Hoare Logic

Zhaoguo Wang

Adapted From:

ETH Zurich program verification, Lecture 6, 7
(https://www.pm.inf.ethz.ch/education/courses/program-verification.html)
<<Forward with Hoare>>
<<Weakest-Precondition of Unstructured Programs>>
Foundations of Programming Languages(https://cs.nju.edu.cn/xyfeng/teaching/FOPL, Hoare Logic)
<<Background reading on Hoare Logic>>, Mike Gordon

Hoare Logic （霍尔逻辑）

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first order logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Convert it into a Hoare logic formula

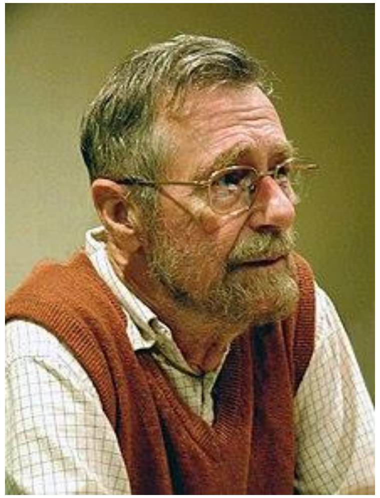
Step 2. Ask the computer to check the invariants

Loops

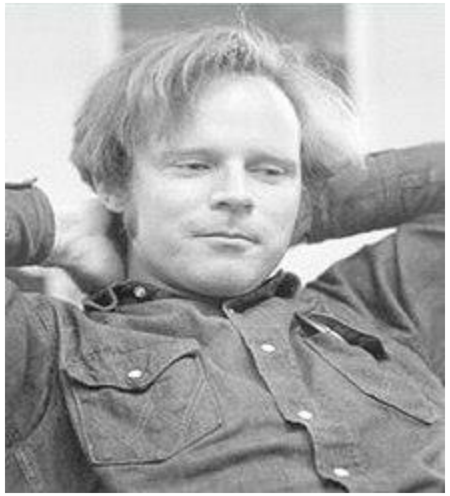
```
void f(unsigned a)
{
    unsigned i = 0;
    while(i < a)
    {
        i = i + 1;
    }
    assert(i == a);
}
```

Hoare Logic Can Do It!

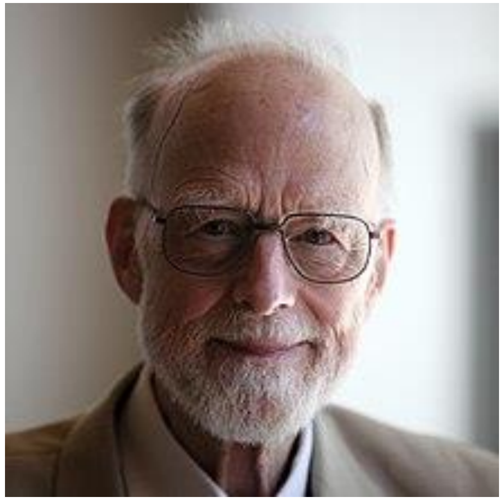
Assign logical meaning to programs



Edsger Wybe Dijkstra
Turing Award(1972)



Robert W. Floyd
Turing Award(1978)



Charles Antony Richard Hoare
Turing Award(1980)

C. A. R. Hoare:
An Axiomatic Basis for Computer Programming.
Commun. ACM 12(10): 576-580, 1969.

Problem1:

How to define program correctness?

Hoare Triples(霍尔三元组)

DEFINITION

Program state includes the value of every variable used in the program.

DEFINITION

Pre-condition describes the program state before executing the program.

DEFINITION

Post-condition describes the program state after executing the program.

pre-condition $x = 1$ $\boxed{x = x + 1;}$ $x = 2$ post-condition

Hoare Triples

DEFINITION

Program state includes the value of every variable used in the program.

DEFINITION

Pre-condition describes the program state before executing the program.

DEFINITION

Post-condition describes the program state after executing the program.

$$x = 3 \wedge y = 5$$

$$x > 3 \wedge x \neq 0$$

Pre-/post-condition can be predicate logic formulas.

Hoare Triples

DEFINITION

A program C is partially correct iff:

whenever C is executed in a state initially satisfying pre-condition and if the execution of C terminates, then the state in which C's execution terminates satisfies post-condition.

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

We use Hoare triples to define program correctness.

Hoare Triples

DEFINITION

A program C is partially correct iff:

whenever C is executed in a state initially satisfying pre-condition and if the execution of C terminates, then the state in which C's execution terminates satisfies post-condition.

pre-condition $\{P\}\ C\ \{Q\}$ post-condition

program

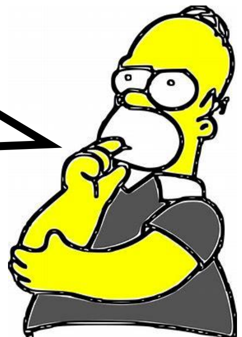
Hoare Triples

DEFINITION

A program C is partially correct iff:

whenever C is executed in a state initially satisfying pre-condition and if the execution of C terminates, then the state in which C's execution terminates satisfies post-condition.

We don't consider whether the given program can terminate (partially correct).



Halting Problem

DEFINITION

Halting problem means if we can have an algorithm that will tell that the arbitrary given program will halt or not.

Basically halting means terminating.

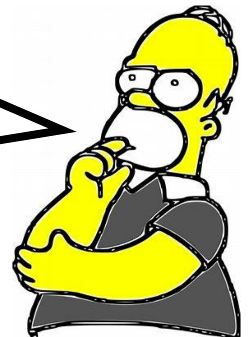
Alan Turing proved in 1936 that a general algorithm to solve the halting problem cannot exist.

Hoare Triples

The Hoare triple is the correctness criterion of program C, which is called specification.

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

P and Q are predicate logic propositions. What about C?



