

Hoare Logic: Proving Programs Correct

17-654/17-765

Analysis of Software Artifacts

Jonathan Aldrich

Reading: C.A.R. Hoare, An Axiomatic Basis for
Computer Programming

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



Testing and Proofs



- Testing
 - Observable properties
 - Verify program for one execution
 - Manual development with automated regression
 - Most practical approach now
- Proofs
 - Any program property
 - Verify program for all executions
 - Manual development with automated proof checkers
 - May be practical for small programs in 10-20 years
- So why learn about proofs if they aren't practical?
 - Proofs tell us how to **think** about program correctness
 - Important for development, inspection
 - Foundation for static analysis tools
 - These are just simple, automated theorem provers
 - Many are practical today!

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
 - i.e. an expression that returns a boolean
 - We often use mathematical symbols as well as program text
- Examples
 - $x=3$
 - $y > x$
 - $(x \neq 0) \Rightarrow (y+z = w)$
 - $s = \sum_{(i \in 1..n)} a[i]$
 - $\forall i \in 1..n . a[i] > a[i-1]$
 - 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?
 - A: Nothing. It can do anything at all.

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, then S will terminate in a state where Q is true

Hoare Triple Examples



- $\{ \text{true} \} x := 5 \{ x=5 \}$
- $\{ x = y \} x := x + 3 \{ x = y + 3 \}$
- $\{ x > -1 \} x := x * 2 + 3 \{ x > 1 \}$
- $\{ x=a \} \text{if } (x < 0) \text{ then } x := -x \{ x=|a| \}$
- $\{ \text{false} \} x := 3 \{ x = 8 \}$
- $\{ x < 0 \} \text{while } (x \neq 0) x := x-1 \{ \quad \}$
 - no such triple!

Strongest Postconditions



- Here are a number of valid Hoare Triples:
 - $\{x = 5\} x := x * 2 \{ \text{true} \}$
 - $\{x = 5\} x := x * 2 \{ x > 0 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \parallel x = 5 \}$
 - $\{x = 5\} x := x * 2 \{ x = 10 \}$
 - 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 $\{P\} S \{Q'\}$, $Q \Rightarrow Q'$, then Q is the strongest postcondition of S with respect to P
 - check: $x = 10 \Rightarrow \text{true}$
 - check: $x = 10 \Rightarrow x > 0$
 - check: $x = 10 \Rightarrow x = 10 \parallel x = 5$
 - check: $x = 10 \Rightarrow x = 10$

Weakest Preconditions



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

Hoare Logic Rules



- Assignment
 - $\{P\} x := 3 \{x+y > 0\}$
 - What is the weakest precondition P?
 - What is most general value of y such that $3 + y > 0$?
 - $y > -3$

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
 - $wp(x := E, P) = [E/x] P$
 - Resulting triple: $\{ [E/x] P \} x := E \{ P \}$
 - $[3 / x] (x + y > 0)$
 - $= (3) + y > 0$
 - $= y > -3$

Hoare Logic Rules



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

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
 - $\{ P \} \text{if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
 - What is the weakest precondition P?

Hoare Logic Rules



- Conditional
 - $\{ P \} \text{ if } x > 0 \text{ then } y := z \text{ else } y := -z \{ y > 5 \}$
 - What is the weakest precondition P?
- Conditional rule
 - $wp(\text{if } B \text{ then } S \text{ else } T, Q)$
 $= B \Rightarrow wp(S, Q) \ \&\& \ \neg B \Rightarrow wp(T, Q)$
 - $wp(\text{if } x > 0 \text{ then } y := z \text{ else } y := -z, y > 5)$
 - $= x > 0 \Rightarrow wp(y := z, y > 5) \ \&\& \ x \leq 0 \Rightarrow wp(y := -z, y > 5)$
 - $= x > 0 \Rightarrow z > 5 \ \&\& \ x \leq 0 \Rightarrow -z > 5$
 - $= x > 0 \Rightarrow z > 5 \ \&\& \ x \leq 0 \Rightarrow z < -5$

