COL874: Advanced Compiler Techniques

Modules 186-190

Arpit Saxena

11 November 2021

Hoare Logic

1. Assignment Rule: `{P[x:=e]} x:=e {P}`

Hoare Logic

1. Assignment Rule: `{P[x:=e]} x:=e {P}`
2. Composition Rule

$$\frac{\{\mathtt{P}\}\ C_1\ \{\mathtt{R}\} \qquad \{\mathtt{R}\}\ C_2\ \{\mathtt{Q}\}}{\{\mathtt{P}\}\ C_1;\ C_2\ \{\mathtt{Q}\}}$$

Hoare Logic

1. Assignment Rule: {P[x:=e]} x:=e {P}
2. Composition Rule

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

3. if-then-else rule

$$\frac{\{P \wedge b\}\ C_1\ \{Q\} \qquad \{P \wedge \neg b\}\ C_2\ \{Q\}}{\{P\}\ \texttt{if b then}\ C_1\ \texttt{else}\ C_2\ \{Q\}}$$

Hoare Logic

1. Assignment Rule: {P[x:=e]} x:=e {P}
2. Composition Rule

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

3. if-then-else rule

$$\frac{\{P \land b\}\ C_1\ \{Q\} \qquad \{P \land \neg b\}\ C_2\ \{Q\}}{\{P\}\ \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \{Q\}}$$

4. Consequence rule

$$\frac{(P \implies P') \qquad \{P'\}\ C\ \{Q'\} \qquad (Q' \implies Q)}{\{P\}\ C\ \{Q\}}$$

Hoare Logic Rule for while

5 While rule:

$$\frac{\{P \wedge b\}\ C\ \{P\}}{\{P\}\ \texttt{while(b)}\ C\ \{P \wedge \neg b\}}$$

Here, P is a loop invariant.

Hoare Logic Rule for while

5 While rule:

$$\frac{\{\mathtt{P} \wedge \mathtt{b}\}\ \mathtt{C}\ \{\mathtt{P}\}}{\{\mathtt{P}\}\ \mathtt{while(b)}\ \mathtt{C}\ \{\mathtt{P} \wedge \neg b\}}$$

Here, P is a loop invariant.

Example

{x $\geq$ 0}
while x $\neq$ 0
x := x - 1
{x = 0}

Hoare Logic Rule for while

5 While rule:

$$\frac{\{\mathtt{P} \wedge \mathtt{b}\}\ \mathtt{C}\ \{\mathtt{P}\}}{\{\mathtt{P}\}\ \mathtt{while(b)}\ \mathtt{C}\ \{\mathtt{P} \wedge \neg b\}}$$

Here, P is a loop invariant.

Example

```
{x ≥ 0} // P
while x ≠ 0 // b
x := x - 1
{x = 0} // Q
```

From the inference rule:

$\{\mathtt{x} \geq 0 \wedge \mathtt{x} \neq 0\}\ \mathtt{x := x - 1}\ \{\mathtt{x} \geq 0\}$

Hoare Logic Rule for while

Example

$\{(\texttt{sum} = 0) \wedge (n_0 \geq 0) \wedge (\texttt{n} = n_0\}$

```
while (n != 0) {
sum := sum + n;
n := n - 1;
}
```

$\{\texttt{sum} = n_0(n_0 + 1)/2\}$

Hoare Logic Rule for while

Example

Try to pattern match:

```
{(sum = 0) ∧ (n₀ ≥ 0) ∧ (n = n₀} // P
while (n != 0) {
sum := sum + n;
n := n - 1;
}
{sum = n₀(n₀ + 1)/2} // Q
```

Hoare Logic Rule for while

Example

```
{(sum = 0) ∧ (n0 ≥ 0) ∧ (n = n0} // P
while (n != 0) {
sum := sum + n;
n := n - 1;
}
{sum = n0(n0 + 1)/2} // Q
```

Q' = P $\wedge \neg b$ = {(sum = 0) $\wedge$ ((n = n_0) $\geq$ 0) $\wedge$ (n = 0)}

Clearly, Q' $\implies$ sum = n_0(n_0+1)/2

So can try to prove with postcondition Q' $\iff$ (sum = 0 $\wedge$ n = 0 $\wedge$ n_0 = 0), which is not a loop invariant and can be shown formally.

