Lecture 10 – Axiomatic Semantics

AAA551: Programming Language Theory

Jihyeok Park

PLRG

2026 Spring

Recall

- Continuations
  - A-Normal Form
- CC Machine
  - Recall: Evaluation Contexts and λ-Calculus
  - CC Machine
  - SCC Machine
- CK, CEK, CESK Machines
  - CK Machine
  - CEK Machine
  - CESK Machine
  - Variants of CEK Machine

Contents

Contents

Semantics

Three approaches to **formal semantics**:

- **Operational** $\sigma \vdash e \Rightarrow v$
  - *How* is a program executed?
  - Useful for implementation of compilers and interpreters.

$$\frac{\vdash e_1 \Rightarrow n_1 \qquad \vdash e_2 \Rightarrow n_2}{\vdash e_1 \texttt{ + } e_2 \Rightarrow n_1 + n_2}$$

- **Denotational** $[\![e]\!]$
  - *What* is the mathematical object for a program?
  - Useful for theoretical foundations.

$$[\![e_1 \texttt{ + } e_2]\!] = [\![e_1]\!] + [\![e_2]\!]$$

- **Axiomatic** $\vdash \{\phi\}\ e\ \{\phi'\}$
  - *Which properties* does a program satisfy?
  - Useful for proving program properties and correctness.

$$\vdash \{x < n \land y < m\}\ z \texttt{ = } x \texttt{ + } y\ \{z < n + m\}$$

Hoare Triples

In axiomatic semantics, a **Hoare triple** represents properties of a statement s:

$$\{\phi\}\ s\ \{\phi'\}$$

which means that if the statement s is executed in a state satisfying the **precondition** ϕ and **terminates**, then the **postcondition** ϕ' will hold in the resulting state.

Hoare Triples

In axiomatic semantics, a **Hoare triple** represents properties of a statement s:

$$\{\phi\}\ s\ \{\phi'\}$$

which means that if the statement s is executed in a state satisfying the **precondition** ϕ and **terminates**, then the **postcondition** ϕ' will hold in the resulting state.

For example,

$$\{x < n \land y < m\}\ z \texttt{ = } x \texttt{ + } y\ \{z < n + m\}$$

Hoare Triples

In axiomatic semantics, a **Hoare triple** represents properties of a statement s:

$$\{\phi\}\ s\ \{\phi'\}$$

which means that if the statement s is executed in a state satisfying the **precondition** ϕ and **terminates**, then the **postcondition** ϕ' will hold in the resulting state.

For example,

$$\{x < n \land y < m\}\ z \texttt{ = } x \texttt{ + } y\ \{z < n + m\}$$

$$\{0 \leq n \land i = n \land x = 1\}$$

```
while (0 < i) {
    x = x * i;
    i = i - 1;
}
```

$$\{x = n!\}$$

Partial Correctness vs. Total Correctness

There are two types of correctness properties:

- **Partial correctness**: **If** a statement s is executed in a state satisfying ϕ and **terminates**, then ϕ' will hold in the resulting state:

$$\{\phi\}\ s\ \{\phi'\}$$

- **Total correctness**: If a statement s is executed in a state satisfying ϕ, **then** it **terminates** and ϕ' will hold in the resulting state:

$$[\phi]\ s\ [\phi']$$

The main difference is that total correctness requires termination, while partial correctness does not.

Partial Correctness vs. Total Correctness

PLRG

Which ones are correct?

1. $\{$false$\}$ `while (true) {}` $\{$false$\}$
2. $\{$true$\}$ `while (true) {}` $\{$false$\}$
3. [false] `while (true) {}` [false]
4. [true] `while (true) {}` [false]

Partial Correctness vs. Total Correctness

Which ones are correct?

1. {false} `while (true) {}` {false}
2. {true} `while (true) {}` {false}
3. [false] `while (true) {}` [false]
4. [true] `while (true) {}` [false]

The 1 and 3 are **correct** because no state satisfies the precondition.

