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?

```c
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**
  - \( x = 3 \)
  - \( y > x \)
  - \( (x \neq 0) \Rightarrow (y+z = w) \)
  - \( s = \sum_{i=1}^{n} a[i] \)
  - \( \forall i \in 1..n . a[i] > a[i-1] \)
  - 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.

/*@
   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

- \{ true \} x := 5 \{ x=5 \}
- \{ x = y \} x := x + 3 \{ x = y + 3 \}
- \{ x > -1 \} x := x * 2 + 3 \{ x > 1 \}
- \{ x=a \} if (x < 0) then x := -x \{ x=|a| \}
- \{ false \} x := 3 \{ x = 8 \}
- \{ x < 0 \} while (x!=0) x := x-1 \}
  - no such triple!
**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 \Rightarrow Q’, then Q is the strongest postcondition
  of S with respect to P
  - check: x = 10 \Rightarrow true
  - check: x = 10 \Rightarrow x > 0
  - check: x = 10 \Rightarrow x = 10 \ || \ 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

• 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 \( y := z \) else \( y := -z \) \{ y > 5 \}
  - What is the weakest precondition P?
Hoare Logic Rules

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

- Conditional rule
  - \( wp(\text{if } B \text{ then } S \text{ else } T, Q) \)
    - \( = B \Rightarrow wp(S,Q) \&\& \neg B \Rightarrow wp(T,Q) \)
  - \( wp(\text{if } x>0 \text{ then } y:=z \text{ else } y:=-z, y>5) \)
  - \( = x>0 \Rightarrow wp(y:=z,y>5) \&\& x\leq0 \Rightarrow wp(y:=-z,y>5) \)
  - \( = x>0 \Rightarrow z > 5 \&\& x\leq0 \Rightarrow -z > 5 \)
  - \( = x>0 \Rightarrow z > 5 \&\& x\leq0 \Rightarrow z < -5 \)

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 \( \text{Inv} \) such that:
    • \( P \Rightarrow \text{Inv} \)
    • The invariant is initially true
    • \( \{ \text{Inv} \land B \} S \{ \text{Inv} \} \)
    • Each execution of the loop preserves the invariant
    • \( (\text{Inv} \land \neg B) \Rightarrow Q \)
    • 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 \}
  \begin{align*}
    & j := 0; \\
    & s := 0;
  \end{align*}

  while \( j < N \) do
    \begin{align*}
      & j := j + 1; \\
      & s := s + a[j];
    \end{align*}
  end
  \{ s = (\sum_{i} | 0 \leq i < N \cdot a[i]) \}
Loop Example

- Prove array sum correct

\[
\{ N \geq 0 \}
\]
\[
j := 0;
\]
\[
s := 0;
\]
\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \}
\]
while \( j < N \) do
\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \text{ && } j < N \}
\]
\[
j := j + 1;
\]
\[
s := s + a[j];
\]
\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \}
\] end
\[
\{ s = (\Sigma i \mid 0 \leq i < N \cdot a[i]) \}
\]

Proof Obligations

- Invariant is initially true

\[
\{ N \geq 0 \}
\]
\[
j := 0;
\]
\[
s := 0;
\]
\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \}
\]
- Invariant is maintained

\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \text{ && } j < N \}
\]
\[
j := j + 1;
\]
\[
s := s + a[j];
\]
\[
\{ 0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \}
\]
- Invariant and exit condition implies postcondition

\[
0 \leq j \leq N \text{ && } s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \text{ && } j \geq N
\]
\[
\Rightarrow s = (\Sigma i \mid 0 \leq i < N \cdot a[i])
\]
Proof Obligations

- Invariant is initially true
  
  \[
  \{ N \geq 0 \} \\
  \{ 0 \leq 0 \leq N \land \& \& 0 = (\Sigma i | 0 \leq i < 0 \cdot a[i]) \} \quad \text{// by assignment rule} \\
  j := 0; \\
  \{ 0 \leq j \leq N \land \& \& 0 = (\Sigma i | 0 \leq i < j \cdot a[i]) \} \quad \text{// by assignment rule} \\
  s := 0; \\
