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

$$\{\overset{\text{preconditions}}{P}\}\ \overset{\text{program}}{C}\ \{\overset{\text{postconditions}}{Q}\}$$

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

i.e., statement of <u>partial</u> correctness

$$[P]\ C\ [Q]$$

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

statement of <u>total</u> correctness

On termination

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

- however, for a program without loops,

$$\{P\}\,C\,\{Q\} \longleftrightarrow [P]\,C\,[Q]$$

Language of Assertions

- predicate logic
  - true, false
  - $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$
  - $\forall$, $\exists$
- arithmetic & logical exps
- program variables — i.e., in the program store σ
  - we say assertion P is valid in store σ : $\sigma \models P$
  - if P is valid in any store, we simply say P holds : $\models P$
  - if $\{P\}\, c\, \{Q\}$, then $\sigma \models P$ and $\langle c, \sigma \rangle \Downarrow \sigma' \rightarrow \sigma' \models Q$

what else would be true?

$\{x > 0\}$

$\{true\}$

e.g., $\{x = 0\}\ x := x + 1\ \{x = 1\}$ ✓

$\{x = 0\}\ x := x + 1\ \{x < 0\}$ ✗

$\{x = 0\}\ x := x + 1\ \{false\}$? ✗

i.e., program doesn't terminate

"Simple Imperative Programming Language" (IMP)

arithmetic exps: $E ::= \text{Integer} \mid \text{Var} \mid E \oplus E$

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

$$\{P\}\ \text{skip}\ \{P\}$$

e.g., $\{x=1\}\ \text{skip}\ \{x=1\}$

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 "$x := y * 2$" and $Q \equiv x < 10$

$$\{ ? \}\ x := y * 2\ \{x < 10\}$$

$y < 5$ ← the least restrictive precondition possible

will any other preconditions do?

- any more restrictive one!

$y < 0$

$y = -1$

$y < 5 \wedge z > 10$

Rule: Sequencing

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

$$\frac{\{P\}\, S_1\, \{R\}\, ,\quad \{R\}\, S_2\, \{Q\}}{\{P\}\, S_1\, ;\, S_2\, \{Q\}}$$

e.g., Derive a precondition P, given that $\models \{P\}\, c\, \{Q\}$:

$$\{P \equiv y = 20 \wedge x = 10\}$$

$$c \left\{ \begin{array}{l} w := x; \\ \qquad \{y = 20 \wedge w = 10\} \\ x := y; \\ \qquad \{x = 20 \wedge w = 10\} \\ y := w; \end{array} \right.$$

$$\{Q \equiv x = 20 \wedge y = 10\}$$

Rule: conditional

same post-condition regardless of branch!

$$\frac{\{P \wedge b\}\ s_1\ \{Q\}, \quad \{P \wedge \neg b\}\ s_2\ \{Q\}}{\{P\}\ \text{if } b \text{ then } s_1 \text{ else } s_2\ \{Q\}}$$

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

(vs. N if-else statements $\longrightarrow 2^N$ possible outcomes) e.g., Given $\models \{P\}\, C\, \{Q\}$, derive a precondition P ← try to make P unrestrictive

where C is "if $x < 0$ then $y := y+1$ else $y := x$"

and $Q \equiv y > 0$

$$\{\ ?\ \}\ \text{if } x<0 \text{ then } y:=y+1 \text{ else } y:=x\ \{y>0\}$$

$$\{P \wedge x<0\}\ y:=y+1\ \{y>0\} \qquad \{P \wedge x \geq 0\}\ y:=x\ \{y>0\}$$

$$\{y>-1 \wedge x<0\}\ y:=y+1\ \{y>0\} \qquad \{x>0 \wedge x \geq 0\}\ y:=x\ \{y>0\}$$

$$P \equiv (y>-1 \wedge x<0) \vee (x>0)$$

Rule: "Consequence"

implication

$$\frac{P \rightarrow P', \ \{P'\}\, c\, \{Q'\}, \ Q' \rightarrow Q}{\{P\}\, c\, \{Q\}}$$

