Axiomatic Semantics

Hongfei Fu

John Hopcroft Center for Computer Science
Shanghai Jiao Tong University

Apr. 8th

Previous Lecture

- finishing denotational semantics (Chapter 5)

Today's Topic

Axiomatic Semantics

- logical specifications for programs
- partial correctness assertions
- proof rules for partial correctness assertions

Axiomatic Semantics: An Intuition

Textbook, Page 77 – 78

Axiomatic Semantics: An Intuition

A Simple Example

Consider the command (program) c as follows:

$$
\begin{array}{l}
S := 0\,; \\
N := 1\,; \\
\textbf{while}\ \ \neg(N = 101)\ \ \textbf{do}\ \ (S := S + N\,;\ \ N := N + 1)
\end{array}
$$

Axiomatic Semantics: An Intuition

A Simple Example

Consider the command (program) c as follows:

$$
\begin{array}{l}
S := 0\,; \\
N := 1\,; \\
\textbf{while}\ \ \neg(N = 101)\ \ \textbf{do}\ \ (S := S + N\,;\ \ N := N + 1)
\end{array}
$$

Our Goal

For any $\sigma, \sigma' \in \Sigma$, $\langle c, \sigma \rangle \to \sigma'$ implies $\sigma'(S) = \sum_{k=1}^{100} k = 5050$.

Axiomatic Semantics: An Intuition

The First Part

$$\{\textbf{true}\}\ \ S := 0;\ \ N := 1\ \ \{S = 0 \wedge N = 1\}$$

Axiomatic Semantics: An Intuition

The Loop Body

$$\{S = \sum_{k=1}^{N-1} k \wedge \neg(N = 101)\}$$

$$S := S + N;\quad N := N + 1$$

$$\{S = \sum_{k=1}^{N-1} k\}$$

The Whole While-Loop

$$\{S = \sum_{k=1}^{N-1} k\}$$

$$\textbf{while}\ \ \neg(N = 101)\ \ \textbf{do}\ \ (S := S + N;\quad N := N + 1)$$

$$\{S = \sum_{k=1}^{N-1} k \wedge N = 101\}$$

Axiomatic Semantics: An Intuition

Putting Together

$$\{\textbf{true}\}\ \ S := 0;\ \ N := 1\ \ \{S = 0 \wedge N = 1\}$$

$$\{S = \textstyle\sum_{k=1}^{N-1} k\}$$

$$\textbf{while}\ \ \neg(N = 101)\ \ \textbf{do}\ \ (S := S + N;\ \ N := N + 1)$$

$$\{S = \textstyle\sum_{k=1}^{N-1} k \wedge N = 101\}$$

- $(S = \sum_{k=1}^{N-1} k \wedge N = 101) \Rightarrow S = \sum_{k=1}^{100} k = 5050$

Axiomatic Semantics: An Intuition

The Logical Layout

- c: a command
- A, B: logical formulas

Then the assertion $\{A\}c\{B\}$ means that

for all states σ that satisfy A, if $\langle c, \sigma \rangle \to \sigma'$ then σ' satisfies B.

Axiomatic Semantics: An Overview

Textbook, Page 78 – 80

Axiomatic Semantics

Partial Correctness Assertions

- A, B: logical formulas
- $\sigma \models A$: σ satisfies A
- c: a command

A partial correctness assertion is of the form $\{A\}c\{B\}$, meaning

$$\forall \sigma, \sigma' \in \Sigma. \left((\sigma \models A \wedge \langle c, \sigma \rangle \to \sigma') \Rightarrow \sigma' \models B \right) \quad .$$

Axiomatic Semantics

Partial Correctness Assertions

- A, B: logical formulas
- $\sigma \models A$: σ satisfies A
- c: a command

A partial correctness assertion is of the form $\{A\}c\{B\}$, meaning

$$\forall \sigma, \sigma' \in \Sigma. \left((\sigma \models A \wedge \langle c, \sigma \rangle \to \sigma') \Rightarrow \sigma' \models B \right) \quad .$$