  \{ 0 \leq j \leq N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i]) \} 
  \]

- Need to show that:
  
  \[
  (N \geq 0) \Rightarrow (0 \leq 0 \leq N \land \& \& 0 = (\Sigma i | 0 \leq i < 0 \cdot a[i])) \\
  = (N \geq 0) \Rightarrow (0 \leq N \land \& \& 0 = 0) \quad \text{// 0 \leq 0 is true, empty sum is 0} \\
  = (N \geq 0) \Rightarrow (0 \leq N) \quad \text{// 0=0 is true, P \& \& true is P} \\
  = \text{true}
  \]

Proof Obligations

- Invariant is maintained
  
  \[
  \{ 0 \leq j \leq N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i]) \land \& \& j < N \} \\
  \{ 0 \leq j + 1 \leq N \land \& \& s + a[j+1] = (\Sigma i | 0 \leq i < j+1 \cdot a[i]) \} \quad \text{// by assignment rule} \\
  j := j + 1; \\
  \{ 0 \leq j \leq N \land \& \& s + a[j] = (\Sigma i | 0 \leq i < j \cdot a[i]) \} \quad \text{// by assignment rule} \\
  s := s + a[j]; \\
  \{ 0 \leq j \leq N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i]) \}
  \]

- Need to show that:
  
  \[
  (0 \leq j \leq N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i]) \land \& \& j < N) \\
  \Rightarrow (0 \leq j + 1 \leq N \land \& \& s + a[j+1] = (\Sigma i | 0 \leq i < j+1 \cdot a[i])) \\
  = (0 \leq j \leq N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i])) \\
  \Rightarrow (-1 \leq j < N \land \& \& s + a[j+1] = (\Sigma i | 0 \leq i < j \cdot a[i]) + a[j]) \quad \text{// simplify bounds of j} \\
  = (0 \leq j < N \land \& \& s = (\Sigma i | 0 \leq i < j \cdot a[i])) \\
  \Rightarrow (-1 \leq j < N \land \& \& s + a[j+1] = (\Sigma i | 0 \leq i < j \cdot a[i]) + a[j]) \quad \text{// separate last element} \\
  \text{// we have a problem – we need a[j+1] and a[j] to cancel out}
  \]
Where's the error?

- Prove array sum correct

\[
\begin{align*}
\{ \text{N } \geq 0 \} \\
j &:= 0; \\
s &:= 0; \\
\text{while (j < N) do} \\
\quad j &:= j + 1; \\
\quad s &:= s + a[j]; \\
\text{end} \\
\{ \text{s }= \langle \sum i \mid 0 \leq i < N \cdot a[i] \rangle \} 
\end{align*}
\]

Need to add element before incrementing j

Corrected Code

- Prove array sum correct

\[
\begin{align*}
\{ \text{N } \geq 0 \} \\
j &:= 0; \\
s &:= 0; \\
\text{while (j < N) do} \\
\quad s &:= s + a[j]; \\
\quad j &:= j + 1; \\
\text{end} \\
\{ \text{s }= \langle \sum i \mid 0 \leq i < N \cdot a[i] \rangle \} 
\end{align*}
\]
Proof Obligations

• Invariant is maintained
  \(0 \leq j \leq N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \land j < N\)
  \(0 \leq j + 1 \leq N \land \land s + a[j] = (\Sigma i \mid 0 \leq i < j + 1 \cdot a[i])\) // by assignment rule
  \(s := s + a[j];\)
  \(0 \leq j + 1 \leq N \land \land s = (\Sigma i \mid 0 \leq i < j + 1 \cdot a[i])\) // by assignment rule
  \(j := j + 1;\)
  \(0 \leq j \leq N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i])\)

• Need to show that:
  \(0 \leq j \leq N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \land j < N\)
  \(\Rightarrow (0 \leq j + 1 \leq N \land \land s + a[j] = (\Sigma i \mid 0 \leq i < j + 1 \cdot a[i])\)
  \(= (0 \leq j < N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]))\)
  \(\Rightarrow (-1 \leq j < N \land \land s + a[j] = (\Sigma i \mid 0 \leq i < j + 1 \cdot a[i])\) // simplify bounds of j
  \(= (0 \leq j < N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]))\)
  \(\Rightarrow (-1 \leq j < N \land \land s + a[j] = (\Sigma i \mid 0 \leq i < j \cdot a[i]) + a[j])\) // separate last part of sum
  \(= (0 \leq j < N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]))\)