- this allows us to update the pre/post-conditions without changing the conclusion's validity

Strong vs. Weak Assertions

recall: $P \to Q \equiv \neg P \vee Q$

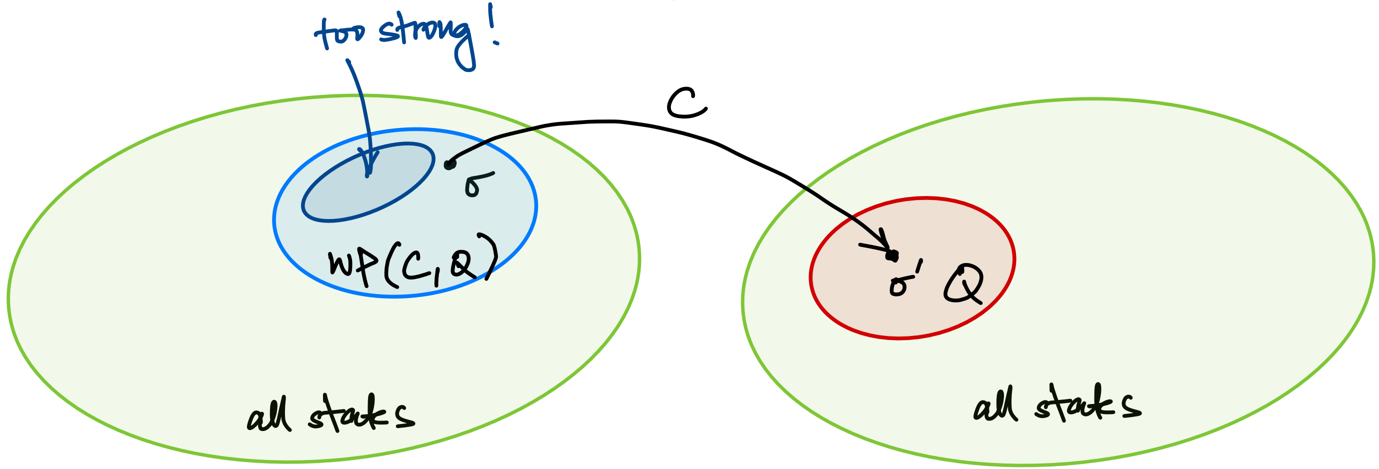
- we say P is stronger than Q if $P \to Q$
  $\equiv$
  (Q is weaker than P)
- intuitively, P is stronger than Q if P is more restrictive

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

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

"Weakest" Precondition

the weakest precondition $WP(C, Q)$ 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



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

true

false

$(x > 100) \vee (y < 25)$

$(x > 100)$

$(x > 50) \vee (y < 10)$

$(x > 50) \wedge (y < 10)$

$(x > 50)$

$$
\begin{tikzcd}
\text{false} \arrow[d] \arrow[dr] & \\
(x > 50) \wedge (y < 10) \arrow[d] \arrow[dr] & (x > 100) \arrow[dl] \arrow[d] \\
(x > 50) \arrow[d] & (x > 100) \vee (y < 25) \arrow[ddl] \\
(x > 50) \vee (y < 10) \arrow[d] & \\
\text{true} &
\end{tikzcd}
$$

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"

$$\frac{\overbrace{P \to P'}^{\text{"strengthen" the precondition}},\ \{P'\}\ c\ \{Q'\},\ \overbrace{Q' \to Q}^{\text{"weaken" the postcondition}}}{\underbrace{\{P\}\ c\ \{Q\}}_{\text{and this will still hold}}}$$

e.g., suppose $\vDash \{x>0\}\ c\ \{y<0\}$. which will hold?