Partial Correctness vs. Total Correctness

PLRG

Which ones are correct?

1. `{false} while (true) {} {false}`

2. `{true} while (true) {} {false}`

3. `[false] while (true) {} [false]`

4. `[true] while (true) {} [false]`

The 1 and 3 are **correct** because no state satisfies the precondition.

The 2 is **correct** because it does not terminate, so the postcondition is never violated even though it is `false`.

Partial Correctness vs. Total Correctness

Which ones are correct?

1. `{false} while (true) {} {false}`
2. `{true} while (true) {} {false}`
3. `[false] while (true) {} [false]`
4. `[true] while (true) {} [false]`

The 1 and 3 are **correct** because no state satisfies the precondition.

The 2 is **correct** because it does not terminate, so the postcondition is never violated even though it is `false`.

The 4 is **incorrect** because it does not terminate, which violates the total correctness requirement.

Assertion Language

Recall the language IMP we defined in previous lectures.

$$
\begin{array}{llcl}
\text{Expressions} & e & ::= & n \mid x \mid e \texttt{ + } e \mid e \texttt{ * } e \mid e \texttt{ < } e \mid \texttt{true} \mid \texttt{false} \\
\text{Statements} & s & ::= & \texttt{skip} \mid x \texttt{ := } e \mid s\texttt{;}s \\
 & & & \mid \texttt{if } e \texttt{ then } s \texttt{ else } s \\
 & & & \mid \texttt{while } e \texttt{ do } s \\
\text{Values} & v & ::= & n \mid \texttt{true} \mid \texttt{false}
\end{array}
$$

Assertion Language

PLRG

Recall the language IMP we defined in previous lectures.

$$
\begin{array}{lrl}
\text{Expressions} & e ::= & n \mid x \mid e \texttt{ + } e \mid e \texttt{ * } e \mid e \texttt{ < } e \mid \texttt{true} \mid \texttt{false} \\
\text{Statements} & s ::= & \texttt{skip} \mid x \texttt{ := } e \mid s \texttt{;} s \\
 & & \mid \texttt{if } e \texttt{ then } s \texttt{ else } s \\
 & & \mid \texttt{while } e \texttt{ do } s \\
\text{Values} & v ::= & n \mid \texttt{true} \mid \texttt{false}
\end{array}
$$

Let's define an **assertion language** for IMP.

$$
\begin{array}{lll}
i, j & \in & \textbf{LVar} \\
a & ::= & x \mid i \mid n \mid a \texttt{ + } a \mid a \texttt{ * } a \\
\phi & ::= & \texttt{true} \mid \texttt{false} \mid a \texttt{ = } a \mid a \texttt{ < } a \mid \neg\phi \mid \phi \land \phi \mid \phi \lor \phi \mid \exists x.\phi \mid \forall x.\phi
\end{array}
$$

where **LVar** is a set of **logical variables** different from the program variables in $\mathbb{X}$ (e.g., x, y, z), a is an **arithmetic expression** used in assertions, and ϕ is a **assertion** that can be true or false.

Denotational Semantics of Arithmetic Expressions

A **program state** $\sigma : \mathbb{X} \to \mathbb{V}$ is a mapping from program variables to values (ignoring the labels $\mathbb{L}$ for now).

A **program state** $\sigma : \mathbb{X} \to \mathbb{V}$ is a mapping from program variables to values (ignoring the labels $\mathbb{L}$ for now).

In addition, we need an **interpretation** I for logical variables, which is a mapping from logical variables to integers:

$$I : \textbf{LVar} \to \mathbb{Z}$$

Denotational Semantics of Arithmetic Expressions

A **program state** $\sigma : \mathbb{X} \rightarrow \mathbb{V}$ is a mapping from program variables to values (ignoring the labels $\mathbb{L}$ for now).

