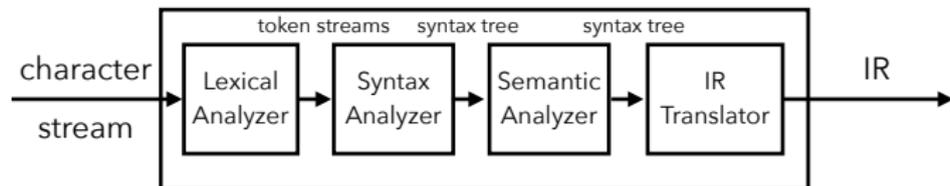# COSE312: Compilers

Lecture 13 — Semantic Analysis (1)

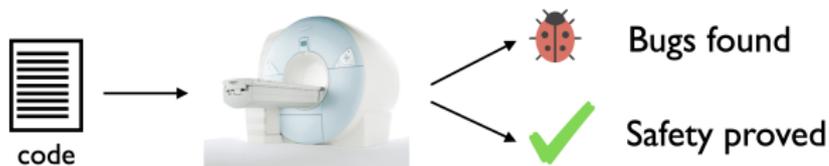Hakjoo Oh
2026 Spring

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

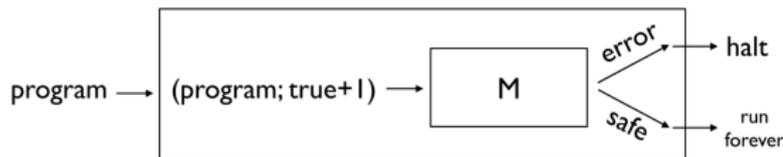- The Halting problem is not computable

Alan Turing (1936)

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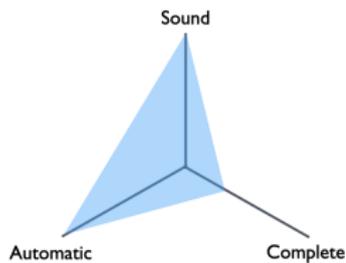

- Rice's theorem (1951): any non-trivial semantic property of a program is undecidable

Tradeoff
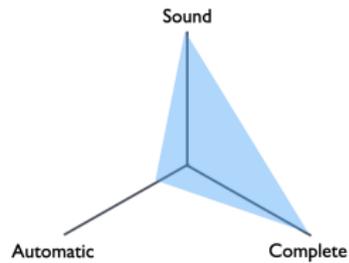
- Three desirable properties
  - Soundness: all program behaviors are captured
  - Completeness: only program behaviors are captured
  - Automation: without human intervention
- Achieving all of them is generally infeasible
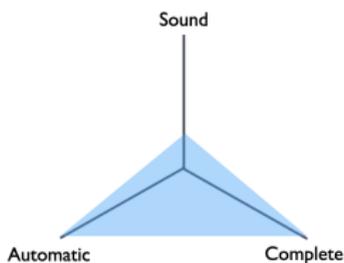
dynamic analysis   static analysis   verification
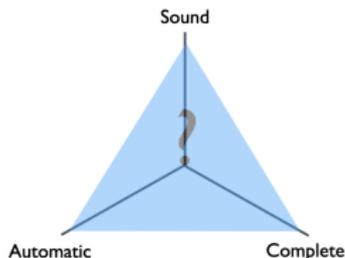
Principles of Static Analysis

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

- 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., $\hat{V} = \{\top, e, o, \bot\}$
  2. Define abstract semantics in terms of abstract values:

$\hat{\times}$	$\top$	e	o	$\bot$
$\top$				
e				
o				
$\bot$				

$\hat{+}$	$\top$	e	o	$\bot$
$\top$				
e				
o				
$\bot$				

  3. "Execute" the program:

