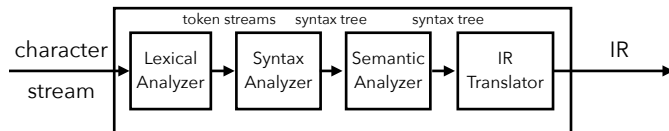


COSE312: Compilers

Lecture 13 — Semantic Analysis (1)

Hakjoo Oh
2025 Spring

Semantic Analysis

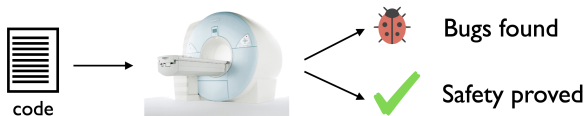


Semantic analysis aims to statically detect runtime errors, e.g.,

```
int a[10] = {...};
int x = rand();
int y = 1;
if (x > 0) {
    if (x < 15) {
        if (x < 10) a[x] = "hello" + y;
        a[x] = 1;
    }
} else {
    y = y / x;
}
```

Underlying Technology: Software Analysis

- Technology for catching bugs or proving correctness of software



- Widely used in software industry



A Hard Limit

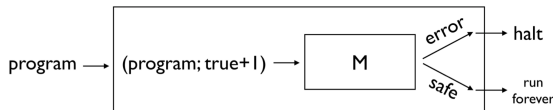
- The Halting problem is not computable



Alan Turing (1936)



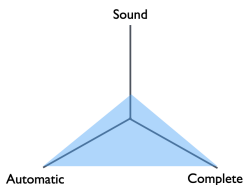
- If exact analysis is possible, we can solve the Halting problem



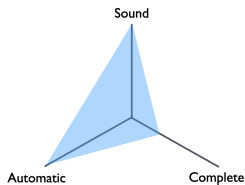
- Rice's theorem (1951): any non-trivial semantic property of a program is undecidable

Tradeoff

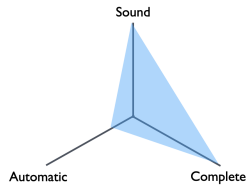
- Three desirable properties
 - ▶ **Soundness**: all program behaviors are captured
 - ▶ **Completeness**: only program behaviors are captured
 - ▶ **Automation**: without human intervention
- Achieving all of them is generally infeasible



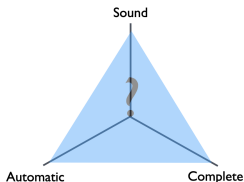
dynamic analysis



static analysis



verification



Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

- Dynamic analysis (testing): 459
- Static analysis: a variety of answers
 - ▶ “integer”, “odd integer”, “positive integer”, “ $400 \leq n \leq 500$ ”, etc
- Static analysis process:
 - 1 Choose abstract value (domain), e.g., $\hat{V} = \{\top, e, o, \perp\}$
 - 2 Define abstract semantics in terms of abstract values:

\hat{x}	\top	e	o	\perp	$\hat{+}$	\top	e	o	\perp
\top					\top				
e					e				
o					o				
\perp					\perp				

- 3 “Execute” the program:

$$e \hat{x} e \hat{+} o \hat{x} o = o$$

Principles of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs:

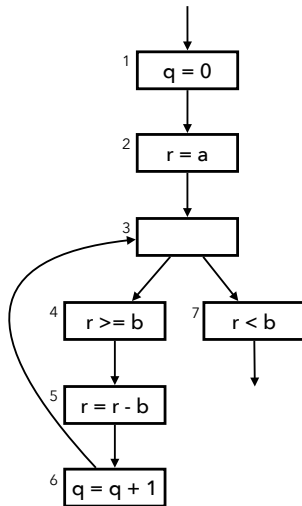
```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

- Instead, static analysis may produce false alarms:

```
void f (int x) {  
    y = x + x;  
    assert (y % 2 == 0);  
}
```

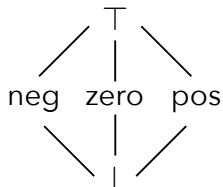
Example Program