Terminology

- A: precondition
- B: postcondition

Axiomatic Semantics

Partial Correctness Assertions

- A, B: logical formulas
- $\sigma \models A$: σ satisfies A
- c: a command

A partial correctness assertion is of the form $\{A\}c\{B\}$, meaning

$$\forall \sigma, \sigma' \in \Sigma. \left((\sigma \models A \wedge \langle c, \sigma \rangle \to \sigma') \Rightarrow \sigma' \models B \right) \quad .$$

The Core of the Axiomatic Semantics

- logical properties for input-output relationships
- no guarantee of termination

Axiomatic Semantics

An Example

- $c :=$ **while true do skip**
- $\{\textbf{true}\}c\{\textbf{false}\}$

Axiomatic Semantics

An Example

- c := **while true do skip**
- $\{\textbf{true}\}c\{\textbf{false}\}$

Question

Does $\{\textbf{true}\}c\{\textbf{false}\}$ hold?

Axiomatic Semantics

Total Correctness Assertions

- A, B: logical formulas
- c: a command

Axiomatic Semantics

Total Correctness Assertions

- A, B: logical formulas
- c: a command

Then $[A]c[B]$ means that

- $\forall \sigma, \sigma' \in \Sigma. ((\sigma \models A \wedge \langle c, \sigma \rangle \to \sigma' \Rightarrow \sigma') \models B)$,
- $\forall \sigma \in \Sigma. [\sigma \models A \Rightarrow \exists \sigma' \in \Sigma. (\langle c, \sigma \rangle \to \sigma')]$.

Axiomatic Semantics

Total Correctness Assertions

- A, B: logical formulas
- c: a command

Then $[A]c[B]$ means that

- $\forall \sigma, \sigma' \in \Sigma. ((\sigma \models A \wedge \langle c, \sigma \rangle \to \sigma' \Rightarrow \sigma') \models B)$,
- $\forall \sigma \in \Sigma. [\sigma \models A \Rightarrow \exists \sigma' \in \Sigma. (\langle c, \sigma \rangle \to \sigma')]$.

Observation

total correctness = termination + partial correctness

The Bottom Element

- $\bot$: the fresh element for non-termination.

The Bottom Element

- $\bot$: the fresh element for non-termination.
- $\mathcal{C}[\![c]\!](\sigma) := \bot$ if c does not terminate on the initial state σ.

Axiomatic Semantics

The Bottom Element

- $\bot$: the fresh element for non-termination.
- $\mathcal{C}[\![c]\!](\sigma) := \bot$ if c does not terminate on the initial state σ.
- $\mathcal{C}[\![c]\!](\bot) := \bot$.

Axiomatic Semantics

The Bottom Element

- $\bot$: the fresh element for non-termination.
- $\mathcal{C}[\![c]\!](\sigma) := \bot$ if c does not terminate on the initial state σ.
- $\mathcal{C}[\![c]\!](\bot) := \bot$.
- $\bot \models A$ for all logical formulas A.

Axiomatic Semantics

Definition with the Bottom Element

A partial correctness assertion $\{A\}c\{B\}$ means equivalently that

$$\forall \sigma \in \Sigma. (\sigma \models A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models B) \quad .$$

Axiomatic Semantics

The Central Question

How to build the axiomatic semantics (i.e. $\{A\}c\{B\}$) ?

Axiomatic Semantics

The Road Map

- a formal language for logical formulas
- a collection of rules for partial correctness assertions

An Assertion Language **Assn**

Textbook, Page 80 – 86

An Assertion Language **Assn**

Informal Description

- first-order logical formulas
- satisfaction defined over states

An Assertion Language **Assn**

Example: Primality

- $\text{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables
- $\sigma \models \mathrm{Prime}$ iff $\sigma(X)$ is a prime number.

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables
- $\sigma \models \mathrm{Prime}$ iff $\sigma(X)$ is a prime number.

Observation

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables
- $\sigma \models \mathrm{Prime}$ iff $\sigma(X)$ is a prime number.

