteq a_2 \Rightarrow x \Delta a_2 \]

Concretization:

\[ \gamma a = \{ x \mid x \Delta a \} \]

// returns the set of described values
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 \(F^# \to F^#\) 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