A Small Imperative Language

- Statements
  - skip (does nothing when executed)
  - a := e (assignment: changes value of a)
  - s1; s2 (sequential composition: execute s1 followed by s2)
  - if(b1) {s1} else {s2} (execute s1 if b is true; s2 otherwise)
  - While(b1) {s} (repeatedly execute s while b is true)
- We don't consider
  - heap, pointer, break, continue, function, bit-operator, object-oriented, struct, union, array, float numbers…

For simplicity, we just consider a simple imperative language.

Hoare Logic

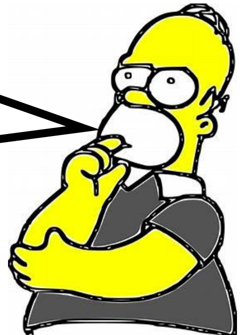
$$\{a = 0\}a := 3\{a = 3\}$$

$$\{a = 1\}a := 2; a := a + 1\{a = 3\}$$

$$\{a = 1\}a := 2\{a = 10\}$$

$$\{a = 1 \land b = 5\}if(a == 1)a := b\{a = 5 \land b = 5\}$$

They are all Hoare triples. But not all of them are valid.



Hoare Logic

$$\{a = 0\}a := 3\{a = 3\}$$

$$\{a = 1\}a := 2; a := a + 1\{a = 3\}$$

$$\{a = 1\}a := 2\{a = 10\}$$

$$\{a = 1 \wedge b = 5\}if(a == 1)a := b\{a = 5 \wedge b = 5\}$$

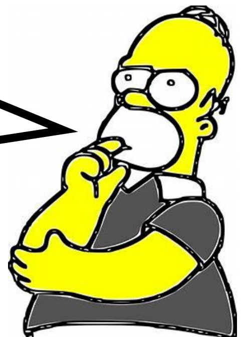
We need to formally prove whether a Hoare triple is right. In other words, we need to prove the program satisfies the specification.

Problem 2:
How to prove program correctness?

Axioms

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

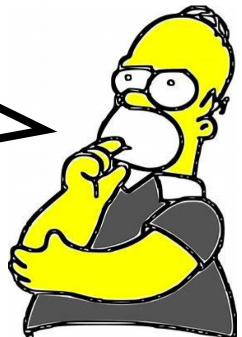
This axiom is clearly true.
Because skip does nothing.



Axioms

$$\{?\}a := e\{?\}$$

The assignment axiom brings troubles.



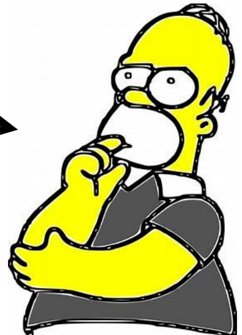
Axioms

Use a to replace e in P

$$\{P\}a := e\{P[a/e]\}$$

$$\{a = 0\}a := 1\{a = 0\}$$

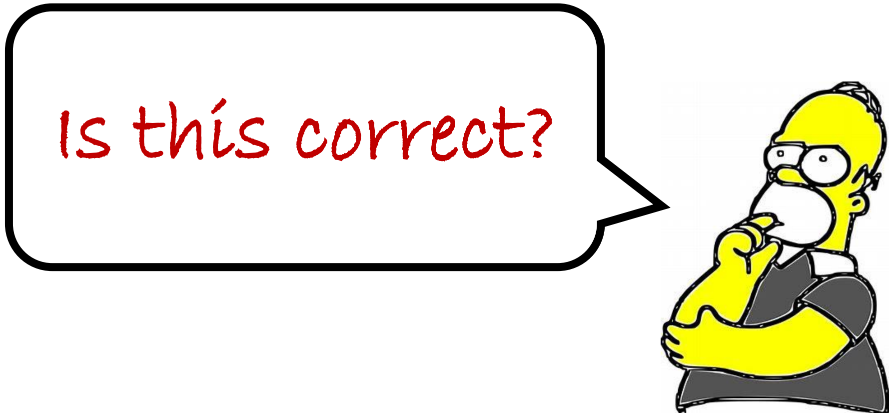
Is this correct?



Axioms

$$\{P\}a := e\{P \wedge a = e\}$$

$$\{a = 5\}a := a + 1\{a = 5 \wedge a = a + 1\}$$



Axioms

v is a fresh variable.

$$\{P\}a := e\{(\exists v)(a = \boxed{e[v/a]} \wedge \boxed{P[v/a]})\}$$

$$\{a = 1\}a := a + 1\{(\exists v)(a = \boxed{(a+1)[v/a]} \wedge \boxed{(a = 1)[v/a]})\}$$

$$\Downarrow$$

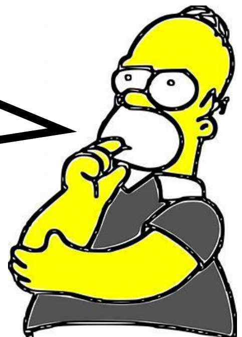
$$\{a = 1\}a := a + 1\{(\exists v)(a = \boxed{v+1} \wedge \boxed{v = 1})\}$$

simplification

$$\Downarrow$$

$$\{a = 1\}a := a + 1\{a = 2\}$$

This is correct and a forward assignment axiom.



Axioms

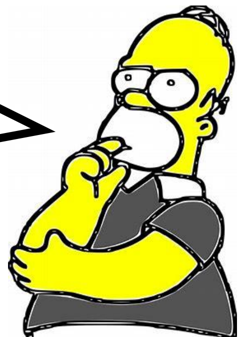
$$\{P[e/a]\}a := e\{P\}$$

$$\{b + 1 = 42\}a := b + 1\{a = 42\}$$

$$\{42 = 42\}a := 42\{a = 42\}$$

$$\{a - b > 3\}a := a - b\{a > 3\}$$

This is correct. But it seems to be "backward".



Axioms

The abbreviation is AS-FW.

The abbreviation is AS.

$$\{P\}a := e\{(\exists v)(a = e[v/a] \wedge P[v/a])\}$$

$$\{P[e/a]\}a := e\{P\}$$

- Forward
- Quantifier
- Intuitive

It is more widely used because it's quantifier-free.