Observation

- locations, arithmetic expressions, propositional logical operators

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables
- $\sigma \models \mathrm{Prime}$ iff $\sigma(X)$ is a prime number.

Observation

- locations, arithmetic expressions, propositional logical operators
- integer variables

An Assertion Language **Assn**

Example: Primality

- $\mathrm{Prime} := X \geq 0 \wedge \neg(\exists i.\exists j.(i \geq 2 \wedge j \geq 2 \wedge X = i \times j))$
- X: a location (program variable)
- i, j: integer variables
- $\sigma \models \mathrm{Prime}$ iff $\sigma(X)$ is a prime number.

Observation

- locations, arithmetic expressions, propositional logical operators
- integer variables
- universal/existential quantification

An Assertion Language **Assn**

Extended Arithmetic Expressions **Aexpv**

$$a ::= n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

An Assertion Language **Assn**

Extended Arithmetic Expressions **Aexpv**

$$a ::= n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

- n: an integer
- X: a location
- i: an integer variable (from **Intvar**)

An Assertion Language **Assn**

Examples

- $X + Y - 3$
- $(i \times j) + k$
- $X + (i \times Y) + 5 - (4 \times j)$

An Assertion Language **Assn**

Examples

- $X + Y - 3$
- $(i \times j) + k$
- $X + (i \times Y) + 5 - (4 \times j)$

Integer Variables

Why do we include integer variables?

An Assertion Language **Assn**

Examples

- $X + Y - 3$
- $(i \times j) + k$
- $X + (i \times Y) + 5 - (4 \times j)$

Integer Variables

Why do we include integer variables?

- more expressibility for organizing logical properties
- more ability for representing unknown initial values

An Assertion Language **Assn**

Extended Boolean Assertions **Assn**

$$
\begin{array}{rcl}
A & ::= & \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \\
& & A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \\
& & \forall i.A \mid \exists i.A
\end{array}
$$

An Assertion Language **Assn**

Extended Boolean Assertions **Assn**

$$
\begin{array}{rcl}
A & ::= & \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \\
& & A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \\
& & \forall i.A \mid \exists i.A
\end{array}
$$

- a_0, a_1: extended arithmetic expressions from **Aexpv**
- i: an integer variable from **Intvar**

An Assertion Language **Assn**

Extended Boolean Assertions **Assn**

$$
\begin{array}{rcl}
A & ::= & \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \\
& & A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \\
& & \forall i.A \mid \exists i.A
\end{array}
$$

- a_0, a_1: extended arithmetic expressions from **Aexpv**
- i: an integer variable from **Intvar**
- $\wedge, \vee, \neg$: logical connectives from propositional logic
- $\forall, \exists$: quantifiers from first-order logic

An Assertion Language **Assn**

Extended Boolean Assertions **Assn**

$$
\begin{array}{rcl}
A & ::= & \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \\
& & A_0 \wedge A_1 \mid A_0 \vee A_1 \mid \neg A \mid A_0 \Rightarrow A_1 \mid \\
& & \forall i.A \mid \exists i.A
\end{array}
$$

Satisfaction Relation $\models$: An Intuition

A state σ satisfies an assertion $A \in \textbf{Assn}$ (written as $\sigma \models A$) if A is true when all locations X in A is replaced by $\sigma(X)$.

An Assertion Language **Assn**

Satisfaction Relation $\models$: Main Issues

- quantifiers
- integer variables

An Assertion Language **Assn**

Satisfaction Relation $\models$: Main Issues

- quantifiers
- integer variables

Important Points

- free/bound variables
- substitution
- interpretations

An Assertion Language **Assn**

Free and Bound Variables

- free variables: integer variables not associated with quantifiers
- bound variables: integer variables associated with quantifiers

An Assertion Language **Assn**

Free and Bound Variables

- free variables: integer variables not associated with quantifiers
- bound variables: integer variables associated with quantifiers
- notation: $FV(-)$

An Assertion Language **Assn**

Free and Bound Variables

- free variables: integer variables not associated with quantifiers
- bound variables: integer variables associated with quantifiers
- notation: $FV(-)$

