17 Axiomatic Semantics

Monday, November 6, 2023 11:10 AM

- Hoare Logic

'69 C.A.R Hoare.
"Tony"

What? :- a Formal system of Logic Rules to reason rigurously about programs.

Objectives.- - Give meaning to a program
- Prove that a program has certain properties
- Prove correctness of a program

Dijksta, Wirth, Knuth.

Program is a complex math object.

- Review of Mathematical Logic.

Logic: the mathematics of correct reasoning.
- true or false.
- evaluate sentences / formulas to know whether they are true or false.

- PROPOSITIONAL LOGIC

Boolean logic / Circuit logic.

- Propositions.

p q r s

propositions are assigned true / false

p = true
q = false
⋮

- Logical Connectives

∧ ∨ ¬ ⊗ →

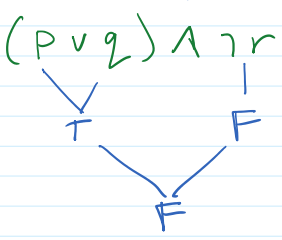
The meaning of logical connectives is usually stated via truth tables.

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

- Propositional Formulas.

are evaluated.



p = true
q = false
r = true

- Inference Rules: rules of pre-evaluated formulas.

premises

- Inference Rules: rules of pre-evaluated formulas.

Premises → $\frac{S_1\ S_2\ \dots\ S_3}{C_0}$ ← Conclusion.

If all of S_n are true then C_0 is true

modus ponens.

$\frac{}{p \vee \neg p}$ $\qquad$ $\frac{p \wedge q \quad q}{p}$ $\qquad$ $\frac{p \rightarrow q \quad p}{q}$

Limitations of propositional Logic:

"All men are mortal, Aristotle is a man therefore Aristotle is mortal"

PREDICATE LOGIC

- Objects.
  - a b aristole
  - 1 7 garfield.
- Variables over objects.
  - X Y Z
- predicates.-
  - represent properties or relationships between objects
  - have an arity, fixed number of arguments.

red(a) man(aristotle)

odd(7) friend(aristotle, garfield)

$1 < 7$ = lessthan(1, 7)

even(X) red(Y) friend(aristotle, Z)

"Ground" predicates (Predicates without variables) are assigned true/false.

man(aristotle) = true friend(aristotle, garfield) = false.

red(a) = false orange(garfield) = true.

- Logical connectives.
  - $\vee$ $\wedge$ $\neg$ $\rightarrow$ $\leftrightarrow$
- Quantifiers
  - $\forall$: universal
  - $\exists$: existencial

$\forall X\ p(X)$: true if p is true for every object in the domain.

$\exists X\ p(X)$: true if p is true for an object in the domain.

- Predicate Formulas:
  - Objects, Variables
  - Logical operators, Quantifiers,

  Note: All variables should be quantified.

  - $\forall X \exists Y \; p(X) \rightarrow q(Y)$ $\qquad \exists X \forall Y \; \text{lessthan}(X, Y) \rightarrow \text{odd}(Y)$

  objects : (aristotle, garfield)

  man(aristotle).

  $\exists X \; \text{man}(X) = \text{True}.$

  $\forall X \; \text{man}(X) = \text{False}.$

- Rules of Inference.

$$\frac{\text{Precondition(s)}}{\text{conclusion}} \qquad \frac{p(a) \; ; \; p(b)}{p(a) \vee p(b)} \qquad \frac{\forall X \; p(X)}{p(c)} \qquad \frac{p(c)}{\exists X \; p(X)}$$

$$\frac{\forall X \; p(X) \rightarrow q(X) \; ; \; p(c)}{q(c)}$$

Power of Predicate Logic.

1) All men are mortal
2) and aristotle is a man

therefore

aristotle is mortal

objects (aristotle, garfield)

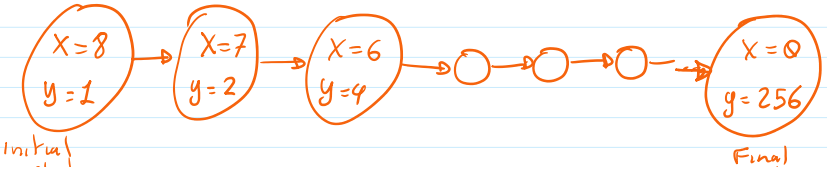
1) $\forall X \; \text{man}(X) \rightarrow \text{mortal}(X) . = \text{True}$
2) man(aristotle). = True

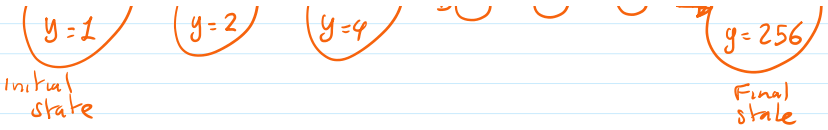
   by inference Rule
3) mortal(aristotle) = true. $\qquad$ mortal(garfield) = False.