- Backward
- Quantifier-free
- Not intuitive

Inference Rules

strong $P \to Q$ weak

The line means inference.

$$\frac{\{P\}C\{Q\} \quad Q \to R}{\{P\}C\{R\}}$$

weakening consequent(WC)

$$\frac{\begin{array}{c}\{a = 1\}a := a + 1\{(\exists v)(a = (a+1)[v/a] \wedge (a = 1)[v/a])\} \\ (\exists v)(a = (a+1)[v/a] \wedge (a = 1)[v/a]) \to a = 2\end{array}}{\{a = 1\}a := a + 1\{a = 2\}}$$

Inference Rules

strong $P \to Q$ weak

$$\frac{P \to Q \quad \{Q\}C\{R\}}{\{P\}C\{R\}}$$

We can also strengthen precedent(SP).

Inference Rules

Program structure

1. Sequence : s1; s2

2. Branch : if(b1) { s1 } then { s2 }

3. Loop structure : while(b1) { s }

We will introduce inference rules for these structures.

This is what we want the most.

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1; C2\{Q\}}$$

The sequential composition rule(SC)

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1;C2\{Q\}}$$

The sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

$Proof.$

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1; C2\{Q\}}$$

The sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

$Proof.$

$(1)\{a - b \geq 4\}a := a - b\{a \geq 4\}$ AS

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1;C2\{Q\}}$$

The sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

$Proof.$

$(1)\{a - b \geq 4\}a := a - b\{a \geq 4\}$ $\quad AS$

$(2)\{2 * b - b \geq 4\}a := 2 * b\{a - b \geq 4\}$ $\quad AS$

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1; C2\{Q\}}$$

The sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

$Proof.$

$(1)\{a - b \geq 4\}a := a - b\{a \geq 4\}$ $\qquad AS$

$(2)\{2 * b - b \geq 4\}a := 2 * b\{a - b \geq 4\}$ $\qquad AS$

$(3) b > 3 \rightarrow 2 * b - b \geq 4$ $\qquad predicate\ logic$

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1;C2\{Q\}}$$

The sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

$Proof.$

$(1)\{a - b \geq 4\}a := a - b\{a \geq 4\}$ AS

$(2)\{2 * b - b \geq 4\}a := 2 * b\{a - b \geq 4\}$ AS

$(3) b > 3 \rightarrow 2 * b - b \geq 4$ $predicate\ logic$

$(4)\{b > 3\}a := 2 * b\{a - b \geq 4\}$ $SP(2)(3)$

Inference Rules

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1;C2\{Q\}}$$

the sequential composition rule(SC)

$\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$

Proof.

$(1)\{a - b \geq 4\}a := a - b\{a \geq 4\}$ AS

$(2)\{2 * b - b \geq 4\}a := 2 * b\{a - b \geq 4\}$ AS

$(3)b > 3 \rightarrow 2 * b - b \geq 4$ *predicate logic*

$(4)\{b > 3\}a := 2 * b\{a - b \geq 4\}$ $SP(2)(3)$

$(5)\{b > 3\}a := 2 * b; a := a - b\{a \geq 4\}$ $SC(1)(4)$

It's clearly proved backwards.

Inference Rules

Convert b to a proposition.

Convert b to a proposition.

$$\frac{\{P \wedge istrue(b1)\}C1\{Q\} \quad \{P \wedge isfalse(b1)\}C2\{Q\}}{\{P\}if(b1)\ then\ C1\ else\ C2\ \{Q\}}$$

The conditional rule (CD)

Inference Rules

$$\{T\}\ if(a < b)\ then\ c := b\ else\ c := a\{c = max(a,b)\}$$

$Proof.$

Inference Rules

$$\{T\}\ if(a < b)\ then\ c := b\ else\ c := a\{c = max(a, b)\}$$

$Proof.$

$$(1)\{b = max(a, b)\}c := b\{c = max(a, b)\} \qquad AS$$

Inference Rules

$$\{T\}\ if(a < b)\ then\ c := b\ else\ c := a\{c = max(a, b)\}$$

$Proof.$

$(1)\{b = max(a, b)\}c := b\{c = max(a, b)\}$ $\qquad AS$

$(2)\{a = max(a, b)\}c := a\{c = max(a, b)\}$ $\qquad AS$

Inference Rules

$\{T\}\ if(a < b)\ then\ c := b\ else\ c := a\{c = max(a, b)\}$

$Proof.$

$(1)\{b = max(a, b)\}c := b\{c = max(a, b)\}$ $\quad AS$

$(2)\{a = max(a, b)\}c := a\{c = max(a, b)\}$ $\quad AS$

$(3)T \wedge a < b \rightarrow b = max(a, b)$ $\quad predicate\ logic$

Inference Rules

$$\{T\}\ if(a<b)\ then\ c:=b\ else\ c:=a\{c=max(a,b)\}$$

$Proof.$

$$(1)\{b=max(a,b)\}c:=b\{c=max(a,b)\} \qquad AS$$

$$(2)\{a=max(a,b)\}c:=a\{c=max(a,b)\} \qquad AS$$

$$(3)T\wedge a<b\rightarrow b=max(a,b) \qquad predicate\ logic$$

$$(4)T\wedge\neg(a<b)\rightarrow a=max(a,b) \qquad predicate\ logic$$

Inference Rules

$$\{T\}\ if(a < b)\ then\ c := b\ else\ c := a\{c = max(a, b)\}$$

$Proof.$

$$(1)\{b = max(a, b)\}c := b\{c = max(a, b)\} \qquad AS$$

$$(2)\{a = max(a, b)\}c := a\{c = max(a, b)\} \qquad AS$$

$$(3) T \wedge a < b \rightarrow b = max(a, b) \qquad predicate\ logic$$

$$(4) T \wedge \neg(a < b) \rightarrow a = max(a, b) \qquad predicate\ logic$$

$$(5)\{T \wedge a < b\}c := b\{c = max(a, b)\} \qquad SP(1)(3)$$

Inference Rules

$$\{T\}\ if(a<b)\ then\ c:=b\ else\ c:=a\{c=max(a,b)\}$$

$Proof.$

$(1)\{b=max(a,b)\}c:=b\{c=max(a,b)\}$	AS
$(2)\{a=max(a,b)\}c:=a\{c=max(a,b)\}$	AS
$(3)T\wedge a<b\rightarrow b=max(a,b)$	$predicate\ logic$
$(4)T\wedge\neg(a<b)\rightarrow a=max(a,b)$	$predicate\ logic$
$(5)\{T\wedge a<b\}c:=b\{c=max(a,b)\}$	$SP(1)(3)$
$(6)\{T\wedge\neg(a<b)\}c:=a\{c=max(a,b)\}$	$SP(2)(4)$