Free and Bound Variables: Examples

- $\exists i.(k = i \times l)$;
- $(i + 100 \leq 77) \wedge \forall i.(j + 1 = i + 3)$

Free and Bound Variables

Definition through Well-Founded Recursion

All integer variables in extended arithmetic expressions are free:

- $FV(n) = FV(X) := \emptyset$;
- $FV(i) := \{i\}$;

Free and Bound Variables

Definition through Well-Founded Recursion

All integer variables in extended arithmetic expressions are free:

- $FV(n) = FV(X) := \emptyset$;
- $FV(i) := \{i\}$;
- $FV(a_0 + a_1) := FV(a_0) \cup FV(a_1)$;
- $FV(a_0 - a_1) := FV(a_0) \cup FV(a_1)$;
- $FV(a_0 \times a_1) := FV(a_0) \cup FV(a_1)$.

Free and Bound Variables

Definition through Well-Founded Recursion

Quantified integer variables are removed from free variables:

- $FV(\textbf{true}) = FV(\textbf{false}) := \emptyset$;

Free and Bound Variables

Definition through Well-Founded Recursion

Quantified integer variables are removed from free variables:

- $FV(\textbf{true}) = FV(\textbf{false}) := \emptyset$;
- $FV(a_0 = a_1) = FV(a_0 \leq a_1) := FV(a_0) \cup FV(a_1)$;

Free and Bound Variables

Definition through Well-Founded Recursion

Quantified integer variables are removed from free variables:

- $FV(\textbf{true}) = FV(\textbf{false}) := \emptyset$;
- $FV(a_0 = a_1) = FV(a_0 \leq a_1) := FV(a_0) \cup FV(a_1)$;
- $FV(A_0 \bowtie A_1) = FV(A_0) \cup FV(A_1)$ for $\bowtie \in \{\wedge, \vee, \Rightarrow\}$;
- $FV(\neg A) = FV(A)$;

Free and Bound Variables

Definition through Well-Founded Recursion

Quantified integer variables are removed from free variables:

- $FV(\textbf{true}) = FV(\textbf{false}) := \emptyset$;
- $FV(a_0 = a_1) = FV(a_0 \leq a_1) := FV(a_0) \cup FV(a_1)$;
- $FV(A_0 \bowtie A_1) = FV(A_0) \cup FV(A_1)$ for $\bowtie \in \{\wedge, \vee, \Rightarrow\}$;
- $FV(\neg A) = FV(A)$;
- $FV(\forall i.A) = FV(\exists i.A) = FV(A) \setminus \{i\}$.

Free and Bound Variables

Definitions

- A: an assertion from **Assn**
- i: an integer variable that appears in A

Free and Bound Variables

Definitions

- A: an assertion from **Assn**
- i: an integer variable that appears in A

Then:

- i is free in A if $i \in FV(A)$.

Free and Bound Variables

Definitions

- A: an assertion from **Assn**
- i: an integer variable that appears in A

Then:

- i is free in A if $i \in FV(A)$.
- i is bound in A if $i \notin FV(A)$.

Free and Bound Variables

Definitions

- A: an assertion from **Assn**
- i: an integer variable that appears in A

Then:

- i is free in A if $i \in FV(A)$.
- i is bound in A if $i \notin FV(A)$.
- A is closed if $FV(A) = \emptyset$.

Free and Bound Variables

Examples

- $FV(i = 1) = \{i\}$;
- $FV(\forall i.\, (i \times i \geq 0)) = \emptyset$;
- $FV(i = 1 \vee \forall i.\, (i \times i \geq 0)) = \{i\}$;

Substitution

Informal Description

- $a \in$ **Aexp**: an arithmetic expression without integer variables

Substitution

Informal Description

- $a \in$ **Aexp**: an arithmetic expression without integer variables
- i: an integer variable

Substitution

Informal Description

- $a \in$ **Aexp**: an arithmetic expression without integer variables
- i: an integer variable
- A: an assertion such that $i \in FV(A)$

Substitution

Informal Description

- $a \in$ **Aexp**: an arithmetic expression without integer variables
- i: an integer variable
- A: an assertion such that $i \in FV(A)$

Then $A[a/i]$ is the assertion obtained by substituting all free occurrences of i in A by a.