Hoare Logic Rule for while

Example

```
{(sum = 0) ∧ (n0 ≥ 0) ∧ (n = n0} // P
while (n != 0) {
sum := sum + n;
n := n - 1;
}
{sum = n0(n0 + 1)/2} // Q
```

So we need to find P' such that

```
{P' ∧ b}
sum := sum + n;
n := n - 1;
{P'}
```

and P $\implies$ P' and P' $\wedge \neg b \implies$ Q

Hoare Logic Rule for while

Example

```
{(sum = 0) ∧ (n0 ≥ 0) ∧ (n = n0} // P
while (n != 0) {
sum := sum + n;
n := n - 1;
}
{sum = n0(n0 + 1)/2} // Q
```

It can be shown formally that `P' : sum = (`n_0`-n)(`n_0`+n+1)/2` works.

Finding the required P' (or Q')

- Soundness: No erroneous fact can be derived by Hoare logic.
- Completeness: All true facts can be derived by Hoare logic.

Finding the required P' (or Q')

- Soundness: No erroneous fact can be derived by Hoare logic.
- Completeness: All true facts can be derived by Hoare logic.

Theorem (Godel Incompleteness Theorem)

If the first-order logic includes arithmetic, there exists no complete axiomatisation of $\Longrightarrow$ *in the consequence rule.*

In simpler words, not always possible to find the required P'. So, Hoare logic is incomplete.

Relative Completeness

All true facts can be derived by Hoare logic provided:

- The first order assertion language is rich enough to express loop invariants
- All first-order theorems needed in the consequence rules are given.

Introduction

- Hoare logic is presented as a deductive system. We don't have any strategy to build the deductions

Introduction

- Hoare logic is presented as a deductive system. We don't have any strategy to build the deductions
- Weakest Preconditions and Strongest Postconditions are complete strategies to build valid Hoare logic deductions

Introduction

- Hoare logic is presented as a deductive system. We don't have any strategy to build the deductions
- Weakest Preconditions and Strongest Postconditions are complete strategies (assuming invariants are provided by the programmer) to build valid Hoare logic deductions

Weakest Preconditions

For a statement $\mathtt{S}$ and a postcondition $\mathtt{R}$, a weakest precondition is a predicate $\mathtt{Q}$ such that for a any precondition $\mathtt{P}$:

$$\{\mathtt{P}\}\ \mathtt{S}\ \{\mathtt{R}\} \iff (\mathtt{P} \implies \mathtt{Q})$$

Weakest Preconditions

For a statement `S` and a postcondition `R`, a weakest precondition is a predicate `Q` such that for a any precondition `P`:

$$\{\mathtt{P}\}\ \mathtt{S}\ \{\mathtt{R}\} \iff (\mathtt{P} \implies \mathtt{Q})$$

Theorem (Uniqueness of Weakest Precondition)

If both Q and Q' are weakest preconditions, then by definition:
$\{\mathtt{Q}\}\ \mathtt{S}\ \{\mathtt{R}\}$ holds $\implies$ $(\mathtt{Q'} \implies \mathtt{Q})$

Weakest Preconditions

For a statement S and a postcondition R, a weakest precondition is a predicate Q such that for a any precondition P:

$$\{P\}\ S\ \{R\} \iff (P \implies Q)$$

Theorem (Uniqueness of Weakest Precondition)

If both Q and Q' are weakest preconditions, then by definition:

