Axiomatic Semantics and Hoare Logic
Axiomatic semantics

central idea: describe the meaning of a program by making logical assertions (i.e., using axioms and predicate logic) that must be satisfied

a program's specification consists of assertions about pre-conditions and post-conditions that must hold around its execution
Hoare Triple

preconditions $\mu$

program $\mu$

postconditions $\mu$

$\{ P \} C \{ Q \}$

"If $P$ holds before $C$, and $C$ terminates, then $Q$ holds afterwards."

i.e., statement of partial correctness

$[P] C [Q]$

"If $P$ holds before $C$, then $C$ will terminate, and $Q$ holds afterwards."

statement of total correctness
On termination

- we will focus on partial correctness and assume that termination can be proved separately

- however, for a program without loops,

\[
\{ P \cup C \cup Q \} \leftrightarrow [P] C [Q]
\]
Language of Assertions

- predicate logic
- arithmetic and logical expressions
- program variables — i.e., in the program store $\sigma$
  - we say assertion $P$ is valid in store $\sigma$: $\Delta \models P$
  - if $P$ is valid in any store, we simply say $P$ holds: $\models P$
  - if $\{P \triangleright c \triangleleft Q\}$, then $\sigma \models P$ and $\langle c, \sigma \rangle \downarrow \sigma' \rightarrow \sigma' \models Q$
e.g., \{ x = 0 \} x := x + 1 \{ x = 1 \}? \checkmark

\{ x = 0 \} x := x + 1 \{ x < 0 \}? \times

\{ x = 0 \} x := x + 1 \{ \text{false} \}? \times

\checkmark \text{i.e., program doesn't terminate}

\text{what else would be true?}\n
\{ x > 0 ? \}

\{ \text{true} ? \}
"Simple Imperative Programming Language" (IMP)

arithmetic exps: \( E ::= \text{Integer} \mid \text{Var} \mid E + E \)

boolean exps: \( B ::= \text{true} \mid \text{false} \mid E \sim E \)

statements: \( S ::= \text{skip} \)
\( \mid \text{Var} ::= E \)
\( \mid S_1; S_2 \)
\( \mid \text{if} B \text{ then } S_1 \text{ else } S_2 \)
\( \mid \text{while} B \text{ do } S \)
Axiom: skip

\{P \models \text{skip} \\{P \}\}

e.g., \{x = 1 \models \text{skip} \\{x = 1 \}\}
Axiom: assignment

substitute e for x in the predicate

\[
\{ [e/x]P \} \ x := e \ \{ P \}
\]

e.g. \( \{ y > 10 \} x := y \ \{ x > 10 \} \)

\( \{ y + z > 100 \} x := y + z \ \{ x > 100 \} \)
e.g., derive a precondition $P$, given that $\models \{ P \} C \{ Q \}$, where $C$ is "$x := y \times 2$" and $Q \equiv x < 10$

\[
\{ ?, ? \} x := y \times 2 \{ x < 10 \}
\]

\[
y < 5 \quad \leftarrow \text{the least restrictive precondition possible}
\]

will any other preconditions do?
- any more restrictive one! \[ y < 0 \]
\[ y = -1 \]
\[ y < 5 \land z > 10 \]
Rule: Sequencing

Do these absolutely need to be identical for the conclusion to be true?

\[
\begin{align*}
\{p \lor s, \{r \lor s\}, \{p \lor s\}, s, r < \{q, s, p, s_2 \} \leq q_3
\end{align*}
\]
e.g., derive a precondition $P$, given that $\models \{P \land C \land \}$:

$\{P \land y = 20 \land x = 10\}$

$\begin{align*}
  w &:= x; \\
  x &:= y; \\
  y &:= w;
\end{align*}$

$\{y = 20 \land w = 10\}$

$\{x = 20 \land w = 10\}$

$\{Q \equiv x = 20 \land y = 10\}$
Rule: conditional

\[
\{ P \land b \} \mathcal{S}_1 \{ Q \} , \{ P \land \neg b \} \mathcal{S}_2 \{ Q \}
\]

\[
\exists P ? \text{ if } b \text{ then } \mathcal{S}_1 \text{ else } \mathcal{S}_2 \{ Q \}
\]