Substitution

Definition: Extended Arithmetic Expressions

- $n\,[a/i] := n$;
- $X\,[a/i] := X$;
- $j\,[a/i] := j$ if $j \neq i$;
- $j\,[a/i] := a$ if $j = i$;
- $(a_0 \bowtie a_1)\,[a/i] := a_0\,[a/i] \bowtie a_1\,[a/i]$ for $\bowtie \in \{+, -, \times\}$;

Substitution

Definition: Extended Boolean Assertions **Assn**

- $\mathbf{true}\,[a/i] := \mathbf{true}$;
- $\mathbf{false}\,[a/i] := \mathbf{false}$;

Substitution

Definition: Extended Boolean Assertions **Assn**

- $\mathbf{true}\,[a/i] := \mathbf{true}$;
- $\mathbf{false}\,[a/i] := \mathbf{false}$;
- $(a_0 = a_1)\,[a/i] := a_0\,[a/i] = a_1\,[a/i]$;
- $(a_0 \leq a_1)\,[a/i] := a_0\,[a/i] \leq a_1\,[a/i]$;

Substitution

Definition: Extended Boolean Assertions **Assn**

- $\textbf{true}\,[a/i] := \textbf{true}$;
- $\textbf{false}\,[a/i] := \textbf{false}$;
- $(a_0 = a_1)\,[a/i] := a_0\,[a/i] = a_1\,[a/i]$;
- $(a_0 \leq a_1)\,[a/i] := a_0\,[a/i] \leq a_1\,[a/i]$;
- $(A_0 \bowtie A_1)\,[a/i] := A_0\,[a/i] \bowtie A_1\,[a/i]$ for $\bowtie \in \{\wedge, \vee, \Rightarrow\}$;
- $(\neg A)\,[a/i] := \neg(A\,[a/i])$;

Substitution

Definition: Extended Boolean Assertions **Assn**

Universal Quantification:

- $(\forall j.A)\,[a/i] := \forall j.(A\,[a/i])$ if $j \neq i$;
- $(\forall j.A)\,[a/i] := \forall j.A$ if $j = i$;

Substitution

Definition: Extended Boolean Assertions **Assn**

Universal Quantification:

- $(\forall j.A)\,[a/i] := \forall j.(A\,[a/i])$ if $j \neq i$;
- $(\forall j.A)\,[a/i] := \forall j.A$ if $j = i$;

Existential Quantification:

- $(\exists j.A)\,[a/i] := \exists j.(A\,[a/i])$ if $j \neq i$;
- $(\exists j.A)\,[a/i] := \exists j.A$ if $j = i$;

Substitution

Examples

Substitution

Examples

- $(\exists j.i = j + 1)\,[X/i] = \exists j.X = j + 1$;

Substitution

Examples

- $(\exists j.i = j + 1)\,[X/i] = \exists j.X = j + 1$;
- $(\exists j.i = j + 1)\,[X/j] = \exists j.i = j + 1$;

Substitution

Examples

- $(\exists j.i = j + 1)\,[X/i] = \exists j.X = j + 1$;
- $(\exists j.i = j + 1)\,[X/j] = \exists j.i = j + 1$;
- $(\exists j.i = j + 1)\,[X + j/i] = ?$;

Interpretation

Definition

- An interpretation is a function $I : \mathbf{Intvar} \to \mathbb{Z}$ which assigns an integer to each integer variable.

Interpretation

Definition

- An interpretation is a function $I : \mathbf{Intvar} \to \mathbb{Z}$ which assigns an integer to each integer variable.
- An interpretation instantiates every free integer variable.

Interpretation

Definition

- An interpretation is a function $I : \mathbf{Intvar} \to \mathbb{Z}$ which assigns an integer to each integer variable.
- An interpretation instantiates every free integer variable.