```
// a >= 0, b >= 0
q = 0;
r = a;
while (r >= b) {
    r = r - b;
    q = q + 1;
}
assert(q >= 0);
assert(r >= 0);
```



A Simple Sign Analysis

- Abstract domain:

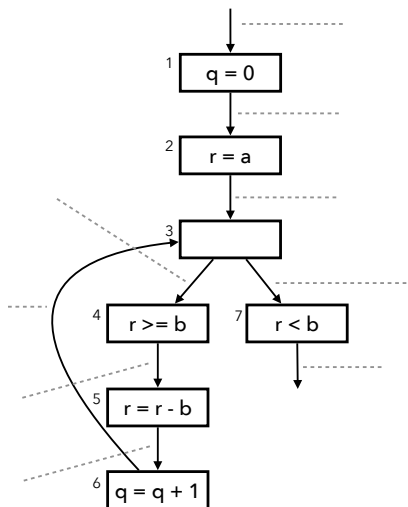


- Abstract semantics:

$+/-$	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

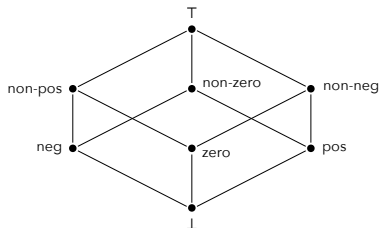
\times	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

Fixed Point Computation



An Extended Sign Analysis

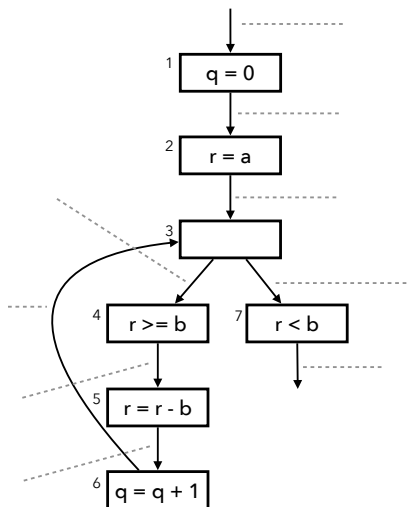
- Abstract domain:



- Abstract semantics:

+	top	neg	zero	pos	non-pos	non-zero	non-neg	bot
top								
neg								
zero								
pos								
non-pos								
non-zero								
non-neg								
bot								

Fixed Point Computation



An Abstract Semantics of **While**

The **While** language:

- Syntax:

$$a \rightarrow n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2$$

$$b \rightarrow \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$$

$$c \rightarrow x := a \mid \text{skip} \mid c_1; c_2 \mid \text{if } b \text{ } c_1 \text{ } c_2 \mid \text{while } b \text{ } c$$

- (Concrete) Semantics

$$\mathcal{A} \llbracket a \rrbracket : \text{State} \rightarrow \mathbb{Z}$$

$$\mathcal{B} \llbracket b \rrbracket : \text{State} \rightarrow \mathbf{T}$$

$$\mathcal{C} \llbracket c \rrbracket : \text{State} \hookrightarrow \text{State}$$

Abstract Values: Integers

- Concrete integers (\mathbb{Z}) are abstracted by the complete lattice $(\hat{\mathbb{Z}}, \sqsubseteq_{\hat{\mathbb{Z}}})$:

$$\hat{\mathbb{Z}} = \{\top_{\hat{\mathbb{Z}}}, \perp_{\hat{\mathbb{Z}}}, \mathbf{Pos}, \mathbf{Neg}, \mathbf{Zero}\}$$

$$\hat{a} \sqsubseteq_{\hat{\mathbb{Z}}} \hat{b} \iff \hat{a} = \hat{b} \vee \hat{a} = \perp_{\hat{\mathbb{Z}}} \vee \hat{b} = \top_{\hat{\mathbb{Z}}}$$

- An abstract integer denotes a set of integers.

► Abstraction function: $\alpha_{\hat{\mathbb{Z}}} : \mathcal{P}(\mathbb{Z}) \rightarrow \hat{\mathbb{Z}}$

► Concretization function: $\gamma_{\hat{\mathbb{Z}}} : \hat{\mathbb{Z}} \rightarrow \mathcal{P}(\mathbb{Z})$

$$\alpha_{\hat{\mathbb{Z}}}(\emptyset) = \perp_{\hat{\mathbb{Z}}}$$

$$\gamma_{\hat{\mathbb{Z}}}(\perp_{\hat{\mathbb{Z}}}) = \emptyset$$

$$\alpha_{\hat{\mathbb{Z}}}(S) = \mathbf{Pos} \quad (\forall n \in S. n > 0)$$

$$\gamma_{\hat{\mathbb{Z}}}(\top_{\hat{\mathbb{Z}}}) = \mathbb{Z}$$

$$\alpha_{\hat{\mathbb{Z}}}(S) = \mathbf{Neg} \quad (\forall n \in S. n < 0)$$

$$\gamma_{\hat{\mathbb{Z}}}(\mathbf{Pos}) = \{n \in \mathbb{Z} \mid n > 0\}$$

$$\alpha_{\hat{\mathbb{Z}}}(S) = \mathbf{Zero} \quad (S = \{0\})$$