- very clever way to think about "branches!"
- both if + else clause work together to ensure \( Q \) holds

\( \text{vs. } N \text{ if-else statements } \rightarrow 2^N \text{ possible outcomes} \)
Given $\{P \land Q\}$, derive a precondition $P$ where $C$ is "if $x < 0$ then $y := y + 1$ else $y := x$" and $Q \equiv y > 0$

\[
\begin{align*}
\{P \land x < 0\} & \Rightarrow \{y > 0\} \\
\{P \land x \geq 0\} & \Rightarrow \{y > 0\} \\
\{y > -1 \land x < 0\} & \Rightarrow \{y > 0\} \\
\{x > 0 \land x \geq 0\} & \Rightarrow \{y > 0\}
\end{align*}
\]

\[P \equiv (y > -1 \land x < 0) \lor (x > 0)\]
Rule: "Consequence"

\[ P \implies P', \{P'\} \subseteq \{Q'\}, Q' \implies Q \]
\[ \{P\} \subseteq \{Q\} \]

This allows us to update the pre/post-conditions without changing the conclusion's validity.
Strong vs. Weak Assertions

- we say $P$ is stronger than $Q$ if $P \Rightarrow Q$ (Q is weaker than P)

- intuitively, $P$ is stronger than $Q$ if $P$ is more restrictive

e.g., "$x$ is a dog" is stronger than "$x$ is a pet"

$x > 10$ is weaker than $x > 10 \land x$ is prime

recall: $P \Rightarrow Q \equiv \neg P \lor Q$
"Weakest" Precondition

The weakest precondition $\text{WP}(C, Q)$ is an assertion that describes the set of all states $S$ s.t. if $C$ is run in $\sigma \in S$ and terminates, its termination state will satisfy $Q$.

\[ \sigma \xrightarrow{C} \sigma' \]

$\text{WP}(C, Q)$

all states

$\sigma'$

all states

too strong!
e.g., rank the following assertions from strongest to weakest:

true
false
false

\((x > 100) \lor (y < 25)\)
\((x > 100)\)
\((x > 50) \lor (y < 10)\)
\((x > 50)\)

\((x > 50) \land (y < 10)\)
\((x > 100) \lor (y < 25)\)
\((x > 50)\)

\((x > 50)\)
\((x > 100)\)

true
Strong vs. Weak Assertions

Observations:

- When we add a conjunction to an assertion, we **strengthen it**
- When we add a disjunction to an assertion, we **weaken it**
- The strongest assertion is **false**
- The weakest assertion is **true**
Revisiting Rule: “Consequence”

“strengthen” the precondition

“weaken” the postcondition

\[
P \rightarrow P', \{P'\} C \{Q'\}, Q' \rightarrow Q
\]

\[
\{P\} C \{Q\}
\]

and this will still hold
e.g., suppose \( \{x > 0\} \cap \{y < 0\} \). Which will hold?

1. \( \{x > 0\} \cap \{y < 0 \lor x > 0\} \) ✓
2. \( \{x > 0 \land y < 0\} \cap \{y < 0\} \) ✓
3. \( \{y < 0\} \cap \{x > 0\} \) X
4. \( \{x > 0\} \cap \{y < 0 \land x > 0\} \) X
5. \( \{x > 0 \lor y < 0\} \cap \{y < 0\} \) X
6. \( \{x > 0\} \cap \{y < 10\} \) ✓
7. \( \{x > -10\} \cap \{y < 0\} \) X
Rule: loop

preserved across each loop iteration
i.e., loop "invariant"

\[ \{ P \land b \} \leq \{ P \} \]

\[ \{ P \} \text{ while } b \text{ do } \leq \{ P \land \neg b \} \]

also true after loop terminates
Reasoning about Loops

**Questions:**
- How is the loop invariant (P) established?
- What is the "purpose" of the loop?
- How can we guarantee that the loop terminates?