In addition, we need an **interpretation** I for logical variables, which is a mapping from logical variables to integers:

$$I : \textbf{LVar} \rightarrow \mathbb{Z}$$

We can define the **denotational semantics** of **arithmetic expressions** with a given pair of state σ and interpretation I:

$$\boxed{\mathcal{A}[\![a]\!](\sigma, I) : \mathbb{Z}}$$

$$\begin{array}{lcll}
\mathcal{A}[\![x]\!](\sigma, I) & = & \sigma(x) & \text{only if } \sigma(x) \in \mathbb{Z} \\
\mathcal{A}[\![i]\!](\sigma, I) & = & I(i) & \\
\mathcal{A}[\![n]\!](\sigma, I) & = & n & \\
\mathcal{A}[\![a_1 \texttt{ + } a_2]\!](\sigma, I) & = & \mathcal{A}[\![a_1]\!](\sigma, I) + \mathcal{A}[\![a_2]\!](\sigma, I) & \\
\mathcal{A}[\![a_1 \texttt{ * } a_2]\!](\sigma, I) & = & \mathcal{A}[\![a_1]\!](\sigma, I) \times \mathcal{A}[\![a_2]\!](\sigma, I) &
\end{array}$$

Satisfaction

Now, we can define the **satisfaction relation** $\models_I$ for assertions:

$$\boxed{\sigma \models_I \phi}$$

$$
\begin{array}{ll}
\sigma \models_I \texttt{true} & \\
\sigma \models_I a_1 = a_2 & \text{if } \mathcal{A}[\![a_1]\!](\sigma, I) = \mathcal{A}[\![a_2]\!](\sigma, I) \\
\sigma \models_I a_1 < a_2 & \text{if } \mathcal{A}[\![a_1]\!](\sigma, I) < \mathcal{A}[\![a_2]\!](\sigma, I) \\
\sigma \models_I \neg\phi & \text{if } \neg\sigma \models_I \phi \\
\sigma \models_I \phi_1 \wedge \phi_2 & \text{if } \sigma \models_I \phi_1 \wedge \sigma \models_I \phi_2 \\
\sigma \models_I \phi_1 \vee \phi_2 & \text{if } \sigma \models_I \phi_1 \vee \sigma \models_I \phi_2 \\
\sigma \models_I \exists i.\phi & \text{if } \exists k \in \mathbb{Z}.\ \sigma \models_{I[i \mapsto k]} \phi \\
\sigma \models_I \forall i.\phi & \text{if } \forall k \in \mathbb{Z}, \sigma \models_{I[i \mapsto k]} \phi
\end{array}
$$

Satisfaction

PLRG

Definition (Partial Correctness Statement Satisfiability)

A partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is **satisfied** in a state σ and an interpretation I:

$$\sigma \models_I \{\phi\}\ s\ \{\phi'\}$$

if and only if the following condition holds:

$$\sigma \models_I \phi \land S[\![s]\!](\sigma) = \sigma' \implies \sigma' \models_I \phi'$$

Satisfaction

PLRG

Definition (Partial Correctness Statement Satisfiability)

A partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is **satisfied** in a state σ and an interpretation I:

$$\sigma \models_I \{\phi\}\ s\ \{\phi'\}$$

if and only if the following condition holds:

$$\sigma \models_I \phi \land S[\![s]\!](\sigma) = \sigma' \implies \sigma' \models_I \phi'$$

Note that $S[\![s]\!](\sigma) = \sigma'$ means that the execution of statement s from state σ terminates in state σ'.

The definition of partial correctness statement satisfiability does not require termination.