Substitution

$$(I\,[n/i])(j) := \begin{cases} n & \text{if } j = i \\ I(j) & \text{otherwise} \end{cases}$$

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Then we have:

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Then we have:

- $\mathcal{A}v[\![n]\!](I,\sigma) := n$;
- $\mathcal{A}v[\![X]\!](I,\sigma) := \sigma(X)$;

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Then we have:

- $\mathcal{A}v[\![n]\!](I,\sigma) := n$;
- $\mathcal{A}v[\![X]\!](I,\sigma) := \sigma(X)$;
- $\mathcal{A}v[\![i]\!](I,\sigma) := I(i)$;

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Then we have:

- $\mathcal{A}v[\![n]\!](I,\sigma) := n$;
- $\mathcal{A}v[\![X]\!](I,\sigma) := \sigma(X)$;
- $\mathcal{A}v[\![i]\!](I,\sigma) := I(i)$;
- $\mathcal{A}v[\![a_0 \bowtie a_1]\!](I,\sigma) := \mathcal{A}v[\![a_0]\!](I,\sigma) \bowtie \mathcal{A}v[\![a_1]\!](I,\sigma)$ for $\bowtie \in \{+,-,\times\}$;

Semantics of Assertions **Assn**

Definition over Extended Arithmetic Expressions

- I: an interpretation
- σ: a state

Then we have:

- $\mathcal{A}v[\![n]\!](I,\sigma) := n$;
- $\mathcal{A}v[\![X]\!](I,\sigma) := \sigma(X)$;
- $\mathcal{A}v[\![i]\!](I,\sigma) := I(i)$;
- $\mathcal{A}v[\![a_0 \bowtie a_1]\!](I,\sigma) := \mathcal{A}v[\![a_0]\!](I,\sigma) \bowtie \mathcal{A}v[\![a_1]\!](I,\sigma)$ for $\bowtie \in \{+,-,\times\}$;

Exercise

For all (unextended) arithmetic expressions $a \in \mathbf{Aexp}$, it holds that

$$\forall \sigma, I.\, (\mathcal{A}[\![a]\!](\sigma) = \mathcal{A}v[\![a]\!](I,\sigma)) \quad .$$

Semantics of Assertions **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Semantics of Assertions **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- it always holds that $\sigma \models^I$ **true**;
- it always does not hold that $\sigma \models^I$ **false**;

Semantics of Assertions **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- $\sigma \models^I (a_0 = a_1)$ iff $\mathcal{A}v[\![a_0]\!](I, \sigma) = \mathcal{A}v[\![a_1]\!](I, \sigma)$;
- $\sigma \models^I (a_0 \leq a_1)$ iff $\mathcal{A}v[\![a_0]\!](I, \sigma) \leq \mathcal{A}v[\![a_1]\!](I, \sigma)$;

Semantics of Assertions from **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- $\sigma \models^I (A \wedge B)$ iff $\sigma \models^I A$ and $\sigma \models^I B$;
- $\sigma \models^I (A \vee B)$ iff $\sigma \models^I A$ or $\sigma \models^I B$;
- $\sigma \models^I \neg A$ iff (not $\sigma \models^I A$);
- $\sigma \models^I (A \Rightarrow B)$ iff (not $\sigma \models^I A$) or $\sigma \models^I B$;

Semantics of Assertions from **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- $\sigma \models^I \forall i.A$ iff for all integers n, $\sigma \models^{I[n/i]} A$;
- $\sigma \models^I \exists i.A$ iff there exists an integer n such that $\sigma \models^{I[n/i]} A$;

Semantics of Assertions from **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- $\bot \models^I A$ for all assertions $A \in$ **Assn**.

Semantics of Assertions from **Assn**

The Satisfaction Relation $\models$

- σ: a state
- I: an interpretation
- A: an assertion from **Assn**

Defining $\sigma \models^I A$ ("σ satisfies A in I"):

- $\bot \models^I A$ for all assertions $A \in$ **Assn**.