**Partial Correctness**

- \{P\}
- \{P \land b\}
- S
- \{P\}
- \{P \land \neg b\}

**Total Correctness**

(Not: ultimately undecidable \(
\rightarrow\)
halting problem \(),
but amenable to other proof methods.)
Additional Pre and Post Conditions

\[ \{R\} S_{\text{init}} \{P\} \]

while \( b \)
\[ \{P \land b\} S_{\text{body}} \{P\} \]
\[ \{P \land \neg b\} \]
\[ \{Q\} \]

updated assertions:
1. \( \{R\} S_{\text{init}} \{P\} \)
2. \( \{P \land b\} S_{\text{body}} \{P\} \)
3. \( P \land \neg b \rightarrow Q \)

establish invariant
loop goal: stronger version of \( P \)
loop must eventually make \( b \) false
e.g., consider the program

\[
\begin{align*}
  f &:= 1; \\
  i &:= 1; \\
  \text{while } i \leq N \text{ do} \\
  & \quad f := f \times i; \\
  & \quad i := i + 1
\end{align*}
\]

- what is the loop goal? \\
- what is a reasonable loop invariant? \\
- validate the assertions:
  1. \{R \}\ Sinit \{P\}
  2. \{P \land b\}\ Sbody \{P\}
  3. \ P \land \neg b \rightarrow Q
e.g., consider the program

\[
\begin{align*}
f &:= 1; \quad \{ \text{Sinit} \} \\
i &:= 1; \quad \{ \text{Sinit} \} \\
\text{while } i \leq N \text{ do} \\
f &:= f \times i; \quad \{ \text{Sbody} \} \\
i &:= i + 1 \quad \{ \text{Sbody} \}
\end{align*}
\]

- what is the loop goal?
  \( Q = f = N! \)
- what is a reasonable loop invariant?
  \( P = f = (i-1)! \land i \leq N+1 \)

1. \( \{ R = \text{true} \} \quad f := 1; \quad i := 1 \quad \{ P \} \quad \checkmark \)
2. \( \{ P \land i \leq N \} \subseteq \text{body} \{ P \} \quad \checkmark \)
3. \( P \land i > N \rightarrow Q \quad \checkmark \)

\[ f = (i-1)! \land i = N+1 \]
e.g., consider the program

\begin{verbatim}
S := 0;
i := 0;
while i < len(arr) do
  S := S + arr[i];
i := i + 1
\end{verbatim}

- what is the loop goal?
- what is a reasonable loop invariant?
- validate the assertions:
  1. \{R \land Sinit \land P\}
  2. \{P^b \land Sbody \land P\}
  3. P^b \land b \rightarrow Q
e.g., consider the program

```
s := 0;  ? Sinit
i := 0;  
while i < len(arr) do
  s := s + arr[i];  
  Sbody
    i := i + 1
```

- what is the loop goal?

\[ Q \equiv s = \sum_{j=0}^{\text{len(arr)}-1} arr[j] \]

- what is a reasonable loop invariant?

\[ P \equiv s = \sum_{j=0}^{i-1} arr[j] \land i \leq \text{len(arr)} \]

1. \{R=true\} Sinit \{P\} \checkmark
2. \{P \land i \leq \text{len(arr)}\} \leq \text{Sbody} \{P\} \checkmark
3. \overset{\text{P} \land i \geq \text{len(arr)} \rightarrow Q \checkmark}{\text{i} = \text{len(arr)}}
Finding loop invariants

- A (correct) loop invariant lets us prove the correctness of a loop!
- But how to pick it?
- Recall: $P \land \neg b \Rightarrow Q$; i.e., the loop invariant is a weaker version of the postcondition to weaken:
  - Add a disjunction
  - Remove a conjunction
  - Replace a constant with a range

- Simplest technique: guess + check
- Automated generation of loop invariants is an active research area!