HOARE LOGIC

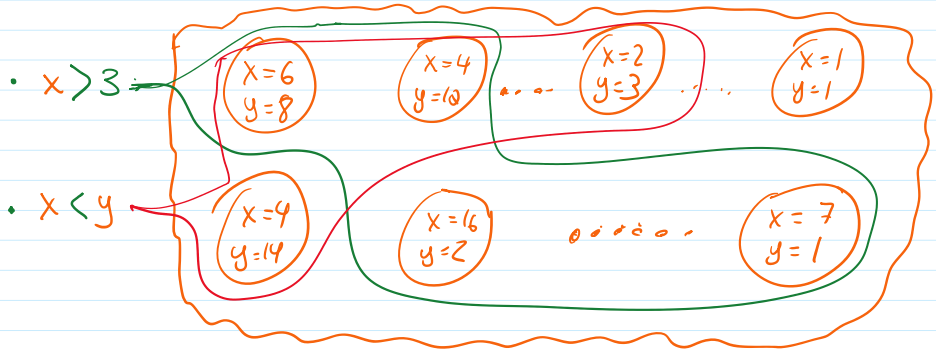
- Objects :- Program Variables.
- State :- An assignment of values to variables.
- Program execution :-

  A sequence of transitions from an initial state to a final state.





- Predicate Formulas in Hoare Logic
  - Not to be evaluated true/false
  - Objects Program Variables.
  - Represent the set of all states over which the formula is true.



- Formulas in Hoare Logic: the Hoare triple

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

P, Q: predicate Formulas — P: "Precondition" Q: "Postcondition"

C: "command"; a piece of program code.

- In English: if I tell you $\{P\}C\{Q\}$ is true.

what I am saying is that

If you execute C in a state in P
it will lead to a state in Q

- How is this useful:

C: your Program
$\{P\}$: spec of the input to your program.
$\{Q\}$: spec of the required output of your program.

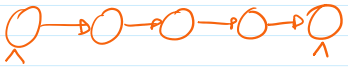
If you can prove $\{P\}C\{Q\}$ is true.
you just proved that your program is correct!

Implications:

- A program can be analysed mathematically
- Use mathematics to prove properties of programs
- not the most popular approach. 😖

SEMANTICS THROUGH LOGIC RULES

collection of axioms and inference rules : { • assignments • conditionals • loops



- Rules of inference of Hoare Logic
  - Axiom of assignment.

$$\{Q_{X \to E}\}\ X \leftarrow E\ \{Q\}$$

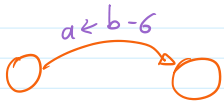
replace every occurance of x in Q by E

Example.

$$\{?\}\ a \leftarrow b-6\ \{a > 0\}$$

$$\{b-6 > 0\}$$

$$\{b > 6\}\ a \leftarrow b-6\ \{a > 0\}$$ true!



Eg-2

$$\{?\}\ a \leftarrow 2\cdot(b-2)\ \{\underbrace{0 \le a < 10}_{Q}\}$$

$$\{0 \le 2\cdot(b-2) < 10\}$$

$$0 \le 2b-4 < 10$$

$$4 \le 2b < 14$$

$$\{2 \le b < 7\}\ a \leftarrow 2\cdot(b-2)\ \{0 \le a < 10\} = \text{true!}$$

- Rule of composition

$$\frac{\{P\}\ C_1\ \{R\},\ \{R\}\ C_2\ \{Q\}}{\{P\}\ C_1 ; C_2\ \{Q\}}$$

Example:

$$\{?\}\ a \leftarrow 3*b-1\ ;\ b \leftarrow 4*a-22\ \{b > 10\}$$

$$\{?\}\ b \leftarrow 4*a-21\ \{b > 10\}$$

$$\{4\cdot a-22 > 10\}$$

$$\{a > \tfrac{10+22}{4}\}$$

- $\{a > 8\}\ b \leftarrow 4*a-21\ \{b > 10\} = \text{true.}$

$$\{?\}\ a \leftarrow 3*b-1\ \{a > 8\}$$

$$\{3\cdot b-1 > 8\}$$

$$\{3b > 9\}$$

$\{?\}\ a \leftarrow 3*b-1\ \{a>8\}$

$\{3\cdot b-1>8\}$

$\{3b>9\}$

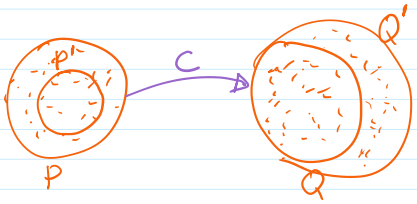
- $\{b>3\}\ a \leftarrow 3*b-1\ \{a>8\}$ = true.

by rule of composition

$\{b>3\}\ a \leftarrow 3*b-1\ ;\ b \leftarrow 4*a-22\ \{b>10\}$

Rule of consequence:

Intuition.



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

- Restricting the Precondition.
- Relaxing the Postcondition

$$\frac{P' \subseteq P,\ \{P\}C\{Q\},\ Q \subseteq Q'}{\{P'\}C\{Q'\}}$$

E.G.