$$\gamma_{\hat{\mathbb{Z}}}(\mathbf{Neg}) = \{n \in \mathbb{Z} \mid n < 0\}$$

$$\alpha_{\hat{\mathbb{Z}}}(S) = \top_{\hat{\mathbb{Z}}} \quad (\text{otherwise})$$

$$\gamma_{\hat{\mathbb{Z}}}(\mathbf{Zero}) = \{0\}$$

- Join (least upper bound) and meet (greatest lower bound):

$$\hat{a} \sqcup_{\hat{\mathbb{Z}}} \hat{b} = \hat{a} \quad (\hat{b} \sqsubseteq_{\hat{\mathbb{Z}}} \hat{a})$$

$$\hat{a} \sqcap_{\hat{\mathbb{Z}}} \hat{b} = \hat{b} \quad (\hat{b} \sqsubseteq_{\hat{\mathbb{Z}}} \hat{a})$$

$$\hat{a} \sqcup_{\hat{\mathbb{Z}}} \hat{b} = \hat{b} \quad (\hat{a} \sqsubseteq_{\hat{\mathbb{Z}}} \hat{b})$$

$$\hat{a} \sqcap_{\hat{\mathbb{Z}}} \hat{b} = \hat{a} \quad (\hat{a} \sqsubseteq_{\hat{\mathbb{Z}}} \hat{b})$$

$$\hat{a} \sqcup_{\hat{\mathbb{Z}}} \hat{b} = \top_{\hat{\mathbb{Z}}}$$

$$\hat{a} \sqcap_{\hat{\mathbb{Z}}} \hat{b} = \perp_{\hat{\mathbb{Z}}}$$

Abstract Values: Booleans

- The truth values $\mathbf{T} = \{true, false\}$ are abstracted by $(\hat{\mathbf{T}}, \sqsubseteq_{\hat{\mathbf{T}}})$:

$$\hat{\mathbf{T}} = \{\top_{\hat{\mathbf{T}}}, \perp_{\hat{\mathbf{T}}}, \widehat{true}, \widehat{false}\}$$

$$\hat{b}_1 \sqsubseteq_{\hat{\mathbf{T}}} \hat{b}_2 \iff \hat{b}_1 = \hat{b}_2 \vee \hat{b}_1 = \perp_{\hat{\mathbf{T}}} \vee \hat{b}_2 = \top_{\hat{\mathbf{T}}}$$

- An abstract boolean denotes a set of concrete booleans:

$\alpha_{\hat{\mathbf{T}}} : \mathcal{P}(\mathbf{T}) \rightarrow \hat{\mathbf{T}}$	$\gamma_{\hat{\mathbf{T}}} : \hat{\mathbf{T}} \rightarrow \mathcal{P}(\mathbf{T})$
$\alpha_{\hat{\mathbf{T}}}(\emptyset) = \perp_{\hat{\mathbf{T}}}$	$\gamma_{\hat{\mathbf{T}}}(\perp_{\hat{\mathbf{T}}}) = \emptyset$
$\alpha_{\hat{\mathbf{T}}}(\{true\}) = \widehat{true}$	$\gamma_{\hat{\mathbf{T}}}(\widehat{true}) = \{true\}$
$\alpha_{\hat{\mathbf{T}}}(\{false\}) = \widehat{false}$	$\gamma_{\hat{\mathbf{T}}}(\widehat{false}) = \{false\}$
$\alpha_{\hat{\mathbf{T}}}(\mathbf{T}) = \top_{\hat{\mathbf{T}}}$	$\gamma_{\hat{\mathbf{T}}}(\top_{\hat{\mathbf{T}}}) = \mathbf{T}$

- Join and meet:

$\hat{a} \sqcup_{\hat{\mathbf{T}}} \hat{b} = \hat{a} \ (\hat{b} \sqsubseteq_{\hat{\mathbf{T}}} \hat{a})$	$\hat{a} \sqcap_{\hat{\mathbf{T}}} \hat{b} = \hat{b} \ (\hat{b} \sqsubseteq_{\hat{\mathbf{T}}} \hat{a})$
$\hat{a} \sqcup_{\hat{\mathbf{T}}} \hat{b} = \hat{b} \ (\hat{a} \sqsubseteq_{\hat{\mathbf{T}}} \hat{b})$	$\hat{a} \sqcap_{\hat{\mathbf{T}}} \hat{b} = \hat{a} \ (\hat{a} \sqsubseteq_{\hat{\mathbf{T}}} \hat{b})$
$\hat{a} \sqcup_{\hat{\mathbf{T}}} \hat{b} = \top_{\hat{\mathbf{T}}}$	$\hat{a} \sqcup_{\hat{\mathbf{T}}} \hat{b} = \perp_{\hat{\mathbf{T}}}$