Hoare Logic Rules



- Loops
 - $\{ P \} \text{ while } (i < x) \text{ f=f*i; } i := i + 1 \{ f = x! \}$
 - 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:
 - $P \Rightarrow \text{Inv}$
 - The invariant is initially true
 - $\{\text{Inv} \ \&\& \ B\} S \ \{\text{Inv}\}$
 - Each execution of the loop preserves the invariant
 - $(\text{Inv} \ \&\& \ \neg B) \Rightarrow Q$
 - The invariant and the loop exit condition imply the postcondition
 - **Why do we need each condition?**

Loop Example



- Prove array sum correct

$\{ N \geq 0 \}$

$j := 0;$

$s := 0;$

while ($j < N$) do

$j := j + 1;$

$s := s + a[j];$

end

$\{ s = (\sum_i \mid 0 \leq i < N \bullet a[i]) \}$

Loop Example



- Prove array sum correct

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

Proof Obligations



- Invariant is initially true
- ```
{ N ≥ 0 }
j := 0;
s := 0;
{ 0 ≤ j ≤ N && s = (∑i | 0 ≤ i < j • a[i]) }
```
- Invariant is maintained
- ```
{ 0 ≤ j ≤ N && s = (∑i | 0 ≤ i < j • a[i]) && j < N }
j := j + 1;
s := s + a[j];
{ 0 ≤ j ≤ N && s = (∑i | 0 ≤ i < j • a[i]) }
```
- Invariant and exit condition implies postcondition
- ```
0 ≤ j ≤ N && s = (∑i | 0 ≤ i < j • a[i]) && j ≥ N
⇒ s = (∑i | 0 ≤ i < N • a[i])
```

## Proof Obligations



- Invariant is initially true
 
$$\{ N \geq 0 \}$$

$$\{ 0 \leq 0 \leq N \ \&\& \ 0 = (\sum_i | 0 \leq i < 0 \cdot a[i]) \} \ // \text{ by assignment rule}$$

$$j := 0;$$

$$\{ 0 \leq j \leq N \ \&\& \ 0 = (\sum_i | 0 \leq i < j \cdot a[i]) \} \ // \text{ by assignment rule}$$

$$s := 0;$$

$$\{ 0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \}$$
- Need to show that:
 
$$(N \geq 0) \Rightarrow (0 \leq 0 \leq N \ \&\& \ 0 = (\sum_i | 0 \leq i < 0 \cdot a[i]))$$

$$= (N \geq 0) \Rightarrow (0 \leq N \ \&\& \ 0 = 0) \ // \ 0 \leq 0 \text{ is true, empty sum is } 0$$

$$= (N \geq 0) \Rightarrow (0 \leq N) \ // \ 0=0 \text{ is true, } P \ \&\& \ \text{true is } P$$

$$= \text{true}$$

## Proof Obligations



- Invariant is maintained
 
$$\{ 0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \ \&\& \ j < N \}$$

$$\{ 0 \leq j+1 \leq N \ \&\& \ s+a[j+1] = (\sum_i | 0 \leq i < j+1 \cdot a[i]) \} \ // \text{ by assignment rule}$$

$$j := j + 1;$$

$$\{ 0 \leq j \leq N \ \&\& \ s+a[j] = (\sum_i | 0 \leq i < j+1 \cdot a[i]) \} \ // \text{ by assignment rule}$$

$$s := s + a[j];$$

$$\{ 0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \}$$
- Need to show that:
 
$$(0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \ \&\& \ j < N)$$

$$\Rightarrow (0 \leq j+1 \leq N \ \&\& \ s+a[j+1] = (\sum_i | 0 \leq i < j+1 \cdot a[i]))$$

$$= (0 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$

$$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j+1] = (\sum_i | 0 \leq i < j+1 \cdot a[i])) \ // \text{ simplify bounds of } j$$

$$= (0 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$

$$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j+1] = (\sum_i | 0 \leq i < j \cdot a[i] + a[j])) \ // \text{ separate last element}$$

**// we have a problem – we need  $a[j+1]$  and  $a[j]$  to cancel out**

## Where's the error?



- Prove array sum correct

```
{ N ≥ 0 }
```

```
j := 0;
```

```
s := 0;
```

```
while (j < N) do
```

```
 j := j + 1;
```

```
 s := s + a[j];
```

Need to add element  
*before* incrementing j

```
end
```

```
{ s = (Σi | 0 ≤ i < N • a[i]) }
```

## Corrected Code



- 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]) }
```

## Proof Obligations



- Invariant is maintained  
 $\{0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \ \&\& \ j < N\}$   
 $\{0 \leq j+1 \leq N \ \&\& \ s+a[j] = (\sum_i | 0 \leq i < j+1 \cdot a[i]) \}$  // by assignment rule  
 $s := s + a[j];$   
 $\{0 \leq j+1 \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j+1 \cdot a[i]) \}$  // by assignment rule  
 $j := j + 1;$   
 $\{0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \}$
  - Need to show that:  
 $(0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \ \&\& \ j < N)$   
 $\Rightarrow (0 \leq j+1 \leq N \ \&\& \ s+a[j] = (\sum_i | 0 \leq i < j+1 \cdot a[i]))$
- $$= (0 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$
- $$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum_i | 0 \leq i < j+1 \cdot a[i]))$$
- // simplify bounds of j
- $$= (0 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$
- $$\Rightarrow (-1 \leq j < N \ \&\& \ s+a[j] = (\sum_i | 0 \leq i < j \cdot a[i]) + a[j])$$
- // separate last part of sum
- $$= (0 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$
- $$\Rightarrow (-1 \leq j < N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]))$$
- // subtract a[j] from both sides
- $$= \text{true}$$
- //
- $0 \leq j \Rightarrow -1 \leq j$

## Proof Obligations



- Invariant and exit condition implies postcondition  
 $0 \leq j \leq N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i]) \ \&\& \ j \geq N$   
 $\Rightarrow s = (\sum_i | 0 \leq i < N \cdot a[i])$
- $$= 0 \leq j \ \&\& \ j = N \ \&\& \ s = (\sum_i | 0 \leq i < j \cdot a[i])$$
- $$\Rightarrow s = (\sum_i | 0 \leq i < N \cdot a[i])$$
- $$\text{// because } (j \leq N \ \&\& \ j \geq N) = (j = N)$$
- $$= 0 \leq N \ \&\& \ s = (\sum_i | 0 \leq i < N \cdot a[i]) \Rightarrow s = (\sum_i | 0 \leq i < N \cdot a[i])$$
- $$\text{// by substituting } N \text{ for } j, \text{ since } j = N$$
- $$= \text{true} \quad \text{// because } P \ \&\& \ Q \Rightarrow Q$$

## Invariant Intuition



- For code without loops, we are simulating execution directly
  - We prove one Hoare Triple for each statement, and each statement is executed once
- For code with loops, we are doing *one* proof of correctness for *multiple* loop iterations
  - Don't know how many iterations there will be
  - Need our proof to cover all of them
  - The invariant expresses a *general* condition that is true for every execution, but is still strong enough to give us the postcondition we need
  - This tension between generality and precision can make coming up with loop invariants hard

## Total Correctness for Loops



- $\{P\}$  while B do S  $\{Q\}$
- Partial correctness:
  - Find an invariant Inv such that:
    - $P \Rightarrow \text{Inv}$ 
      - The invariant is initially true
    - $\{\text{Inv} \ \&\& \ B\} S \ \{\text{Inv}\}$ 
      - Each execution of the loop preserves the invariant
    - $(\text{Inv} \ \&\& \ \neg B) \Rightarrow Q$ 
      - The invariant and the loop exit condition imply the postcondition
- Total correctness
  - Loop will terminate

## Termination



- How would you prove this program terminates?

```
{ N ≥ 0 }
```

```
j := 0;
```

```
s := 0;
```

```
while (j < N) do
```

```
 s := s + a[j];
```

```
 j := j + 1;
```

```
end
```

```
{ s = (∑i | 0 ≤ i < N • a[i]) }
```

## Total Correctness for Loops



- {P} while B do S {Q}
- Partial correctness:
  - Find an invariant *Inv* such that:
    - $P \Rightarrow \text{Inv}$ 
      - The invariant is initially true
    - $\{ \text{Inv} \ \&\& \ B \} S \{ \text{Inv} \}$ 
      - Each execution of the loop preserves the invariant
    - $(\text{Inv} \ \&\& \ \neg B) \Rightarrow Q$ 
      - The invariant and the loop exit condition imply the postcondition
- Termination bound
  - Find a *variant function* *v* such that:
    - *v* is an upper bound on the number of loops remaining
    - $(\text{Inv} \ \&\& \ B) \Rightarrow v > 0$ 
      - The variant function evaluates to a finite integer value greater than zero at the beginning of the loop
    - $\{ \text{Inv} \ \&\& \ B \ \&\& \ v = V \} S \{ v < V \}$ 
      - The value of the variant function decreases each time the loop body executes (here *V* is a constant)

## Total Correctness Example



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

- Variant function for this loop?
  - N-j

## Guessing Variant Functions



- Loops with an index
  - $N \pm i$
  - Applies if you always add or always subtract a constant, and if you exit the loop when the index reaches some constant
  - Use N-i if you are incrementing i, N+i if you are decrementing i
  - Set N such that  $N \pm i \leq 0$  at loop exit
- Other loops
  - Find an expression that is an upper bound on the number of iterations left in the loop

## Additional Proof Obligations



- Variant function for this loop:  $N-j$
- To show: variant function initially positive  
 $(0 \leq j \leq N \ \&\& \ s = (\sum_i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j < N)$   
 $\Rightarrow N-j > 0$
- To show: variant function is decreasing  
 $\{0 \leq j \leq N \ \&\& \ s = (\sum_i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j < N \ \&\& \ N-j = V\}$   
 $s := s + a[j];$   
 $j := j + 1;$   
 $\{N-j < V\}$

## Additional Proof Obligations



- To show: variant function initially positive  
 $(0 \leq j \leq N \ \&\& \ s = (\sum_i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j < N)$   
 $\Rightarrow N-j > 0$   
 $= (0 \leq j \leq N \ \&\& \ s = (\sum_i \mid 0 \leq i < j \cdot a[i]) \ \&\& \ j < N)$   
 $\Rightarrow \mathbf{N} > \mathbf{j} \quad // \text{added } j \text{ to both sides}$   
 $= \mathbf{true} \quad // (N > j) = (j < N), P \ \&\& \ Q \Rightarrow P$

## Additional Proof Obligations



- To show: variant function is decreasing  
 $\{0 \leq j \leq N \ \&\& \ s = (\sum_{i \mid 0 \leq i < j} a[i]) \ \&\& \ j < N \ \&\& \ N-j = V\}$   
 $\{N-(j+1) < V\}$       // by assignment  
 $s := s + a[j];$   
 $\{N-(j+1) < V\}$       // by assignment  
 $j := j + 1;$   
 $\{N-j < V\}$
- Need to show:  
 $(0 \leq j \leq N \ \&\& \ s = (\sum_{i \mid 0 \leq i < j} a[i]) \ \&\& \ j < N \ \&\& \ N-j = V)$   
 $\Rightarrow (N-(j+1) < V)$   
Assume  $0 \leq j \leq N \ \&\& \ s = (\sum_{i \mid 0 \leq i < j} a[i]) \ \&\& \ j < N \ \&\& \ N-j = V$   
By weakening we have  $N-j = V$   
Therefore  $N-j-1 < V$   
But this is equivalent to  $N-(j+1) < V$ , so we are done.

## Factorial



- ```
{ N ≥ 1 }  
k := 1  
f := 1  
while (k < N) do  
  f := f * k  
  k := k + 1  
end  
{ f = N! }
```
- Loop invariant?
 - Variant function?

Factorial



```
{ N ≥ 1 }
```

```
k := 1
```

```
f := 1
```

```
while (k < N) do
```

```
  k := k + 1
```

```
  f := f * k
```

```
end
```

```
{ f = N! }
```

- Loop invariant?
 - $f = k! \ \&\& \ 0 \leq k \leq N$
- Variant function?
 - $N - k$

← Need to increment k **before** multiplying

Factorial



```
{ N ≥ 1 }
```

```
{ 1 = 1! && 0 ≤ 1 ≤ N }
```

```
k := 1
```

```
{ 1 = k! && 0 ≤ k ≤ N }
```

```
f := 1
```

```
{ f = k! && 0 ≤ k ≤ N }
```

```
while (k < N) do
```

```
  { f = k! && 0 ≤ k ≤ N && k < N && N - k = V }
```

```
  { f*(k+1) = (k+1)! && 0 ≤ k+1 ≤ N && N - (k+1) < V }
```

```
  k := k + 1
```

```
  { f*k = k! && 0 ≤ k ≤ N && N - k < V }
```

```
  f := f * k
```

```
  { f = k! && 0 ≤ k ≤ N && N - k < V }
```

```
end
```

```
{ f = k! && 0 ≤ k ≤ N && k ≥ N }
```

```
{ f = N! }
```

Factorial Obligations (1)


$$\begin{aligned} & (N \geq 1) \Rightarrow (1 = 1! \ \&\& \ 0 \leq 1 \leq N) \\ = & (N \geq 1) \Rightarrow (1 \leq N) \quad // \text{because } 1 = 1! \text{ and } 0 \leq 1 \\ = & \text{true} \quad // \text{because } (N \geq 1) = (1 \leq N) \end{aligned}$$

Factorial Obligations (2)


$$\begin{aligned} & (f = k! \ \&\& \ 0 \leq k \leq N \ \&\& \ k < N \ \&\& \ N - k = V) \\ \Rightarrow & (f^*(k+1) = (k+1)! \ \&\& \ 0 \leq k+1 \leq N \ \&\& \ N - (k+1) < V) \\ = & (f = k! \ \&\& \ 0 \leq k < N \ \&\& \ N - k = V) \\ \Rightarrow & (f^*(k+1) = k! * (k+1) \ \&\& \ 0 \leq k+1 \leq N \ \&\& \ N - k - 1 < V) \\ & // \text{by simplification and } (k+1)! = k! * (k+1) \\ \text{Assume } & (f = k! \ \&\& \ 0 \leq k < N \ \&\& \ N - k = V) \\ \text{Check each RHS clause:} \\ \bullet & (f^*(k+1) = k! * (k+1)) \\ & = (f = k!) \quad // \text{division by } (k+1) \text{ (nonzero by assumption)} \\ & = \text{true} \quad // \text{by assumption} \\ \bullet & 0 \leq k+1 \\ & = \text{true} \quad // \text{by assumption that } 0 \leq k \\ \bullet & k+1 \leq N \\ & = \text{true} \quad // \text{by assumption that } k < N \\ \bullet & N - k - 1 < V \\ & = N - k - 1 < N - k \quad // \text{by assumption that } N - k = V \\ & = N - 1 < N \quad // \text{by addition of } k \\ & = \text{true} \quad // \text{by properties of } < \end{aligned}$$

Factorial Obligations (3)



$(f = k! \ \&\& \ 0 \leq k \leq N \ \&\& \ k \geq N) \Rightarrow (f = N!)$

Assume $f = k! \ \&\& \ 0 \leq k \leq N \ \&\& \ k \geq N$

Then $k=N$ by $k \leq N \ \&\& \ k \geq N$

So $f = N!$ by substituting $k=N$