$\{b<7\}\ a \leftarrow 2\cdot(b-2)\ \{a<10\}$ = true.

- $\{b<0\} \subseteq \{b<7\}$ = true
- $\{a<10\} \subseteq \{a<20\}$ = true.

$\therefore\ \{b<0\}\ a \leftarrow 2\cdot(b-2)\ \{a<20\}$ = true.

Conditional Rule:

$$\frac{\{B \wedge P\}\ C_1\ \{Q\};\ \{\neg B \wedge P\}\ C_2\ \{Q\}}{\{P\}\ \text{IF } B \text{ THEN } C_1 \text{ ELSE } C_2\ \{Q\}}$$

E.G.

Prove:

$\{0 \le x \le 12\}$ IF $x<12$ THEN $x \leftarrow x+1$ ELSE $x \leftarrow 0$ $\{0 \le x \le 12\} \equiv$ True.

P Q

We need to prove:

1) $\{0 \le x \le 12 \wedge x<12\}\ x \leftarrow x+1\ \{0 \le x \le 12\} \equiv$ True.

P B Q

$\{0 \le x < 12\}$

2) $\{0 \le x \le 12 \wedge \neg(x<12)\}\ x \leftarrow 0\ \{0 \le x \le 12\} \equiv$ True.

P $\{x=12\}$ $\neg B$ Q

Prove #1

by Axiom of assignment.

$\{0 \le x+1 \le 12\}\ x \leftarrow x+1\ \{0 \le x \le 12\} \equiv$ True

$\{-1 \le x \le 11\}$

- $\{0 \le x < 12\} \subseteq \{-1 \le x \le 11\}$ true.

$\{-1 \le x \le 11\}$

- $\{0 \le x < 12\} \subseteq \{-1 \le x \le 11\}$ true.

$\therefore$ #1 ✓

Prove #2
by Axiom of assignment

$\{0 \le 0 \le 12\}\ x \leftarrow 0\ \{0 \le x \le 12\}$ · True

$\{\text{True}\}$

- $\{x = 12\} \subseteq \{\text{True}\}$ · true

WHILE RULE

$$\frac{\{B \wedge P\}\ C\ \{P\}}{\{P\}\ \text{WHILE } B \text{ DO } C\ \{\neg B \wedge P\}}$$

P: Loop invariant

- You want an invariant for which $\{\neg B \wedge P\}$ tells you something

E.G

$\{x \le y\}$ WHILE $x < y$ DO $x \leftarrow x+1$ $\{\neg(x<y) \wedge x \le y\}$

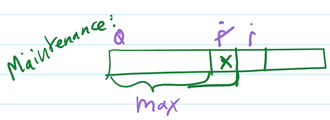
(P = $x \le y$; $\neg B$, P: $\{x = y\}$)

Invariant Properties

- Initialization :- Invariant is true before the loop
- Maintenance :- Invariant is true before any iteration, after completion of an iteration.
- Termination :- Invariant is true after the loop is completed.

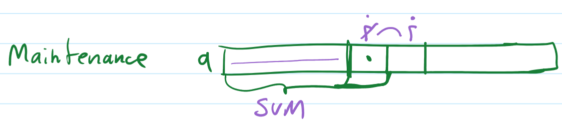
E.G.

```
FUNCTION max(a[0..n-1])
   max ← -∞
   i ← 0
   {P: max is the maximum value of a[0..i-1]}
   WHILE i < n DO
      IF a[i] > max THEN
         max ← a[i]
      i ← i+1
   {P ∧ ¬B: max is the maximum value of a[0..i-1] AND i = n}
   RETURN max  ⇒ max is maximum value of a[0..n-1]
```



E.G.

```
FUNCTION sum(a[0..n-1])
   sum ← 0;   i ← 0;
   {P:            sum = Σ_{0}^{i-1} a[i] }
   WHILE i < n DO
      sum ← sum + a[i]
      i ← i+1
   {P ∧ ¬B:  sum = Σ_{0}^{i-1} a[i]  AND  i = n
   RETURN sum.     ⇒     sum = Σ^{n-1} a[ ]
```



{P ∧ ¬B: $sum = \sum_{0}^{i-1} a[i]$ AND $i = n$
RETURN sum.

$\Rightarrow$ $sum = \sum_{0}^{n-1} a[\,]$

E.G.

```
FUNCTION  sort( a[0.. n-1] )
  j ← 0
  {P:  a[0..j]  is sorted
  WHILE j < n-1 DO
      k ← j+1
      { P: a[j] is the smallest of a[j.. k-1] }
      WHILE k < n DO
          IF a[k] < a[j] THEN
              swap (a[k], a[j])
          k ← k+1
      { P ∧ ¬B:  a[j] is the smallest of a[j...n-1] }
      j ← j+1
{P ∧ ¬B: a[0..n-1] is sorted}
```

Maintenance