Abstract States

- Concrete states (State) are abstracted by $(\widehat{\text{State}}, \sqsubseteq_{\widehat{\text{State}}})$:

$$\widehat{\text{State}} = \text{Var} \rightarrow \widehat{\mathbb{Z}}$$

$$\hat{s}_1 \sqsubseteq_{\widehat{\text{State}}} \hat{s}_2 \iff \forall x \in \text{Var}. \hat{s}_1(x) \sqsubseteq_{\widehat{\mathbb{Z}}} \hat{s}_2(x).$$

- An abstract state denotes a set of concrete states:

$$\begin{aligned} \alpha_{\widehat{\text{State}}} &: \mathcal{P}(\text{State}) \rightarrow \widehat{\text{State}} \\ \alpha_{\widehat{\text{State}}}(S) &= \lambda x. \bigsqcup_{s \in S} \alpha_{\widehat{\mathbb{Z}}}(\{s(x)\}) \\ \gamma_{\widehat{\text{State}}} &= \widehat{\text{State}} \rightarrow \mathcal{P}(\text{State}) \\ \gamma_{\widehat{\text{State}}}(\hat{s}) &= \{s \in \text{State} \mid \forall x \in \text{Var}. s(x) \in \gamma_{\widehat{\mathbb{Z}}}(\hat{s}(x))\} \end{aligned}$$

- Join and meet:

$$\begin{aligned} \hat{s}_1 \sqcup_{\widehat{\text{State}}} \hat{s}_2 &= \lambda x. \hat{s}_1(x) \sqcup_{\widehat{\mathbb{Z}}} \hat{s}_2(x) \\ \hat{s}_1 \sqcap_{\widehat{\text{State}}} \hat{s}_2 &= \lambda x. \hat{s}_1(x) \sqcap_{\widehat{\mathbb{Z}}} \hat{s}_2(x) \end{aligned}$$

Abstract Semantics

$$\begin{aligned}\widehat{\mathcal{A}}[a] &: \widehat{\text{State}} \rightarrow \widehat{\mathbb{Z}} \\ \widehat{\mathcal{A}}[n](\hat{s}) &= \alpha_{\widehat{\mathbb{Z}}}(\{n\}) \\ \widehat{\mathcal{A}}[x](\hat{s}) &= \hat{s}(x) \\ \widehat{\mathcal{A}}[a_1 + a_2](\hat{s}) &= \widehat{\mathcal{A}}[a_1](\hat{s}) +_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[a_2](\hat{s}) \\ \widehat{\mathcal{A}}[a_1 \star a_2](\hat{s}) &= \widehat{\mathcal{A}}[a_1](\hat{s}) \star_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[a_2](\hat{s}) \\ \widehat{\mathcal{A}}[a_1 - a_2](\hat{s}) &= \widehat{\mathcal{A}}[a_1](\hat{s}) -_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[a_2](\hat{s}) \\ \widehat{\mathcal{B}}[b] &: \widehat{\text{State}} \rightarrow \widehat{\mathbb{T}} \\ \widehat{\mathcal{B}}[\text{true}](\hat{s}) &= \widehat{\text{true}} \\ \widehat{\mathcal{B}}[\text{false}](\hat{s}) &= \widehat{\text{false}} \\ \widehat{\mathcal{B}}[a_1 = a_2](\hat{s}) &= \widehat{\mathcal{A}}[a_1](\hat{s}) =_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[a_2](\hat{s}) \\ \widehat{\mathcal{B}}[a_1 \leq a_2](\hat{s}) &= \widehat{\mathcal{A}}[a_1](\hat{s}) \leq_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[a_2](\hat{s}) \\ \widehat{\mathcal{B}}[\neg b](\hat{s}) &= \neg_{\widehat{\mathbb{T}}} \widehat{\mathcal{B}}[b](\hat{s}) \\ \widehat{\mathcal{B}}[b_1 \wedge b_2](\hat{s}) &= \widehat{\mathcal{B}}[b_1](\hat{s}) \wedge_{\widehat{\mathbb{T}}} \widehat{\mathcal{B}}[b_2](\hat{s})\end{aligned}$$