Validity

Definition (Assertion Validity)

An assertion ϕ is **valid** if and only if it holds for all states and interpretations:

$$\models \phi \iff \forall \sigma, I.\ \sigma \models_I \phi$$

Validity

Definition (Assertion Validity)

An assertion ϕ is **valid** if and only if it holds for all states and interpretations:

$$\models \phi \iff \forall \sigma, I.\ \sigma \models_I \phi$$

Definition (Partial Correctness Statement Validity)

A partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is **valid** if and only if it is satisfied in all states and interpretations:

$$\models \{\phi\}\ s\ \{\phi'\} \iff \forall \sigma, I.\ \sigma \models_I \{\phi\}\ s\ \{\phi'\}$$

Validity

PLRG

Definition (Assertion Validity)

An assertion ϕ is **valid** if and only if it holds for all states and interpretations:

$$\models \phi \iff \forall \sigma, I.\ \sigma \models_I \phi$$

Definition (Partial Correctness Statement Validity)

A partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is **valid** if and only if it is satisfied in all states and interpretations:

$$\models \{\phi\}\ s\ \{\phi'\} \iff \forall \sigma, I.\ \sigma \models_I \{\phi\}\ s\ \{\phi'\}$$

Then, how to prove the validity of a partial correctness statement?

Validity

Definition (Assertion Validity)

An assertion ϕ is **valid** if and only if it holds for all states and interpretations:

$$\models \phi \iff \forall \sigma, I.\ \sigma \models_I \phi$$

Definition (Partial Correctness Statement Validity)

A partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is **valid** if and only if it is satisfied in all states and interpretations:

$$\models \{\phi\}\ s\ \{\phi'\} \iff \forall \sigma, I.\ \sigma \models_I \{\phi\}\ s\ \{\phi'\}$$

Then, how to prove the validity of a partial correctness statement?

We can use **Hoare logic**, which provides a set of inference rules to derive valid partial correctness statements.

Contents

Hoare Logic

Let's define the inference rules for Hoare logic, which allow us to derive valid partial correctness statements.

We will define the following **judgment**:

$$\vdash \{\phi\}\ s\ \{\phi'\}$$

which means that the partial correctness statement $\{\phi\}\ s\ \{\phi'\}$ is derivable using the inference rules of Hoare logic.

Hoare Logic – Skip

$$\textsc{Skip}\ \frac{}{\vdash \{\phi\}\ \texttt{skip}\ \{\phi\}}$$

The precondition and postcondition are the same for the `skip` statement.

Hoare Logic – Skip

$$\textsc{Skip}\ \frac{}{\vdash \{\phi\}\ \texttt{skip}\ \{\phi\}}$$

The precondition and postcondition are the same for the `skip` statement.

For example,

$$\textsc{Skip}\ \frac{}{\vdash \{x < n\}\ \texttt{skip}\ \{x < n\}}$$

Hoare Logic – Assignment

$$\textsc{Assign}\ \frac{}{\vdash \{\phi[x \mapsto a]\}\ x\ \texttt{:=}\ a\ \{\phi\}}$$

$$\textsc{Assign}\ \frac{}{\vdash \{\phi[x \mapsto a]\}\ x\ \texttt{:=}\ a\ \{\phi\}}$$

To satisfy the postcondition ϕ after the assignment x := a, the precondition must be ϕ with x replaced by a (i.e., $\phi[x \mapsto a]$).

Hoare Logic – Assignment

$$\textsc{Assign}\ \frac{}{\vdash \{\phi[x \mapsto a]\}\ x\ \texttt{:=}\ a\ \{\phi\}}$$

To satisfy the postcondition ϕ after the assignment x `:=` a, the precondition must be ϕ with x replaced by a (i.e., $\phi[x \mapsto a]$).

For example,

$$\textsc{Assign}\ \frac{}{\vdash \{\texttt{true}\}\ x\ \texttt{:=}\ 42\ \{x = 42\}}$$

Hoare Logic – Assignment

$$\textsc{Assign}\ \frac{}{\vdash \{\phi[x \mapsto a]\}\ x\ \texttt{:=}\ a\ \{\phi\}}$$

To satisfy the postcondition ϕ after the assignment x := a, the precondition must be ϕ with x replaced by a (i.e., $\phi[x \mapsto a]$).