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

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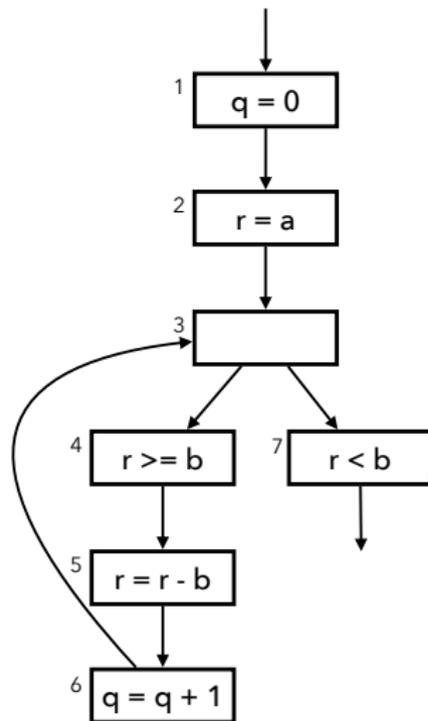
```
void f (int x)  {
    y = x + x;
    assert (y % 2 == 0);
}
```

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:

```
            ⊤
          / | \
         /  |  \
      neg  zero  pos
         \  |  /
          \ | /
            ⊥
```
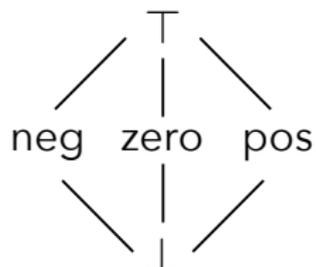
- Abstract semantics:

+/−	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

×	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

Fixed Point Computation
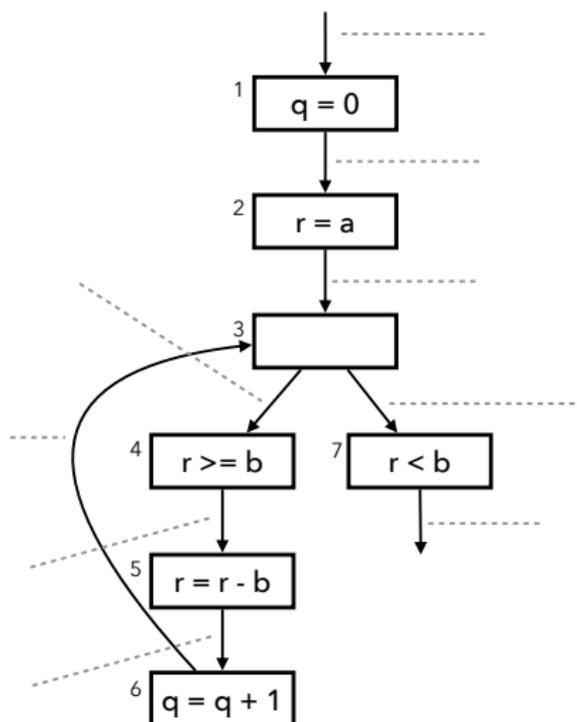
An Extended Sign Analysis

- Abstract domain:
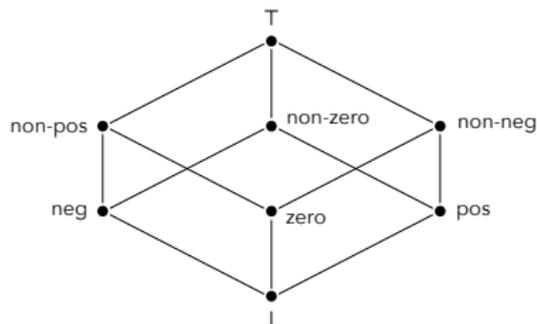
- Abstract semantics:

+	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
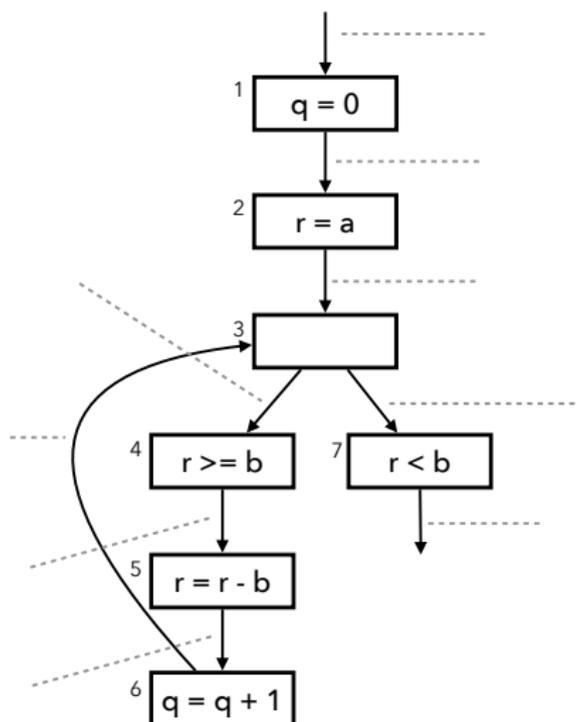
An Abstract Semantics of **While**

The **While** language:

- Syntax:

$$a \rightarrow n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2$$
$$b \rightarrow \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$$
$$c \rightarrow x := a \mid \text{skip} \mid c_1; c_2 \mid \text{if } b \ c_1 \ c_2 \mid \text{while } b \ c$$

- (Concrete) Semantics

$$\mathcal{A}[\![\, a \,]\!] : \text{State} \rightarrow \mathbb{Z}$$

$$\mathcal{B}[\![\, b \,]\!] : \text{State} \rightarrow \mathbf{T}$$

$$\mathcal{C}[\![\, c \,]\!] : \text{State} \hookrightarrow \text{State}$$

Abstract Values: Integers

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

$$\widehat{\mathbb{Z}} = \{\top_{\widehat{\mathbb{Z}}}, \bot_{\widehat{\mathbb{Z}}}, \textbf{Pos}, \textbf{Neg}, \textbf{Zero}\}$$

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

- An abstract integer denotes a set of integers.
    - Abstraction function: $\alpha_{\widehat{\mathbb{Z}}} : \mathcal{P}(\mathbb{Z}) \to \widehat{\mathbb{Z}}$
    - Concretization function: $\gamma_{\widehat{\mathbb{Z}}} : \widehat{\mathbb{Z}} \to \mathcal{P}(\mathbb{Z})$

$\alpha_{\widehat{\mathbb{Z}}}(\emptyset) = \bot_{\widehat{\mathbb{Z}}}$ $\gamma_{\widehat{\mathbb{Z}}}(\bot_{\widehat{\mathbb{Z}}}) = \emptyset$
$\alpha_{\widehat{\mathbb{Z}}}(S) = \textbf{Pos} \ \ (\forall n \in S.\ n > 0)$ $\gamma_{\widehat{\mathbb{Z}}}(\top_{\widehat{\mathbb{Z}}}) = \mathbb{Z}$
$\alpha_{\widehat{\mathbb{Z}}}(S) = \textbf{Neg} \ \ (\forall n \in S.\ n < 0)$ $\gamma_{\widehat{\mathbb{Z}}}(\textbf{Pos}) = \{n \in \mathbb{Z} \mid n > 0\}$
$\alpha_{\widehat{\mathbb{Z}}}(S) = \textbf{Zero} \ \ (S = \{0\})$ $\gamma_{\widehat{\mathbb{Z}}}(\textbf{Neg}) = \{n \in \mathbb{Z} \mid n < 0\}$
$\alpha_{\widehat{\mathbb{Z}}}(S) = \top_{\widehat{\mathbb{Z}}} \ \ (\text{otherwise})$ $\gamma_{\widehat{\mathbb{Z}}}(\textbf{Zero}) = \{0\}$

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

$$\hat{a} \sqcup_{\widehat{\mathbb{Z}}} \hat{b} = \hat{a} \ (\hat{b} \sqsubseteq_{\widehat{\mathbb{Z}}} \hat{a}) \qquad \hat{a} \sqcap_{\widehat{\mathbb{Z}}} \hat{b} = \hat{b} \ (\hat{b} \sqsubseteq_{\widehat{\mathbb{Z}}} \hat{a})$$
$$\hat{a} \sqcup_{\widehat{\mathbb{Z}}} \hat{b} = \hat{b} \ (\hat{a} \sqsubseteq_{\widehat{\mathbb{Z}}} \hat{b}) \qquad \hat{a} \sqcap_{\widehat{\mathbb{Z}}} \hat{b} = \hat{a} \ (\hat{a} \sqsubseteq_{\widehat{\mathbb{Z}}} \hat{b})$$
$$\hat{a} \sqcup_{\widehat{\mathbb{Z}}} \hat{b} = \top_{\widehat{\mathbb{Z}}} \qquad\qquad \hat{a} \sqcap_{\widehat{\mathbb{Z}}} \hat{b} = \bot_{\widehat{\mathbb{Z}}}$$

Abstract Values: Booleans

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

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

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

- An abstract boolean denotes a set of concrete booleans:

$$
\begin{array}{ll}
\alpha_{\widehat{\mathbf{T}}} : \mathcal{P}(\mathbf{T}) \to \widehat{\mathbf{T}} & \gamma_{\widehat{\mathbf{T}}} : \widehat{\mathbf{T}} \to \mathcal{P}(\mathbf{T}) \\
\alpha_{\widehat{\mathbf{T}}}(\emptyset) = \bot_{\widehat{\mathbf{T}}} & \gamma_{\widehat{\mathbf{T}}}(\bot_{\widehat{\mathbf{T}}}) = \emptyset \\
\alpha_{\widehat{\mathbf{T}}}(\{true\}) = \widehat{true} & \gamma_{\widehat{\mathbf{T}}}(\widehat{true}) = \{true\} \\
\alpha_{\widehat{\mathbf{T}}}(\{false\}) = \widehat{false} & \gamma_{\widehat{\mathbf{T}}}(\widehat{false}) = \{false\} \\
\alpha_{\widehat{\mathbf{T}}}(\mathbf{T}) = \top_{\widehat{\mathbf{T}}} & \gamma_{\widehat{\mathbf{T}}}(\top_{\widehat{\mathbf{T}}}) = \mathbf{T}
\end{array}
$$

- Join and meet:

$$
\begin{array}{ll}
\hat{a} \sqcup_{\widehat{\mathbf{T}}} \hat{b} = \hat{a} \ (\hat{b} \sqsubseteq_{\widehat{\mathbf{T}}} \hat{a}) & \hat{a} \sqcap_{\widehat{\mathbf{T}}} \hat{b} = \hat{b} \ (\hat{b} \sqsubseteq_{\widehat{\mathbf{T}}} \hat{a}) \\
\hat{a} \sqcup_{\widehat{\mathbf{T}}} \hat{b} = \hat{b} \ (\hat{a} \sqsubseteq_{\widehat{\mathbf{T}}} \hat{b}) & \hat{a} \sqcap_{\widehat{\mathbf{T}}} \hat{b} = \hat{a} \ (\hat{a} \sqsubseteq_{\widehat{\mathbf{T}}} \hat{b}) \\
\hat{a} \sqcup_{\widehat{\mathbf{T}}} \hat{b} = \top_{\widehat{\mathbf{T}}} & \hat{a} \sqcap_{\widehat{\mathbf{T}}} \hat{b} = \bot_{\widehat{\mathbf{T}}}
\end{array}
$$

Abstract States

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

$$\widehat{\mathbf{State}} = \mathbf{Var} \to \widehat{\mathbb{Z}}$$

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

- An abstract state denotes a set of concrete states:

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

- Join and meet:

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

Abstract Semantics

$$\widehat{\mathcal{A}}[\![\, a \,]\!] \quad : \quad \widehat{\mathbf{State}} \to \widehat{\mathbb{Z}}$$

$$\widehat{\mathcal{A}}[\![\, n \,]\!](\hat{s}) \;=\; \alpha_{\widehat{\mathbb{Z}}}(\{n\})$$

$$\widehat{\mathcal{A}}[\![\, x \,]\!](\hat{s}) \;=\; \hat{s}(x)$$

$$\widehat{\mathcal{A}}[\![\, a_1 + a_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{A}}[\![\, a_1 \,]\!](\hat{s}) +_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[\![\, a_2 \,]\!](\hat{s})$$

$$\widehat{\mathcal{A}}[\![\, a_1 \star a_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{A}}[\![\, a_1 \,]\!](\hat{s}) \star_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[\![\, a_2 \,]\!](\hat{s})$$

$$\widehat{\mathcal{A}}[\![\, a_1 - a_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{A}}[\![\, a_1 \,]\!](\hat{s}) -_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[\![\, a_2 \,]\!](\hat{s})$$

$$\widehat{\mathcal{B}}[\![\, b \,]\!] \quad : \quad \widehat{\mathbf{State}} \to \widehat{\mathbf{T}}$$

$$\widehat{\mathcal{B}}[\![\, \texttt{true} \,]\!](\hat{s}) \;=\; \widehat{true}$$

$$\widehat{\mathcal{B}}[\![\, \texttt{false} \,]\!](\hat{s}) \;=\; \widehat{false}$$

$$\widehat{\mathcal{B}}[\![\, a_1 = a_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{A}}[\![\, a_1 \,]\!](\hat{s}) =_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[\![\, a_2 \,]\!](\hat{s})$$

$$\widehat{\mathcal{B}}[\![\, a_1 \leq a_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{A}}[\![\, a_1 \,]\!](\hat{s}) \leq_{\widehat{\mathbb{Z}}} \widehat{\mathcal{A}}[\![\, a_2 \,]\!](\hat{s})$$

$$\widehat{\mathcal{B}}[\![\, \neg b \,]\!](\hat{s}) \;=\; \neg_{\widehat{\mathbf{T}}} \widehat{\mathcal{B}}[\![\, b \,]\!](\hat{s})$$

$$\widehat{\mathcal{B}}[\![\, b_1 \wedge b_2 \,]\!](\hat{s}) \;=\; \widehat{\mathcal{B}}[\![\, b_1 \,]\!](\hat{s}) \wedge_{\widehat{\mathbf{T}}} \widehat{\mathcal{B}}[\![\, b_2 \,]\!](\hat{s})$$

Abstract Semantics

$$\widehat{\mathcal{C}}[\![\, c \,]\!] \quad : \quad \widehat{\mathbf{State}} \to \widehat{\mathbf{State}}$$

$$\widehat{\mathcal{C}}[\![\, x := a \,]\!] \quad = \quad \lambda \hat{s}.\hat{s}[x \mapsto \widehat{\mathcal{A}}[\![\, a \,]\!](\hat{s})]$$

$$\widehat{\mathcal{C}}[\![\, \mathtt{skip} \,]\!] \quad = \quad \mathbf{id}$$

$$\widehat{\mathcal{C}}[\![\, c_1; c_2 \,]\!] \quad = \quad \widehat{\mathcal{C}}[\![\, c_2 \,]\!] \circ \widehat{\mathcal{C}}[\![\, c_1 \,]\!]$$

$$\widehat{\mathcal{C}}[\![\, \mathtt{if}\ b\ c_1\ c_2 \,]\!] \quad = \quad \widehat{\mathbf{cond}}(\widehat{\mathcal{B}}[\![\, b \,]\!], \widehat{\mathcal{C}}[\![\, c_1 \,]\!], \widehat{\mathcal{C}}[\![\, c_2 \,]\!])$$