Abstract Semantics

$$\begin{aligned}\widehat{\mathcal{C}}[c] &: \widehat{\text{State}} \rightarrow \widehat{\text{State}} \\ \widehat{\mathcal{C}}[x := a] &= \lambda \hat{s}. \hat{s}[x \mapsto \widehat{\mathcal{A}}[a](\hat{s})] \\ \widehat{\mathcal{C}}[\text{skip}] &= \text{id} \\ \widehat{\mathcal{C}}[c_1; c_2] &= \widehat{\mathcal{C}}[c_2] \circ \widehat{\mathcal{C}}[c_1] \\ \widehat{\mathcal{C}}[\text{if } b \text{ } c_1 \text{ } c_2] &= \widehat{\text{cond}}(\widehat{\mathcal{B}}[b], \widehat{\mathcal{C}}[c_1], \widehat{\mathcal{C}}[c_2]) \\ \widehat{\mathcal{C}}[\text{while } b \text{ } c] &= \text{fix } \widehat{F}\end{aligned}$$

where

$$\widehat{\text{cond}}(\hat{f}, \hat{g}, \hat{h}) = \lambda \hat{s}. \begin{cases} \perp_{\widehat{\text{State}}} & \dots \hat{f}(\hat{s}) = \perp_{\widehat{\text{T}}} \\ \hat{g}(\hat{s}) & \dots \hat{f}(\hat{s}) = \text{true} \\ \hat{h}(\hat{s}) & \dots \hat{f}(\hat{s}) = \text{false} \\ \hat{g}(\hat{s}) \sqcup_{\widehat{\text{State}}} \hat{h}(\hat{s}) & \dots \hat{f}(\hat{s}) = \top_{\widehat{\text{T}}} \end{cases}$$
$$\widehat{F}(\hat{g}) = \widehat{\text{cond}}(\widehat{\mathcal{B}}[b], \hat{g} \circ \widehat{\mathcal{C}}[c], \text{id})$$

Example: while $\neg(x = 0)$ skip

- $\widehat{F} : (\widehat{\text{State}} \rightarrow \widehat{\text{State}}) \rightarrow (\widehat{\text{State}} \rightarrow \widehat{\text{State}})$:

$$\widehat{F}(\widehat{g}) = \widehat{\text{cond}}(\widehat{\mathcal{B}}[\neg(x = 0)], \widehat{g} \circ \widehat{\mathcal{C}}[\text{skip}], \text{id}) = \widehat{\text{cond}}(\lambda \hat{s}. \hat{s}(x) \neq_{\widehat{\mathbb{Z}}} \text{Zero}, \widehat{g}, \text{id})$$

$$= \lambda \hat{s}. \begin{cases} \perp & \text{if } (\hat{s}(x) \neq_{\widehat{\mathbb{Z}}} \text{Zero}) = \perp \\ \widehat{g}(\hat{s}) & \text{if } (\hat{s}(x) \neq_{\widehat{\mathbb{Z}}} \text{Zero}) = \widehat{\text{true}} \\ \hat{s} & \text{if } (\hat{s}(x) \neq_{\widehat{\mathbb{Z}}} \text{Zero}) = \widehat{\text{false}} \\ \widehat{g}(\hat{s}) \sqcup \hat{s} & \text{if } (\hat{s}(x) \neq_{\widehat{\mathbb{Z}}} \text{Zero}) = \top \end{cases}$$

$$= \lambda \hat{s}. \begin{cases} \perp & \text{if } \hat{s}(x) = \perp \\ \widehat{g}(\hat{s}) & \text{if } \hat{s}(x) \in \{\text{Pos}, \text{Neg}\} \\ \hat{s} & \text{if } \hat{s}(x) = \text{Zero} \\ \widehat{g}(\hat{s}) \sqcup \hat{s} & \text{if } \hat{s}(x) = \top \end{cases}$$