For example,

$$\textsc{Assign}\ \frac{}{\vdash \{\texttt{true}\}\ x\ \texttt{:=}\ 42\ \{x = 42\}}$$

$$\textsc{Assign}\ \frac{}{\vdash \{x < n\}\ x\ \texttt{:=}\ x + 1\ \{x < n + 1\}}$$

Hoare Logic – Assignment (Incorrect)

The following inference rule for assignment is **incorrect**:

$$\textsc{Assign}\ \frac{}{\vdash \{\phi\}\ x\ \texttt{:=}\ a\ \{\phi[x \mapsto a]\}}$$

For example,

$$\textsc{Assign}\ \frac{}{\vdash \{x = 0\}\ x\ \texttt{:=}\ 5\ \{5 = 0\}}$$

Hoare Logic – Assignment (Incorrect)

The following inference rule for assignment is **incorrect**:

$$\textsc{Assign}\ \frac{}{\vdash \{\phi\}\ x\ \texttt{:=}\ a\ \{\phi[x \mapsto a]\}}$$

For example,

$$\textsc{Assign}\ \frac{}{\vdash \{x = 0\}\ x\ \texttt{:=}\ 5\ \{5 = 0\}}$$

Another **incorrect** inference rule for assignment is:

$$\textsc{Assign}\ \frac{}{\vdash \{\phi\}\ x\ \texttt{:=}\ a\ \{\phi[a \mapsto x]\}}$$

For example,

$$\textsc{Assign}\ \frac{}{\vdash \{x = 0\}\ x\ \texttt{:=}\ 5\ \{x = 0\}}$$

Hoare Logic – Sequence

$$\textsc{Seq}\ \frac{\vdash \{\phi\}\ s_1\ \{\phi''\} \qquad \vdash \{\phi''\}\ s_2\ \{\phi'\}}{\vdash \{\phi\}\ s_1\,;s_2\ \{\phi'\}}$$

To satisfy the postcondition ϕ' after executing $s_1\,;s_2$, we need to find an intermediate assertion ϕ'' such that s_1 satisfies ϕ'' and s_2 satisfies ϕ'.

Hoare Logic – Sequence

$$\textsc{Seq}\ \frac{\vdash \{\phi\}\ s_1\ \{\phi''\} \qquad \vdash \{\phi''\}\ s_2\ \{\phi'\}}{\vdash \{\phi\}\ s_1 \,;\, s_2\ \{\phi'\}}$$

To satisfy the postcondition ϕ' after executing $s_1 \,;\, s_2$, we need to find an intermediate assertion ϕ'' such that s_1 satisfies ϕ'' and s_2 satisfies ϕ'.

For example,

$$\textsc{Seq}\ \frac{\vdash \{\texttt{true}\}\ x\ \texttt{:=}\ 42\ \{x = 42\} \qquad \vdash \{x = 42\}\ y\ \texttt{:=}\ x + 1\ \{y = 43\}}{\vdash \{\texttt{true}\}\ x\ \texttt{:=}\ 42;\ y\ \texttt{:=}\ x + 1\ \{y = 43\}}$$

Hoare Logic – Conditionals

$$\textsc{If}\ \frac{\vdash \{\phi \land e\}\ s_1\ \{\phi'\} \qquad \vdash \{\phi \land \neg e\}\ s_2\ \{\phi'\}}{\vdash \{\phi\}\ \texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \{\phi'\}}$$

To satisfy the postcondition ϕ' after executing the conditional statement, we need to ensure that

- s_1 satisfies ϕ' when the condition e is true, and
- s_2 satisfies ϕ' when the condition e is false.

Hoare Logic – Conditionals

$$\text{IF}\ \frac{\vdash \{\phi \land e\}\ s_1\ \{\phi'\} \qquad \vdash \{\phi \land \neg e\}\ s_2\ \{\phi'\}}{\vdash \{\phi\}\ \texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \{\phi'\}}$$

To satisfy the postcondition ϕ' after executing the conditional statement, we need to ensure that

- s_1 satisfies ϕ' when the condition e is true, and
- s_2 satisfies ϕ' when the condition e is false.