Inference Rules

$$\{T\}\ if(a<b)\ then\ c:=b\ else\ c:=a\{c=max(a,b)\}$$

$Proof.$

$(1)\{b=max(a,b)\}c:=b\{c=max(a,b)\}$	AS
$(2)\{a=max(a,b)\}c:=a\{c=max(a,b)\}$	AS
$(3)T\wedge a<b\rightarrow b=max(a,b)$	$predicate\ logic$
$(4)T\wedge\neg(a<b)\rightarrow a=max(a,b)$	$predicate\ logic$
$(5)\{T\wedge a<b\}c:=b\{c=max(a,b)\}$	$SP(1)(3)$
$(6)\{T\wedge\neg(a<b)\}c:=a\{c=max(a,b)\}$	$SP(2)(4)$
$(7)\{T\}\ if(a<b)\ then\ c:=b\ else\ c:=a\{c=max(a,b)\}$	$CD(5)(6)$

Inference Rules

loop invariant

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

The while rule (WHP)

It says that

- if executing C once preserves the truth of `I`, then executing C

  any number of times also preserves the truth of `I`

- after a while loop has terminated, the test must be false

Inference Rules

The while rule (WHP)

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

loop invariant

It can be proved by induction on number of loop times

- If the loop execute for 0 time, `I` of course holds.
- If `I` holds for n times loops, it holds for n+1 times because of the condition above the line.

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$$

$Proof.$

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$$

$Proof.$

$(1)\{a + 1 \leq 10\}a := a + 1\boxed{\{a \leq 10\}}$ $\quad AS$

loop invariant

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$$

$Proof.$

$(1)\{a + 1 \leq 10\}a := a + 1\{a \leq 10\}$ $\quad AS$

$(2)a \leq 10 \wedge a \neq 10 \rightarrow a + 1 \leq 10$ $\quad predicate\ logic$

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$

$Proof.$

$(1)\{a + 1 \leq 10\}a := a + 1\{a \leq 10\}$ $\qquad AS$

$(2)a \leq 10 \wedge a \neq 10 \rightarrow a + 1 \leq 10$ $\qquad predicate\ logic$

$(3)\{a \leq 10 \wedge a \neq 10\}a := a + 1\{a \leq 10\}$ $\qquad SP(1)(2)$

loop invariant always holds.

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$$

$Proof.$

$(1)\{a + 1 \leq 10\}a := a + 1\{a \leq 10\}$ $\qquad AS$

$(2)a \leq 10 \wedge a \neq 10 \rightarrow a + 1 \leq 10$ $\qquad predicate\ logic$

$(3)\{a \leq 10 \wedge a \neq 10\}a := a + 1\{a \leq 10\}$ $\qquad SP(1)(2)$

$(4)\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a \leq 10 \wedge a = 10\}$ $\qquad WHP(3)$

Inference Rules

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$$

$Proof.$

$(1)\{a + 1 \leq 10\}a := a + 1\{a \leq 10\}$ $\qquad AS$

$(2)a \leq 10 \wedge a \neq 10 \rightarrow a + 1 \leq 10$ $\qquad predicate\ logic$

$(3)\{a \leq 10 \wedge a \neq 10\}a := a + 1\{a \leq 10\}$ $\qquad SP(1)(2)$

$(4)\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a \leq 10 \wedge a = 10\}$ $\qquad WHP(3)$

$(5)a \leq 10 \wedge a = 10 \rightarrow a = 10$ $\qquad predicate\ logic$

$(6)\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$ $\qquad WC(4)(5)$

Exercise

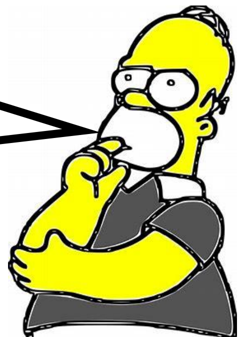
$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

```
void f(unsigned a)
{
    unsigned i = 0;
    {i = 0 ∧ a ≥ 0}
    while(i < a)
    {
        i = i + 1;
    }
    {i = a}
}
```

pre-condition: $\{i = 0 \wedge a \geq 0\}$

post-condition: $\{i = a\}$

What is the loop invariant?



Exercise

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

```
void f(unsigned a)
{
    unsigned i = 0;
```

$\{i = 0 \wedge a \geq 0\}$ pre-condition

```
    while(i < a)
    {
        i = i + 1;
    }
```

$\{i = a\}$ post-condition

```
}
```

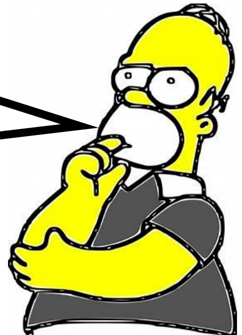
$$I : i \geq 0 \wedge i \leq a \wedge a \geq 0$$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

Try to prove it. The loop invariant is a >=2.



Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\} \qquad AS$$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$ $\qquad AS$

$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$ $\qquad predicate \quad logic$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\} \qquad AS$$

$$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2) \qquad predicate \quad logic$$

$$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\} \qquad SP(1)(2)$$

Example

$$\frac{\{I \land istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \land isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\} \qquad AS$$

$$(2)(a \geq 2 \land a > 2) \rightarrow (a - 1 \geq 2) \qquad predicate \quad logic$$

$$(3)\{a \geq 2 \land a > 2\}a := a - 1; \{a \geq 2\} \qquad SP(1)(2)$$

$$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \land a \leq 2\} \qquad WHP(3)$$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\} \qquad AS$$

$$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2) \qquad predicate \quad logic$$

$$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\} \qquad SP(1)(2)$$

$$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\} \qquad WHP(3)$$

$$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1) \qquad predicate \quad logic$$

$$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\} \qquad WC(4)(5)$$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$
$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\}$	$WC(4)(5)$
$(7)\{b \geq 2\}a := b\{a \geq 2\}$	AS

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$
$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\}$	$WC(4)(5)$
$(7)\{b \geq 2\}a := b\{a \geq 2\}$	AS
$(8)b \geq 20 \rightarrow b \geq 2$	$predicate \quad logic$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$
$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\}$	$WC(4)(5)$
$(7)\{b \geq 2\}a := b\{a \geq 2\}$	AS
$(8)b \geq 20 \rightarrow b \geq 2$	$predicate \quad logic$
$(9)\{b \geq 20\}a := b\{a \geq 2\}$	$SP(7)(8)$

Example

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$$

$(1)\{a - 1 \geq 2\}a := a - 1; \{a \geq 2\}$	AS
$(2)(a \geq 2 \wedge a > 2) \rightarrow (a - 1 \geq 2)$	$predicate \quad logic$
$(3)\{a \geq 2 \wedge a > 2\}a := a - 1; \{a \geq 2\}$	$SP(1)(2)$
$(4)\{a \geq 2\}while(a > 2)a := a - 1; \{a \geq 2 \wedge a \leq 2\}$	$WHP(3)$
$(5)(a \geq 2 \wedge a \leq 2) \rightarrow (a > 1)$	$predicate \quad logic$
$(6)\{a \geq 2\}while(a > 2)a := a - 1; \{a > 1\}$	$WC(4)(5)$
$(7)\{b \geq 2\}a := b\{a \geq 2\}$	AS
$(8)b \geq 20 \rightarrow b \geq 2$	$predicate \quad logic$
$(9)\{b \geq 20\}a := b\{a \geq 2\}$	$SP(7)(8)$
$(10)\{b \geq 20\}a := b; while(a > 2)a := a - 1; \{a > 1\}$	$SC(6)(9)$

Partial Correctness

$$\frac{\boxed{\{I \wedge istrue(b1)\}C\{I\}}}{\boxed{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}}$$

The loop invariant is "true"

$\{\textbf{true}\}$ **while** $x \neq 10$ **do skip** $\{x = 10\}$

Proof:

1. $\{\textbf{true} \wedge x \neq 10\}$ **skip** $\{\textbf{true} \wedge x \neq 10\}$ — SK
2. $\textbf{true} \wedge x \neq 10 \Rightarrow \textbf{true}$
3. $\boxed{\{\textbf{true} \wedge x \neq 10\}\ \textbf{skip}\ \{\textbf{true}\}}$ — WC, 1, 2
4. $\boxed{\{\textbf{true}\}\ \textbf{while}\ x \neq 10\ \textbf{do skip}\ \{\textbf{true} \wedge \neg(x \neq 10)\}}$ — WHP, 3
5. $\textbf{true} \wedge \neg(x \neq 10) \Rightarrow x = 10$
6. $\{\textbf{true}\}$ **while** $x \neq 10$ **do skip** $\{x = 10\}$ — WC, 4, 5

Partial Correctness

$$\frac{\boxed{\{I \wedge istrue(b1)\}C\{I\}}}{\boxed{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}}$$

The loop invariant is "true"

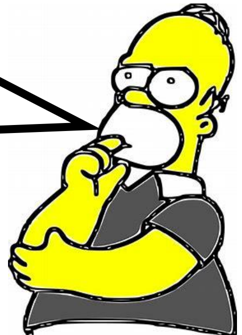
$\{\textbf{true}\}$ **while** $x \neq 10$ **do skip** $\{x = 10\}$

Proof:

1. $\{\textbf{true} \wedge x \neq 10\}$ **skip** $\{\textbf{true} \wedge x \neq 10\}$ SK
2. $\textbf{true} \wedge x \neq 10 \Rightarrow \textbf{true}$
3. $\boxed{\{\textbf{true} \wedge x \neq 10\}\ \textbf{skip}\ \{\textbf{true}\}}$ WC, 1, 2
4. $\boxed{\{\textbf{true}\}\ \textbf{while}\ x \neq 10\ \textbf{do skip}\ \{\textbf{true} \wedge \neg(x \neq 10)\}}$ WHP, 3

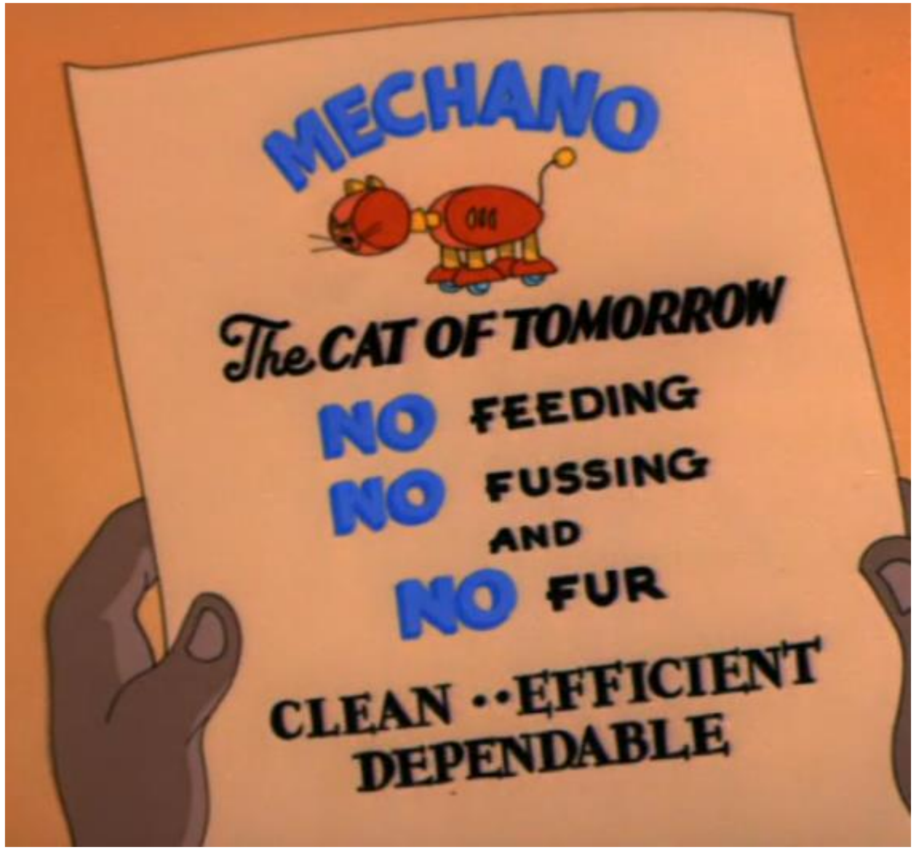
The proof does not ensure that the program can always terminate.

6. $\{\textbf{true}\}$ **while** $x \neq 10$ **do skip** $\{x = 10\}$



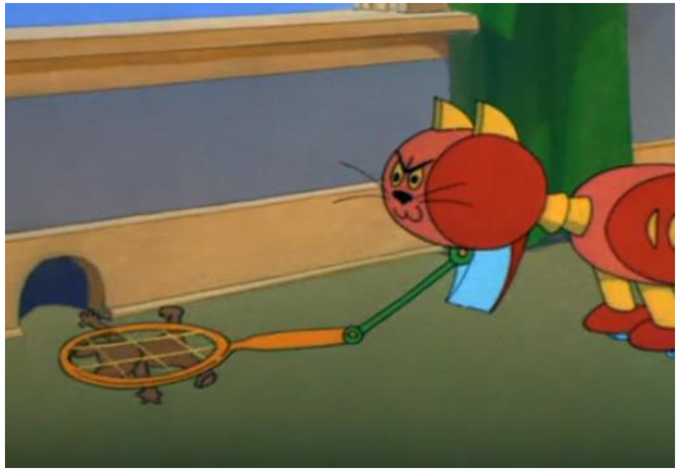
However,

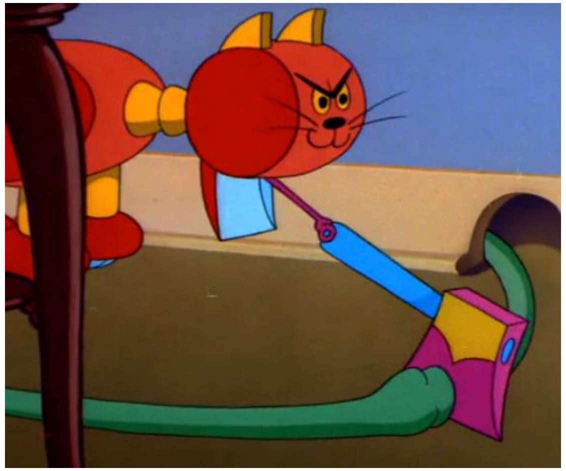
Manual Proof is usually tedious

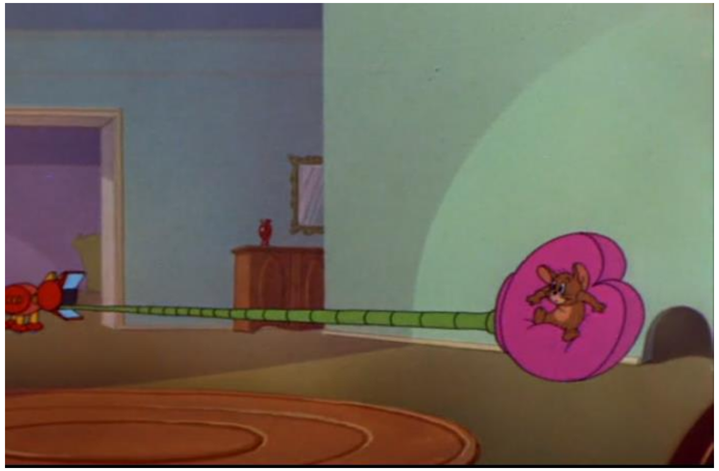


We want an automated program verifier.

But previous methods (symbolic execution) cannot handle loops







What if we keep all invariants/conditions in the proof?

Automatic Proof, Again???

Predicate Transformer

stronger $P \rightarrow Q$ weaker

DEFINITION

Given a program C and postcondition Q, P is the weakest precondition means for all P’, {P’} C {Q} iff P’ → P. We use wp(C, Q) to denote P.

weakest precondition

$$\{a = 2\} a := a + 1 \{a = 3\}$$

$$\{a = 2 \land b = 4\} a := a + 1 \{a = 3\} \qquad a = 2 \land b = 4 \rightarrow a = 2$$

$$\{a = 2 \land a > 1\} a := a + 1 \{a = 3\} \qquad a = 2 \land a > 1 \rightarrow a = 2$$

Predicate Transformer

stronger $P \to Q$ weaker

DEFINITION

Given a program C and precondition P, Q is the strongest postcondition means for all Q', {P} C {Q'} iff Q → Q'. We use sp(C, P) to denote Q.

$$\{a = 1\}a := 3\{a = 3\}$$ strongest postcondition

$$\{a = 1\}a := 3\{T\} \qquad a = 3 \to T$$

$$\{a = 1\}a := 3\{(\exists v)(a = v)\} \qquad a = 3 \to (\exists v)(a = v)$$

Predicate Transformer

We can consider the problem from the following perspective:

- Given a precondition and program, can we find the strongest postcondition?

- Or given a postcondition and program, can we find the weakest precondition?

- This is called predicate transformer semantics: how programs transform assertions

$$\{P\}C\{Q\} \Rightarrow P \rightarrow wp(C, Q)$$

$$\{P\}C\{Q\} \Rightarrow sp(C, P) \rightarrow Q$$

If we can find wp or sp, we can convert hoare triples to predicate logic formula.

Predicate Transformer

We can consider the problem from the following perspective:

- Given a precondition and program, can we find the strongest postcondition?

- Or given a postcondition and program, can we find the weakest precondition?

- This is called predicate transformer semantics: how programs transform assertions

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

$$P \rightarrow wp(C, Q)$$

We just need to prove it is true.

$$sp(C, P) \rightarrow Q$$

We just need to prove it is true.

Predicate Transformer

We can consider the problem from the following perspective:

- Given a precondition and program, can we find the strongest postcondition?

- Or given a postcondition and program, can we find the weakest precondition?

- This is called predicate transformer semantics: how programs transform assertions

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

$$P \land \neg wp(C, Q)$$

We just need to prove it is unsatisfiable.

$$sp(C, P) \land \neg Q$$

We just need to prove it is unsatisfiable.

Basic Idea

To prove `{P}C{Q}`

- Find the `wp(C, Q)`, then prove `P` implies `wp(C, Q)`.
- Find the `sp(C, P)`, then prove `sp(C, P)` implies `Q`.

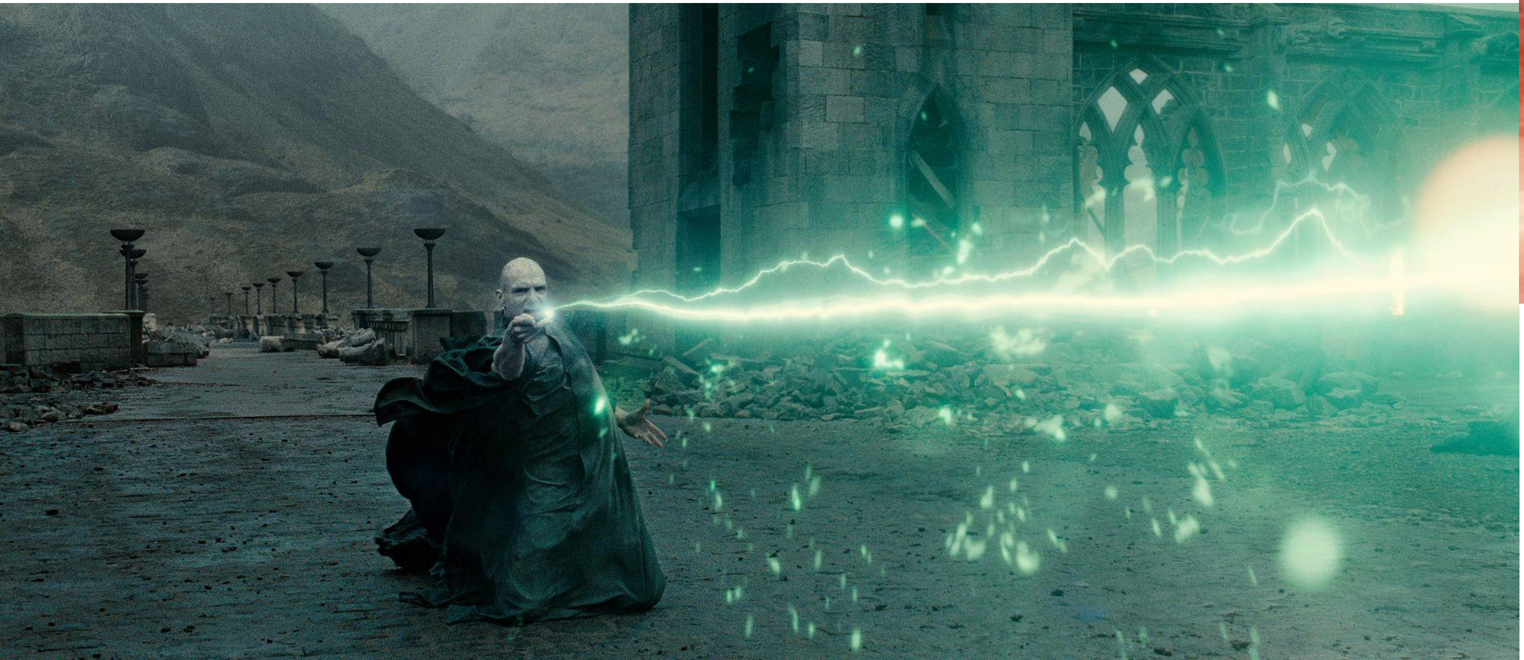
$$\{P\}C\{Q\}$$

$$P \land \neg wp(C, Q)$$

$$sp(C, P) \land \neg Q$$

How to prove the formula?

Working with SMT solvers is like magic



Question: How to automatically find the weakest precondition (strongest postcondition)?

By rules (Theorems)…

A Quick Recap

AS-FW gives the strongest postcondition.

$$\{P\}a := e\{(\exists v)(a = e[v/a] \wedge P[v/a])\}$$

AS gives the weakest precondition.

$$\{P[e/a]\}a := e\{P\}$$

- Forward
- Quantifier
- Intuitive

It is more widely used because it's simpler.

- Backward
- Quantifier-free
- Not intuitive

A Quick Recap

AS-FW gives the strongest postcondition.

$$\{P\}a := e\{(\exists v)(a = e[v/a] \wedge P[v/a])\}$$

AS gives the weakest postcondition.

$$\{P[e/a]\}a := e\{P\}$$

- Forward
- Quantifier
- Intuitive

It is more widely used because it's simpler.

- Backward
- Quantifier-free

We will focus on wp.

tive

Weakest Precondition

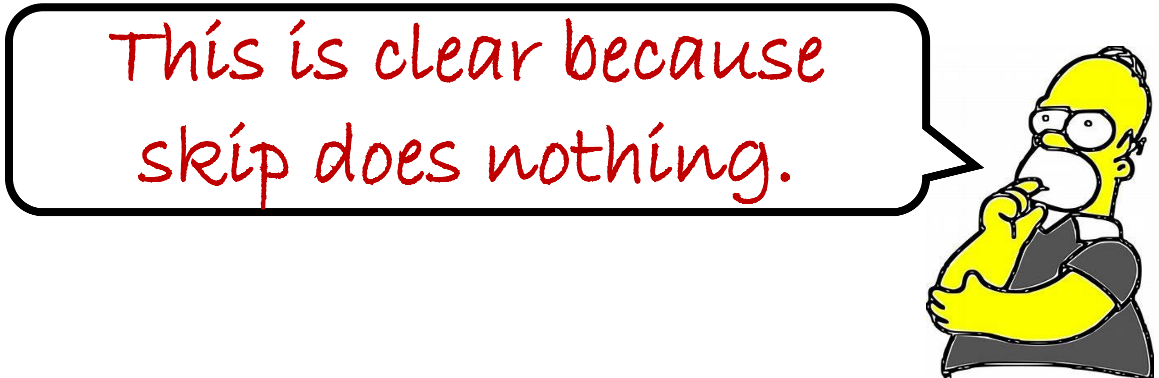
We use wlp(C, Q) instead of wp(C, Q) in following slides:

- C is the program, Q is the intended postcondition
- Name stands for weakest “liberal” precondition: it ignores termination
- For all C and Q, it is guaranteed that { wlp(C, Q) } C {Q} is true
- For all P, C and Q, {P} C {Q} iff $P \rightarrow wlp(C, Q)$

Wlp must be computable. We will introduce the details.

Weakest Precondition

$$wlp(skip, Q) = Q$$



Weakest Precondition

$$\{P[e/a]\}a := e\{P\}$$

This is similar to AS in Hoare logic.

$$wlp(a := e, P) = P[e/a]$$

$$wlp(a := b + 1, a = 42) = b + 1 = 42$$

$$wlp(a := a - b, a > 3) =$$?

Weakest Precondition

$$\{P[e/a]\}a := e\{P\}$$

This is similar to AS in Hoare logic.

$$wlp(a := e, P) = P[e/a]$$

$$wlp(a := b + 1, a = 42) = b + 1 = 42$$

$$wlp(a := a - b, a > 3) = a - b > 3$$

Weakest Precondition

$$wlp(C1; C2, Q) = wlp(C1, wlp(C2, Q))$$

$$wlp(a := a + 1; b := a, a = 3 \wedge b > 0)$$

Weakest Precondition

$$wlp(C1; C2, Q) = wlp(C1, wlp(C2, Q))$$

$$wlp(a := a + 1; b := a, a = 3 \wedge b > 0)$$

$$= wlp(a := a + 1, wlp(b := a, a = 3 \wedge b > 0))$$

$$= wlp(a := a + 1, a = 3 \wedge a > 0)$$

$$= \boxed{a + 1 = 3 \wedge a + 1 > 0}$$

Weakest Precondition

$$wlp(if(b1)\ then\ \{C1\}\ else\ \{C2\}, Q)$$

$$=(istrue(b1) \rightarrow wlp(C1, Q)) \wedge (isfalse(b1) \rightarrow wlp(C2, Q))$$

$$wlp(if(a == 0)\ then\ \{b := 1\}\ else\ \{b := 0\}, b = 1 \vee b = 0)$$

Weakest Precondition

$$wlp(if(b1)\ then\ \{C1\}\ else\ \{C2\}, Q)$$

$$=(istrue(b1) \rightarrow wlp(C1, Q)) \wedge (isfalse(b1) \rightarrow wlp(C2, Q))$$

$$wlp(if(a == 0)\ then\ \{b := 1\}\ else\ \{b := 0\}, b = 1 \vee b = 0)$$

$$=(a = 0 \rightarrow wlp(b := 1, b = 1 \vee b = 0)) \wedge (a \neq 0 \rightarrow wlp(b := 0, b = 1 \vee b = 0))$$

$$=(a = 0 \rightarrow 1 = 1 \vee 1 = 0) \wedge (a \neq 0 \rightarrow 0 = 1 \vee 0 = 0)$$

Exercise

$$a := b + 2;\ \ if(a > 2)\ then\ b := c + 1;\ \ else\ b := c - 1;$$

Assume the post-condition is $b > 5 \wedge a < 10$

Exercise

$$a := b + 2;\ \ if(a > 2)\ then\ b := c + 1;\ \ else\ b := c - 1;$$

$$\begin{aligned}
&wlp(\quad a := b + 2;\ \ if(a > 2)\ then\ b := c + 1;\ \ else\ b := c - 1;,\quad b > 5 \wedge a < 10) \\
=&wlp(a := b + 2, ((a > 2) \rightarrow wlp(b := c + 1, b > 5 \wedge a < 10)) \wedge (\neg(a > 2) \rightarrow wlp(b := c - 1, b > 5 \wedge a < 10))) \\
=&wlp(a := b + 2, ((a > 2) \rightarrow (c + 1 > 5 \wedge a < 10)) \wedge (\neg(a > 2) \rightarrow (c - 1 > 5 \wedge a < 10))) \\
=&((b + 2 > 2) \rightarrow (c + 1 > 5 \wedge b + 2 < 10)) \wedge (\neg(b + 2 > 2) \rightarrow (c - 1 > 5 \wedge b + 2 < 10))
\end{aligned}$$

Weakest Precondition

$$wlp(while(b1)\{C\}, Q) =?$$

:-) 呵呵

Weakest Precondition

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{P\}while(b1)C\{Q\} \quad \begin{cases} P \rightarrow I \\ (I \wedge isfalse(b1)) \rightarrow Q \\ \{I \wedge istrue(b1)\}C\{I\} \end{cases}$$

The loop invariant needs to be written manually

This can be further proved via the weakest precondition

A Quick Recap

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first order logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants

PySAT

Z3

Dafny

Soundness and Completeness for Verifiers

DEFINITION

A verification algorithm is **sound** means that when a program passes the verification, it is a “correct” program.

Note that we can define the correctness in multiple different ways.

All verification algorithms introduced before are sound

Soundness and Completeness for Verifiers

DEFINITION

A verification algorithm is **complete** means that every correct program can pass the verification.

Previous algorithms are not complete because of two main reasons:

- They may not automatically prove programs with unbounded loops.
- The satisfiability problem of FOL formulas is undecidable.

Thanks & Questions