Notation

"not $\sigma \models^I A$" by "$\sigma \not\models^I A$"

Semantics of Assertions from **Assn**

Exercise

For all (unextended) boolean expressions $b \in \mathbf{Bexp}$, states $\sigma \in \Sigma$ and interpretations I, it holds that

- $\mathcal{B}[\![b]\!](\sigma) = \mathbf{true}$ iff $\sigma \models^I b$, and
- $\mathcal{B}[\![b]\!](\sigma) = \mathbf{false}$ iff $\sigma \not\models^I b$.

Semantics of Assertions from **Assn**

Exercise

For any extended arithmetic expression $a \in \mathbf{Aexpv}$, interpretation I and state σ, it holds that

$$\mathcal{A}v[\![a]\!](I\,[n/i], \sigma) = \mathcal{A}v[\![a\,[n/i]]\!](I, \sigma)$$

for all integers n and integer variables i.

Semantics of Assertions from **Assn**

Exercise

For any extended arithmetic expression $a \in \mathbf{Aexpv}$, interpretation I and state σ, it holds that

$$\mathcal{A}v[\![a]\!](I\,[n/i], \sigma) = \mathcal{A}v[\![a\,[n/i]]\!](I, \sigma)$$

for all integers n and integer variables i.

Exercise

- $\sigma \models^I \forall i.A$ iff $\sigma \models^I A\,[n/i]$ for all integers n.
- $\sigma \models^I \exists i.A$ iff $\sigma \models^I A\,[n/i]$ for some integer n.

Semantics of Assertions from **Assn**

Exercise

For any extended arithmetic expression $a \in \mathbf{Aexpv}$, interpretation I and state σ, it holds that

$$\mathcal{A}v[\![a]\!](I\,[n/i], \sigma) = \mathcal{A}v[\![a\,[n/i]]\!](I, \sigma)$$

for all integers n and integer variables i.

Exercise

- $\sigma \models^I \forall i.A$ iff $\sigma \models^I A\,[n/i]$ for all integers n.
- $\sigma \models^I \exists i.A$ iff $\sigma \models^I A\,[n/i]$ for some integer n.
- solution: prove by induction on the structure of A that $\sigma \models^{I[n/i]} A$ iff $\sigma \models^I A\,[n/i]$

Semantics of Assertions from **Assn**

Extension of Assertions

- A: an assertion in **Assn**
- I: an interpretation

Semantics of Assertions from **Assn**

Extension of Assertions

- A: an assertion in **Assn**
- I: an interpretation
- $A^I := \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}$.

Semantics of Assertions from **Assn**

Extension of Assertions

- A: an assertion in **Assn**
- I: an interpretation
- $A^I := \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}$.

Validity for **Assn**

- A is valid: $\models A$ iff for all interpretations I and all states σ, $\sigma \models^I A$.

Partial Correctness Assertions

Textbook, Page 87 – 89

Partial Correctness Assertions

Definition

A partial correctness assertion is of the form

$$\{A\}c\{B\}$$

where $A, B \in \mathbf{Assn}$ and $c \in \mathbf{Com}$.

Partial Correctness Assertions

Definition

A partial correctness assertion is of the form

$$\{A\}c\{B\}$$

where $A, B \in \mathbf{Assn}$ and $c \in \mathbf{Com}$.

Satisfaction Relation $\models$

- I: an interpretation
- σ: an element in $\Sigma_\perp$

We define that $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models^I B)$.

Partial Correctness Assertions

Satisfaction Relation $\models$

- I: an interpretation
- σ: an element in $\Sigma_\perp$

We define that $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models^I B)$.

Partial Correctness Assertions

Satisfaction Relation $\models$

- I: an interpretation
- σ: an element in $\Sigma_\perp$

We define that $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models^I B)$.

Validity

- Define that $\models^I \{A\}c\{B\}$ iff $\forall \sigma \in \Sigma_\perp . \sigma \models^I \{A\}c\{B\}$.

Partial Correctness Assertions

Satisfaction Relation $\models$

- I: an interpretation
- σ: an element in $\Sigma_\perp$

We define that $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models^I B)$.