For example,

$$\text{IF}\ \frac{\vdash \{x > 0\}\ y\ \texttt{:=}\ x\ \{y \geq 0\} \qquad \vdash \{x \leq 0\}\ y\ \texttt{:=}\ -1\ \texttt{*}\ x\ \{y \geq 0\}}{\vdash \{\texttt{true}\}\ \texttt{if}\ x > 0\ \texttt{then}\ y\ \texttt{:=}\ x\ \texttt{else}\ y\ \texttt{:=}\ -1\ \texttt{*}\ x\ \{y \geq 0\}}$$

Hoare Logic – Loops

$$\textsc{While}\ \frac{\vdash \{\phi \land e\}\ s\ \{\phi\}}{\vdash \{\phi\}\ \texttt{while}\ e\ \texttt{do}\ s\ \{\phi \land \neg e\}}$$

The precondition ϕ is called the **loop invariant**, which must hold before and after each iteration of the loop. The postcondition $\phi \land \neg e$ holds when the loop terminates.

Hoare Logic – Loops

$$\textsc{While}\ \frac{\vdash \{\phi \land e\}\ s\ \{\phi\}}{\vdash \{\phi\}\ \texttt{while}\ e\ \texttt{do}\ s\ \{\phi \land \neg e\}}$$

The precondition ϕ is called the **loop invariant**, which must hold before and after each iteration of the loop. The postcondition $\phi \land \neg e$ holds when the loop terminates.

For example,

$$\textsc{While}\ \frac{\vdash \{i < n\}\ i\ \texttt{:=}\ i\ \texttt{+}\ 1\ \{i \leq n\}}{\vdash \{i \leq n\}\ \texttt{while}\ i\ \texttt{<}\ n\ \texttt{do}\ i\ \texttt{:=}\ i\ \texttt{+}\ 1\ \{i = n\}}$$

Hoare Logic – Consequences

$$\textsc{Consequence}\ \frac{\vdash \phi \Rightarrow \phi_1 \qquad \vdash \{\phi_1\}\ s\ \{\phi_1'\} \qquad \vdash \phi_1' \Rightarrow \phi'}{\vdash \{\phi\}\ s\ \{\phi'\}}$$

We can always **strengthen precondition** or **weaken postcondition** of a valid partial correctness statement.

Hoare Logic – Consequences

$$\textsc{Consequence}\ \frac{\vdash \phi \Rightarrow \phi_1 \qquad \vdash \{\phi_1\}\ s\ \{\phi'_1\} \qquad \vdash \phi'_1 \Rightarrow \phi'}{\vdash \{\phi\}\ s\ \{\phi'\}}$$

We can always **strengthen precondition** or **weaken postcondition** of a valid partial correctness statement.

For example,

$$\textsc{Consequence}\ \frac{\begin{array}{c}\vdash x < n \Rightarrow x < n + 1 \\ \vdash \{x < n + 1\}\ x\ \texttt{:=}\ x + 1\ \{x < n + 2\} \\ \vdash x < n + 2 \Rightarrow x < n + 3\end{array}}{\vdash \{x < n\}\ x\ \texttt{:=}\ x + 1\ \{x < n + 3\}}$$

Example – Factorial

$\{0 \leq n \wedge i = n \wedge x = 1\}$

$\{0 \leq n \wedge i = n \wedge x \times i! = n!\}$

$\{0 \leq n \wedge 0 \leq i \leq n \wedge x \times i! = n!\}$

```
while (0 < i) {
```

$\{0 \leq n \wedge 0 < i \leq n \wedge x \times i! = n!\}$

```
    x = x * i;
```

$\{0 \leq n \wedge 0 < i \leq n \wedge x \times (i-1)! = n!\}$

```
    i = i - 1;
```

$\{0 \leq n \wedge 0 \leq i < n \wedge x \times i! = n!\}$

$\{0 \leq n \wedge 0 \leq i \leq n \wedge x \times i! = n!\}$

```
}
```

$\{0 \leq n \wedge i = 0 \wedge x \times i! = n!\}$

$\{x = n!\}$

Soundness and Completeness

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

Soundness and Completeness

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

We can prove the soundness of Hoare logic by induction.