- $\widehat{\mathcal{C}}[\text{while } \neg(x = 0) \text{ skip}] = \bigsqcup_{i \geq 0} \widehat{F}^i(\perp_{\widehat{\text{State}} \rightarrow \widehat{\text{State}}})$:

$$\textcircled{1} \widehat{g}_0 = \perp_{\widehat{\text{State}} \rightarrow \widehat{\text{State}}} = \lambda \hat{s}. \perp_{\widehat{\text{State}}}$$

$$\textcircled{2} \widehat{g}_1 = \widehat{F}(\widehat{g}_0) = \lambda \hat{s}. \begin{cases} \perp & \text{if } \hat{s}(x) = \perp \\ \perp & \text{if } \hat{s}(x) \in \{\text{Pos}, \text{Neg}\} \\ \hat{s} & \text{if } \hat{s}(x) = \text{Zero} \\ \hat{s} & \text{if } \hat{s}(x) = \top \end{cases}$$

$$\textcircled{3} \widehat{g}_2 = \widehat{F}(\widehat{g}_1) = \lambda \hat{s}. \begin{cases} \perp & \text{if } \hat{s}(x) = \perp \\ \widehat{g}_1(\hat{s}) = \perp & \text{if } \hat{s}(x) \in \{\text{Pos}, \text{Neg}\} \\ \hat{s} & \text{if } \hat{s}(x) = \text{Zero} \\ \widehat{g}_1(\hat{s}) \sqcup \hat{s} = \hat{s} \sqcup \hat{s} = \hat{s} & \text{if } \hat{s}(x) = \top \end{cases} = \widehat{g}_1$$

Static Analysis of Control-Flow Graphs

- Programs in **While** can be represented by control-flow graph $G = (N, \hookrightarrow)$, where each node $n \in N$ contains a command (denoted $cmd(n)$) defined as follows:

$$c \rightarrow x := a \mid assume(b) \mid skip$$

- Abstract semantics (transfer function) $\hat{f}_n : \widehat{\text{State}} \rightarrow \widehat{\text{State}}$:

$$\hat{f}_n(\hat{s}) = \begin{cases} \hat{s} & \text{if } cmd(n) = skip \\ \hat{s}[x \mapsto \hat{\mathcal{A}}[a](\hat{s})] & \text{if } x := a \\ \hat{s} & \text{if } assume(b), \widehat{true} \sqsubseteq \hat{\mathcal{B}}[b](\hat{s}) \\ \perp & \text{if } assume(b), \widehat{false} \sqsupseteq \hat{\mathcal{B}}[b](\hat{s}) \end{cases}$$

- The analysis is to compute the least fixed point of the function:

$$\hat{F} : (N \rightarrow \widehat{\text{State}}) \rightarrow (N \rightarrow \widehat{\text{State}})$$

$$\hat{F}(X) = \lambda n. \hat{f}_n(\bigsqcup_{n' \hookrightarrow n} X(n'))$$

Fixed Point Computation

- Tabulation algorithm:

$$\bigsqcup_{i \geq 0} \hat{F}^i(\lambda n. \perp) =$$

```
X := X' := λn.⊥
repeat
  X' := X
  X := X ⊔ F̂(X)
until X ⊆ X'
return X'
```

- Worklist algorithm:

```
W := N
X := λn.⊥
repeat
  n := choose(W)
  W := W \ {n}
  s := f̂_n(⊔_{n' ↦ n} X(n'))
  if s ⊈ X(n)
    X(n) := X(n) ⊔ s
    W := W ∪ {n | n ↦ n'}
until W = ∅
```

Summary

- Approaches to software analysis
- Principles of static analysis
- Simple and extended sign analysis
- Static analysis for **While**