Validity

- Define that $\models^I \{A\}c\{B\}$ iff $\forall \sigma \in \Sigma_\perp . \sigma \models^I \{A\}c\{B\}$.
- Define that $\models \{A\}c\{B\}$ iff $\forall I. \models^I \{A\}c\{B\}$.

Partial Correctness Assertions

Satisfaction Relation $\models$

- I: an interpretation
- σ: an element in $\Sigma_\perp$

We define that $\sigma \models^I \{A\}c\{B\}$ iff $(\sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!](\sigma) \models^I B)$.

Validity

- Define that $\models^I \{A\}c\{B\}$ iff $\forall \sigma \in \Sigma_\perp . \sigma \models^I \{A\}c\{B\}$.
- Define that $\models \{A\}c\{B\}$ iff $\forall I. \models^I \{A\}c\{B\}$.
- The partial correctness assertion $\{A\}c\{B\}$ is valid if $\models \{A\}c\{B\}$.

Partial Correctness Assertions

Validity

We have $\models \{A\}c\{B\}$ *holds iff for all interpretations* I *and all states* σ, *if* σ *satisfies* A *in* I *and the execution of* c *terminates in* σ' *from* σ, *then* σ' *satisfies* B *in* I.

Partial Correctness Assertions

Validity

We have $\models \{A\}c\{B\}$ *holds iff for all interpretations* I *and all states* σ, *if* σ *satisfies* A *in* I *and the execution of* c *terminates in* σ' *from* σ, *then* σ' *satisfies* B *in* I.

Recall: Validity for **Assn**

- A is valid: $\models A$ iff for all interpretations I and all states σ, $\sigma \models^I A$.

Partial Correctness Assertions

Validity: Examples

- $\{i \leq X\}X := X + 1\{i \leq X\}$ is valid.
- $\{i \leq X\}X := X - 1\{i \leq X\}$ is not valid.

Partial Correctness Assertions

Validity: Examples

- $\{i \leq X\}X := X + 1\{i \leq X\}$ is valid.
- $\{i \leq X\}X := X - 1\{i \leq X\}$ is not valid.

Validity and Extension Sets

- $\models A \Rightarrow B$ iff $A^I \subseteq B^I$ for all interpretations I.
- $\models \{A\}c\{B\}$ iff $\mathcal{C}[\![c]\!](A^I) \subseteq B^I$ for all interpretations I.

Proof Rules for Partial Correctness Assertions

Textbook, Page 89 – 93

Proof Rules

Motivation

- Manual validation of the validity $\models \{A\}c\{B\}$ is tedious.
- Rules for deriving the validity $\models \{A\}c\{B\}$ makes the task easier.

Proof Rules

Hoare Rules

- rules for each type of commands
- derivation trees built from rule instances
- correctness for each rule

Hoare Rules

Skip

$$\frac{}{\{A\}\textbf{skip}\{A\}}$$

Hoare Rules

Assignment

$$\overline{\{B\,[a/X]\}X := a\{B\}}$$

Hoare Rules

Sequencing

$$\frac{\{A\}c_0\{C\}\ ,\ \{C\}c_1\{B\}}{\{A\}c_0; c_1\{B\}}$$

Hoare Rules

Conditional Branch

$$\frac{\{A \wedge b\}c_0\{B\}\ ,\ \{A \wedge \neg b\}c_1\{B\}}{\{A\}\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\{B\}}$$

Hoare Rules

While Loop

$$\frac{\{A \wedge b\}c\{A\}}{\{A\}\textbf{while } b \textbf{ do } c\{A \wedge \neg b\}}$$

- A: the loop invariant

Hoare Rules

Consequence

$$\frac{\models A \Rightarrow A' \ , \ \{A'\}c\{B'\} \ , \ \models B' \Rightarrow B}{\{A\}c\{B\}}$$

Hoare Rules

The Proof System

- proofs as derivation trees
- theorems as conclusions
- notation for theorems: $\vdash \{A\}c\{B\}$

Summary

- extended arithmetic and boolean assertions
- partial correctness assertions
- a proof system from Hoare rules

Homework

No Homework This Week!