Soundness and Completeness

PLRG

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

We can prove the soundness of Hoare logic by induction.

Theorem (Completeness)

All valid partial correctness statements are derivable in Hoare logic:

$$\models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\}$$

Soundness and Completeness

PLRG

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

We can prove the soundness of Hoare logic by induction.

Theorem (Completeness)

All valid partial correctness statements are derivable in Hoare logic:

$$\models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\}$$

The following statement is valid only if s does not terminate.

$$\{\texttt{true}\}\ s\ \{\texttt{false}\}$$

Soundness and Completeness

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

We can prove the soundness of Hoare logic by induction.

Theorem (Completeness)

All valid partial correctness statements are derivable in Hoare logic:

$$\models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\}$$

The following statement is valid only if s does not terminate.

$$\{\texttt{true}\}\ s\ \{\texttt{false}\}$$

If Hoare logic is complete, $\vdash \{\texttt{true}\}\ s\ \{\texttt{false}\}$ must be derivable.

Soundness and Completeness

Theorem (Soundness)

All derivable partial correctness statements are valid:

$$\vdash \{\phi\}\ s\ \{\phi'\} \implies \models \{\phi\}\ s\ \{\phi'\}$$

We can prove the soundness of Hoare logic by induction.

Theorem (Completeness)

All valid partial correctness statements are derivable in Hoare logic:

$$\models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\}$$

The following statement is valid only if s does not terminate.

$$\{\texttt{true}\}\ s\ \{\texttt{false}\}$$

If Hoare logic is complete, $\vdash \{\texttt{true}\}\ s\ \{\texttt{false}\}$ must be derivable.

However, it is impossible to solve the halting problem, which means that Hoare logic is **not complete** for all statements.

Relative Completeness

PLRG

Theorem (Relative Completeness)

If we assume that all valid assertions are provable in the assertion logic, then all valid partial correctness statements are derivable in Hoare logic:

$$(\forall \phi.\ \models \phi \implies \vdash \phi) \implies (\forall \phi, s, \phi'.\ \models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\})$$

Relative Completeness

Theorem (Relative Completeness)

If we assume that all valid assertions are provable in the assertion logic, then all valid partial correctness statements are derivable in Hoare logic:

$$(\forall \phi.\ \models \phi \implies \vdash \phi) \implies (\forall \phi, s, \phi'.\ \models \{\phi\}\ s\ \{\phi'\} \implies \vdash \{\phi\}\ s\ \{\phi'\})$$

So, if we have an **oracle** that can prove all valid assertions, we can use it to derive all valid partial correctness statements in Hoare logic.

Relative Completeness

Theorem (Relative Completeness)

If we assume that all valid assertions are provable in the assertion logic, then all valid partial correctness statements are derivable in Hoare logic:

$$(\forall \phi.\ \models \phi \implies \vdash \phi) \implies (\forall \phi, s, \phi'.\ \models \{\phi\}\ s\ \{\phi'\} \implies\ \vdash \{\phi\}\ s\ \{\phi'\})$$

So, if we have an **oracle** that can prove all valid assertions, we can use it to derive all valid partial correctness statements in Hoare logic.

If we do not have a derivation (proof) of a valid partial correctness statement, the reason must be that we cannot prove some valid assertion used in the proof, not because of the incompleteness of Hoare logic itself.

Next Lecture

- Systematic Program Proofs

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr