Hoare Logic

15-413: Introduction to Software Engineering

Jonathan Aldrich

Some presentation ideas from a lecture by
K. Rustan M. Leino

How would you argue that this program is correct?

```
float sum(float *array, int length) {
    float sum = 0.0;
    int i = 0;
    while (i < length) {
        sum = sum + array[i];
        i = i + 1;
    }
    return sum;
}
```

Function Specifications

- Predicate: a boolean function over program state
  - $x=3$
  - $y > x$
  - $(x \neq 0) \Rightarrow (y+z = w)$
  - $s = \Sigma_{(i \in 1..n)} a[i]$
  - $\forall i \in 1..n \;.\; a[i] > a[i-1]$
  - true

Function Specifications

- Contract between client and implementation
  - Precondition:
    - A predicate describing the condition the function relies on for correct operation
  - Postcondition:
    - A predicate describing the condition the function establishes after correctly running
- Correctness with respect to the specification
  - If the client of a function fulfills the function's precondition, the function will execute to completion and when it terminates, the postcondition will be true
- What does the implementation have to fulfill if the client violates the precondition?

Function Specifications

```
/*@ requires len >= 0 && array.length = len
  @
  @ ensures \result ==
  @             (\sum int j;  0 <= j && j < len;  array[j])
  @*/
float sum(int array[], int len) {
      float sum = 0.0;
      int i = 0;
      while (i < length) {
            sum = sum + array[i];
            i = i + 1;
      }
      return sum;
}
```


Hoare Triples

- Formal reasoning about program correctness using pre- and postconditions
- Syntax: {P} S {Q}
  - P and Q are predicates
  - S is a program
- If we start in a state where P is true and execute S, S will terminate in a state where Q is true

Hoare Triple Examples

- { true } x := 5 { x=5 }
- { x = y } x := x + 3 { x = y + 3 }
- { x > 0 } x := x * 2 { x > -2 }
- { x=a } if (x < 0) then x := -x { x=|a| }
- { false } x := 3 { x = 8 }

Strongest Postconditions

- Here are a number of valid Hoare Triples:
  - {x = 5} x := x * 2 { true }
  - {x = 5} x := x * 2 { x > 0 }
  - {x = 5} x := x * 2 { x = 10 || x = 5 }
  - {x = 5} x := x * 2 { x = 10 }
    - All are true, but this one is the most *useful*
    - x=10 is the *strongest postcondition*
- If {P} S {Q} and for all Q' such that {P} S {Q'}, Q ⇒ Q', then Q is the strongest postcondition of S with respect to P
  - check: x = 10 ⇒ true
  - check: x = 10 ⇒ x > 0
  - check: x = 10 ⇒ x = 10 || x = 5
  - check: x = 10 ⇒ x = 10

Weakest Preconditions

- Here are a number of valid Hoare Triples:
  - {x = 5 && y = 10} z := x / y { z < 1 }
  - {x < y && y > 0} z := x / y { z < 1 }
  - {y ≠ 0 && x / y < 1} z := x / y { z < 1 }
    - All are true, but this one is the most *useful* because it allows us to invoke the program in the most general condition
    - y ≠ 0 && x / y < 1 is the *weakest precondition*
- If {P} S {Q} and for all P' such that {P'} S {Q}, P' ⇒ P, then P is the weakest precondition *wp*(S,Q) of S with respect to Q

Hoare Triples and Weakest Preconditions

- {P} S {Q} holds if and only if P ⇒ *wp*(S,Q)
  - In other words, a Hoare Triple is still valid if the precondition is stronger than necessary, but not if it is too weak
- Question: Could we state a similar theorem for a strongest postcondition function?
  - e.g. {P} S {Q} holds if and only if *sp*(S,P) ⇒ Q

Hoare Logic Rules

- Assignment
  - { P } x := 3 { x+y > 0 }
  - What is the weakest precondition P?
    - Student answer: y > -3
    - How to get it:
      - what is most general value of y such that 3 + y > 0

Hoare Logic Rules

- Assignment
  - { P } x := 3*y + z { x * y - z > 0 }
  - What is the weakest precondition P?

Hoare Logic Rules

- Assignment
  - { P } x := 3 { x+y > 0 }
  - What is the weakest precondition P?
- Assignment rule
  - *wp*(x := E, P) = [E/x] P
  - { [E/x] P } x := E { P }
  - [3 / x] (x + y > 0)
  - = (3) + y > 0
  - = y > -3

16 November 2005

Hoare Logic Rules

- Assignment
  - { P } x := 3*y + z { x * y - z > 0 }
  - What is the weakest precondition P?
- Assignment rule
  - *wp*(x := E, P) = [E/x] P
  - { [E/x] P } x := E { P }
  - [3*y+z / x] (x * y – z > 0)
  - = (3*y+z) * y - z > 0
  - = $3*y^2$ + z*y - z > 0

16 November 2005

Hoare Logic Rules

- Sequence
  - { P } x := x + 1; y := x + y { y > 5 }
  - What is the weakest precondition P?

Hoare Logic Rules

- Sequence
  - { P } x := x + 1; y := x + y { y > 5 }
  - What is the weakest precondition P?
- Sequence rule
  - *wp*(S;T, Q) = *wp*(S, *wp*(T, Q))
  - *wp*(x:=x+1; y:=x+y, y>5)
  - = *wp*(x:=x+1, *wp*(y:=x+y, y>5))
  - = *wp*(x:=x+1, x+y>5)
  - = x+1+y>5
  - = x+y>4

Hoare Logic Rules

- Conditional
  - { P } if x > 0 then y := x else y := -x { y > 5 }
  - What is the weakest precondition P?
    - Student answer:
    - case then: {P1} y :=x { y > 5}
    - P1 = x>5
    - case else: {P1} y :=-x { y > 5}
    - P2 = -x > 5
    - P2 = x < -5
    - P = x > 5 || x < -5

Hoare Logic Rules

- Conditional
  - { P } if x > 0 then y := x else y := -x { y > 5 }
  - What is the weakest precondition P?
- Conditional rule
  - *wp*(if B then S else T, Q)
    = B ⇒ *wp*(S,Q) && ¬B ⇒ *wp*(T,Q)
  - *wp*(if x>0 then y:=x else y:=-x, y>5)
  - = x>0 ⇒ *wp*(y:=x,y>5) && x≤0 ⇒ *wp*(y:=-x,y>5)
  - = x>0 ⇒ *x*>5 && x≤0 ⇒ -*x*>5
  - = x>0 ⇒ *x*>5 && x≤0 ⇒ *x* < -5
  - = *x* > 5 || *x* < -5

Hoare Logic Rules

- Loops
  - { P } while (i < x) f=f*i; i := i + 1 { f = x! }
  - What is the weakest precondition P?

Proving loops correct

- First consider *partial correctness*
  - The loop may not terminate, but if it does, the postcondition will hold
- {P} while B do S {Q}
  - Find an invariant Inv such that:
    - $P \Rightarrow Inv$
      - The invariant is initially true
    - { Inv && B } S {Inv}
      - Each execution of the loop preserves the invariant
    - $(Inv \&\& \neg B) \Rightarrow Q$
      - The invariant and the loop exit condition imply the postcondition

Loop Example

- Prove array sum correct

```
{ N ≥ 0 }
j := 0;
s := 0;

while (j < N) do
    s := s + a[j];
    j := j + 1;

end
{ s = (Σi | 0≤i<N • a[i]) }
```

16 November 2005

Loop Example

- Prove array sum correct

```
{ N ≥ 0 }
j := 0;
s := 0;
{ Inv }
while (j < N) do { Inv && j < N}
    s := s + a[j];
    j := j + 1;
    { Inv }
end
{ s = (Σi | 0≤i<N • a[i]) }
```

16 November 2005

Guessing Loop Invariants

- Usually has same form as postcondition
  - s = (Σi | 0≤i<N • a[i])
- But depends on loop index j in some way
  - We know that j is initially 0 and is incremented until it reaches N
  - Thus 0 ≤ j ≤ N is probably part of the invariant
- Loop exit && invariant ⇒ postcondition
  - Loop exits when j = N
  - Good guess: replace N with j in postcondition
  - s = (Σi | 0≤i<j • a[i])
- Overall: 0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i])

Loop Example

- Prove array sum correct

```
{ N ≥ 0 }
j := 0;
s := 0;
{ 0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }
while (j < N) do
    {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) && j < N}
    s := s + a[j];
    j := j + 1;
    {0 ≤ j ≤ N && s = (Σi | 0≤i<j • a[i]) }
end
{ s = (Σi | 0≤i<N • a[i]) }
```