$$\widehat{\mathcal{C}}[\![\, \mathtt{while}\ b\ c \,]\!] \quad = \quad \mathit{fix}\widehat{F}$$

where

$$\widehat{\mathbf{cond}}(\hat{f}, \hat{g}, \hat{h}) = \lambda \hat{s}. \begin{cases} \bot_{\widehat{\mathbf{State}}} & \cdots \hat{f}(\hat{s}) = \bot_{\widehat{\mathbf{T}}} \\ \hat{g}(\hat{s}) & \cdots \hat{f}(\hat{s}) = \widehat{true} \\ \hat{h}(\hat{s}) & \cdots \hat{f}(\hat{s}) = \widehat{false} \\ \hat{g}(\hat{s}) \sqcup_{\widehat{\mathbf{State}}} \hat{h}(\hat{s}) & \cdots \hat{f}(\hat{s}) = \top_{\widehat{\mathbf{T}}} \end{cases}$$

$$\widehat{F}(\hat{g}) = \widehat{\mathbf{cond}}(\widehat{\mathcal{B}}[\![\, b \,]\!], \hat{g} \circ \widehat{\mathcal{C}}[\![\, c \,]\!], \mathbf{id})$$

Example: `while ¬(x = 0) skip`

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

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

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

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

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

  1 $\widehat{g_0} = \bot_{\widehat{\text{State} \to \text{State}}} = \lambda \hat{s}.\bot_{\widehat{\text{State}}}$

  2 $\widehat{g_1} = \widehat{F}(\widehat{g_0}) = \lambda \hat{s}. \begin{cases} \bot & \text{if } \hat{s}(x) = \bot \\ \bot & \text{if } \hat{s}(x) \in \{\text{Pos, Neg}\} \\ \hat{s} & \text{if } \hat{s}(x) = \text{Zero} \\ \hat{s} & \text{if } \hat{s}(x) = \top \end{cases}$

  3 $\widehat{g_2} = \widehat{F}(\widehat{g_1}) = \lambda \hat{s}. \begin{cases} \bot & \text{if } \hat{s}(x) = \bot \\ \widehat{g_1}(\hat{s}) = \bot & \text{if } \hat{s}(x) \in \{\text{Pos, Neg}\} \\ \hat{s} & \text{if } \hat{s}(x) = \text{Zero} \\ \widehat{g_1}(\hat{s}) \sqcup \hat{s} = \hat{s} \sqcup \hat{s} = \hat{s} & \text{if } \hat{s}(x) = \top \end{cases} = \widehat{g_1}$

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:

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

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

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

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

$$\widehat{F} : (N \to \widehat{\textbf{State}}) \to (N \to \widehat{\textbf{State}})$$

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

Fixed Point Computation

- Tabulation algorithm:

$$\bigsqcup_{i \geq 0} \widehat{F}^i(\lambda n.\bot) = \begin{array}{l} X := X' := \lambda n.\bot \\ \texttt{repeat} \\ \quad X' := X \\ \quad X := X \sqcup \widehat{F}(X) \\ \texttt{until } X \sqsubseteq X' \\ \texttt{return } X' \end{array}$$

- Worklist algorithm:

$$\begin{array}{l} W := N \\ X := \lambda n.\bot \\ \texttt{repeat} \\ \quad n := \mathbf{choose}(W) \\ \quad W := W \setminus \{n\} \\ \quad s := \hat{f}_n(\bigsqcup_{n' \hookrightarrow n} X(n')) \\ \quad \texttt{if } s \not\sqsubseteq X(n) \\ \qquad X(n) := X(n) \sqcup s \\ \qquad W := W \cup \{n' \in N \mid n \hookrightarrow n'\} \\ \texttt{until } W = \emptyset \end{array}$$

Summary

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