  \(\Rightarrow (-1 \leq j < N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]))\) // subtract a[j] from both sides
  \(= true\) // 0 ≤ j ⇒ -1 ≤ j

Proof Obligations

• Invariant and exit condition implies postcondition
  \(0 \leq j \leq N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \land j \geq N\)
  \(\Rightarrow s = (\Sigma i \mid 0 \leq i < N \cdot a[i])\)
  \(= 0 \leq j \land \land j = N \land \land s = (\Sigma i \mid 0 \leq i < j \cdot a[i])\)
  \(\Rightarrow s = (\Sigma i \mid 0 \leq i < N \cdot a[i])\)
  // because (j ≤ N \land \land j ≥ N) = (j = N)
  \(= 0 \leq N \land \land s = (\Sigma i \mid 0 \leq i < N \cdot a[i]) \Rightarrow s = (\Sigma i \mid 0 \leq i < N \cdot a[i])\)
  // by substituting N for j, since j = N
  \(= true\) // because P \land \land Q ⇒ 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:
    - \(P \Rightarrow Inv\)
      - 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
- Total correctness
  - Loop will terminate
Termination

- How would you prove this program terminates?
  \[
  \{ N \geq 0 \} \\
  j := 0; \\
  s := 0; \\
  \]
  while (j < N) do \\
  \[
  s := s + a[j]; \\
  j := j + 1; \\
  \]
  end \\
  \[
  \{ s = (\Sigma i \mid 0 \leq i < N \cdot a[i]) \} \\
  \]

Total Correctness for Loops

- \{P\} while B do S \{Q\}
- Partial correctness:
  - Find an invariant Inv such that:
    - \( P \Rightarrow Inv \)
      - The invariant is initially true
    - \( \{ Inv \land \neg B \} \Rightarrow \neg Inv \)
      - Each execution of the loop preserves the invariant
    - \( (Inv \land \neg B) \Rightarrow Q \)
      - 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
    - \( (Inv \land B) \Rightarrow v > 0 \)
      - The variant function evaluates to a finite integer value greater than zero at the beginning of the loop
    - \( \{ Inv \land B \land v=V \} \Rightarrow (v < V) \)
      - 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 \leq j \leq N \&\& s = (\Sigma i \mid 0 \leq i < j \cdot a[i]) \&\& j < N\}
    s := s + a[j];
    j := j + 1;
    \{0 \leq j \leq N \&\& s = (\Sigma i \mid 0 \leq i < j \cdot 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
  \[ (0 \leq j \leq N \land s = (\sum_{i=0}^{j} a[i]) \land j < N) \Rightarrow N-j > 0 \]
- To show: variant function is decreasing
  \[ \{0 \leq j \leq N \land s = (\sum_{i=0}^{j} a[i]) \land j < N \land N-j = V\} \]
  \[ s := s + a[j]; \]
  \[ j := j + 1; \]
  \[ \{N-j < V\} \]

Additional Proof Obligations

- To show: variant function initially positive
  \[ (0 \leq j \leq N \land s = (\sum_{i=0}^{j} a[i]) \land j < N) \Rightarrow N-j > 0 \]
  \[ = (0 \leq j \leq N \land s = (\sum_{i=0}^{j} a[i]) \land j < N) \Rightarrow N > j \quad // added j to both sides \]
  \[ = \text{true} \quad // (N > j) = (j < N), P \land Q \Rightarrow P \]
Additional Proof Obligations

- To show: variant function is decreasing
  \( \{0 \leq j \leq N ~\&\&~ s = (\sum_i |0 \leq i < j \cdot a[i]) ~\&\&~ j < N ~\&\&~ N-j = V\} \)
  \( \{N-(j+1) < V\} \)  // by assignment
  \( s := s + a[j] \);
  \( \{N-(j+1) < V\} \)  // by assignment
  \( j := j + 1 \);
  \( \{N-j < V\} \)
- Need to show:
  \( \{0 \leq j \leq N ~\&\&~ s = (\sum_i |0 \leq i < j \cdot a[i]) ~\&\&~ j < N ~\&\&~ N-j = V\} \)
  \( \Rightarrow \{N-(j+1) < V\} \)
  Assume \( 0 \leq j \leq N ~\&\&~ s = (\sum_i |0 \leq i < j \cdot a[i]) ~\&\&~ j < N ~\&\&~ N-j = V \)
  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

\( \{ N \geq 1 \} \)
\( k := 1 \)
\( f := 1 \)
while \( (k < N) \) do
  \( f := f \cdot k \)
  \( k := k + 1 \)
end
\( \{ f = N! \} \)
- Loop invariant?
- Variant function?
Factorial

\{ N \geq 1 \}
\begin{align*}
& k := 1 \\
& f := 1 \\
\text{while } (k < N) \text{ do} \\
& \quad k := k + 1 \\
& \quad f := f \times k \\
\end{align*}
\text{end}
\{ f = N! \}

- Loop invariant?
  - \( f = k! \text{ } \&\& \text{ } 0 \leq k \leq N \)
- Variant function?
  - \( N-k \)

---

Factorial

\{ N \geq 1 \}
\begin{align*}
& 1 = 1! \text{ } \&\& \text{ } 0 \leq 1 \leq N \\
& k := 1 \\
& 1 = k! \text{ } \&\& \text{ } 0 \leq k \leq N \\
& f := 1 \\
& f = k! \text{ } \&\& \text{ } 0 \leq k \leq N \\
\text{while } (k < N) \text{ do} \\
& \quad \{ f = k! \text{ } \&\& \text{ } 0 \leq k \leq N \text{ } \&\& \text{ } k < N \text{ } \&\& \text{ } k \leq N \} \\
& \quad \{ f \times (k+1) = (k+1)! \text{ } \&\& \text{ } 0 \leq k+1 \leq N \text{ } \&\& \text{ } N \times (k+1) < V \} \\
& \quad k := k + 1 \\
& \quad \{ f \times k = k! \text{ } \&\& \text{ } 0 \leq k \leq N \text{ } \&\& \text{ } N \times k < V \} \\
& \quad f := f \times k \\
& \quad \{ f = k! \text{ } \&\& \text{ } 0 \leq k \leq N \text{ } \&\& \text{ } N \times k < V \} \\
\end{align*}
\text{end}
\{ f = k! \text{ } \&\& \text{ } 0 \leq k \leq N \text{ } \&\& \text{ } k \geq N \}
\{ f = N! \}
Factorial Obligations (1)

\[(N \geq 1) \Rightarrow (1 = 1! \land 0 \leq 1 \leq N) \]
\[= (N \geq 1) \Rightarrow (1 \leq N) \quad // \text{because } 1 = 1! \text{ and } 0 \leq 1 \]
\[= \text{true} \quad // \text{because } (N \geq 1) = (1 \leq N) \]

Factorial Obligations (2)

\[f = k! \land 0 \leq k \leq N \land k < N \land N-k = V \]
\[\Rightarrow (f+(k+1) = (k+1)! \land 0 \leq k+1 \leq N \land N-(k+1) < V) \]
\[= (f = k! \land 0 \leq k < N \land N-k = V) \]
\[\Rightarrow (f+(k+1) = k!(k+1) \land 0 \leq k+1 \leq N \land N-k-1 < V) \]
// by simplification and \((k+1)! = k!(k+1)\)
Assume \((f = k! \land 0 \leq k < N \land N-k = V)\)
Check each RHS clause:
  \(f = k! \land 0 \leq k < N \land N-k = V\)
  // division by \((k+1)\) (nonzero by assumption)
  \[= \text{true} \quad // \text{by assumption} \]
  \[0 \leq k+1 \]
  \[= \text{true} \quad // \text{by assumption that } 0 \leq k\]
  \[k+1 \leq N \]
  \[= \text{true} \quad // \text{by assumption that } k < N\]
  \[N-k-1 < V \]
  \[= N-k-1 < N-k \quad // \text{by assumption that } N-k = V\]
  \[= N < k \quad // \text{by addition of } k\]
  \[= \text{true} \quad // \text{by properties of } <\]
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 \(k=N\)