① $\{x>0\}\ c\ \{y<0 \vee x>0\}$ ✓

② $\{x>0 \wedge y<0\}\ c\ \{y<0\}$ ✓

③ $\{y<0\}\ c\ \{x>0\}$ ✗

④ $\{x>0\}\ c\ \{y<0 \wedge x>0\}$ ✗

⑤ $\{x>0 \vee y<0\}\ c\ \{y<0\}$ ✗

⑥ $\{x>0\}\ c\ \{y<10\}$ ✓

⑦ $\{x>-10\}\ c\ \{y<0\}$ ✗

Rule: loop

preserved across each loop iteration

i.e., loop "invariant"

$$\frac{\{P \wedge b\}\ s\ \{P\}}{\{P\}\ \text{while } b \text{ do } s\ \{P \wedge \neg b\}}$$

also true after loop terminates

Reasoning about Loops

$\{P\}$
while b
$\quad\{P \wedge b\}$
$\quad S_{body}$
$\quad\{P\}$
$\{P \wedge \neg b\}$

partial correctness

questions:

- how is the loop invariant (P) established?
- what is the "purpose" of the loop?

- how can we guarantee that the loop terminates?

total correctness

(note: ultimately undecideable → halting problem; but amenable to other proof methods)

Additional Pre+Post Conditions

$\{R\}$
S_{init}
$\{P\}$
while b
$\quad\{P \wedge b\}$
$\quad S_{body}$
$\quad\{P\}$
$\{P \wedge \neg b\}$
$\{Q\}$

updated assertions:

1. $\{R\}\, S_{init}\, \{P\}$ establish invariant
2. $\{P \wedge b\}\, S_{body}\, \{P\}$
3. $P \wedge \neg b \rightarrow Q$ loop goal: stronger version of P

loop must eventually make b false

e.g., consider the program

```
f := 1 ;
i := 1 ;
while i ≤ N do
    f := f × i ;
    i := i + 1
```

- what is the loop goal?
- what is a reasonable loop invariant?
- validate the assertions:

1. $\{R\}\ S_{init}\ \{P\}$
2. $\{P \wedge b\}\ S_{body}\ \{P\}$
3. $P \wedge \neg b \rightarrow Q$

e.g., consider the program

$$\begin{aligned}
&\left.\begin{array}{l} f := 1; \\ i := 1; \end{array}\right\} S_{init} \\
&\text{while } i \le N \text{ do} \\
&\quad \left.\begin{array}{l} f := f \times i; \\ i := i + 1 \end{array}\right\} S_{body}
\end{aligned}$$

– what is the loop goal?

$Q \equiv f = N!$

– what is a reasonable loop invariant?

$P \equiv f = (i-1)! \land i \le N+1$

1. $\{R \equiv true\}\ f := 1;\ i := 1\ \{P\}$ ✓

2. $\{P \land i \le N\}\ S_{body}\ \{P\}$ ✓

3. $\underbrace{P \land i > N}_{f = (i-1)! \land i = N+1} \rightarrow Q$ ✓

e.g., consider the program

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

- what is the loop goal?
- what is a reasonable loop invariant?
- validate the assertions:

1. $\{R\}\ S_{init}\ \{P\}$
2. $\{P \wedge b\}\ S_{body}\ \{P\}$
3. $P \wedge \neg b \rightarrow Q$

e.g., consider the program

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

$\left.\begin{array}{l} s := 0; \\ i := 0; \end{array}\right\}$ Sinit

Sbody $\left\{\begin{array}{l} s := s + arr[i]; \\ i := i + 1 \end{array}\right.$

– what is the loop goal?

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

– what is a reasonable loop invariant?

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

1. $\{R \equiv true\}\ S_{init}\ \{P\}$ ✓
2. $\{P \wedge i \leq len(arr)\}\ S_{body}\ \{P\}$ ✓
3. $\underbrace{P \wedge i \geq len(arr)}_{\rightarrow\ i = len(arr)} \rightarrow Q$ ✓

Finding loop invariants

- a (correct) loop invariant lets us prove the correctness of a loop!
  - but how to pick it?
- recall: $P \wedge \neg b \rightarrow Q$; i.e., the loop invariant is a weaker version of the postcondition

to weaken:
- add a disjunction
- remove a conjunction
- replace a constant w/ a range

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