$\{Q\}\ S\ \{R\}$ *holds* $\implies (Q' \implies Q)$

$\{Q'\}\ S\ \{R\}$ *holds* $\implies (Q \implies Q')$

Weakest Preconditions

For a statement S and a postcondition R, a weakest precondition is a predicate Q such that for a any precondition P:

$$\{P\}\ S\ \{R\} \iff (P \implies Q)$$

Theorem (Uniqueness of Weakest Precondition)

If both Q and Q' are weakest preconditions, then by definition:

$\{Q\}\ S\ \{R\}$ *holds* $\implies (Q' \implies Q)$

$\{Q'\}\ S\ \{R\}$ *holds* $\implies (Q \implies Q')$

$\implies Q = Q'$

Weakest Precondition Rules

Notation: wp(S, R) denotes the weakest precondition for statement S and postcondition R.

Weakest Precondition Rules

- `wp(skip, R) = R` Skip Rule

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x ← e]` — Assignment Rule

Weakest Precondition Rules

- `wp(skip, R) = R` Skip Rule
- `wp(x := e, R) = R[x ← e]` Assignment Rule

Example

`wp(x := x - 5, x > 10) = (x > 10)[x ← x - 5]`

Weakest Precondition Rules

- `wp(skip, R) = R` Skip Rule
- `wp(x := e, R) = R[x ← e]` Assignment Rule

Example

`wp(x := x - 5, x > 10) = x - 5 > 10`

Weakest Precondition Rules

- `wp(skip, R) = R` Skip Rule
- `wp(x := e, R) = R[x ← e]` Assignment Rule

Example

`wp(x := x - 5, x > 10) = x > 15`

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule

Example

`wp(x := x - 5; x := x * 2, x > 20)`

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x ← e]` — Assignment Rule
- `wp(`S_1`; `S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule

Example

`wp(x := x - 5; x := x * 2, x > 20)`
$=$ `wp(x := x - 5, wp(x := x * 2, x > 20))`

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule

Example

`wp(x := x - 5; x := x * 2, x > 20)`

= `wp(x := x - 5, x * 2 > 20)`

Weakest Precondition Rules

- wp(skip, R) = R — Skip Rule
- wp(x := e, R) = R[x ← e] — Assignment Rule
- wp(S_1; S_2, R = wp(S_1, wp(S_2, R)) — Sequence Rule

Example

wp(x := x - 5; x := x * 2, x > 20)
= (x - 5) * 2 > 20

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule

Example

`wp(x := x - 5; x := x * 2, x > 20)`

$\iff$ `x > 15`

Weakest Precondition Rules

- wp(skip, R) = R — Skip Rule
- wp(x := e, R) = R[x $\leftarrow$ e] — Assignment Rule
- wp(S_1; S_2, R = wp(S_1, wp(S_2, R)) — Sequence Rule
- wp(if e then S_1 else S_2, R) = (e $\implies$ wp(S_1, R)) $\wedge$ ($\neg e$ $\implies$ wp(S_2, R) — Conditional Rule

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule
- `wp(if e then` S_1 `else` S_2`, R) = (e` $\implies$ `wp(`S_1`, R))` $\wedge$ `(`$\neg e$ $\implies$ `wp(`S_2`, R)` — Conditional Rule

Example

`wp(if x < y then x := y else skip, x >= y)`

Weakest Precondition Rules

- wp(skip, R) = R — Skip Rule
- wp(x := e, R) = R[x ← e] — Assignment Rule
- wp(S_1; S_2, R = wp(S_1, wp(S_2, R)) — Sequence Rule
- wp(if e then S_1 else S_2, R) = (e $\implies$ wp(S_1, R)) $\wedge$ ($\neg e$ $\implies$ wp(S_2, R) — Conditional Rule

Example

wp(if x < y then x := y else skip, x >= y)
= ((x < y) $\implies$ wp(x := y, x >= y)) $\wedge$ ((x >= y) $\implies$ wp(skip, x >= y)

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule
- `wp(if e then` S_1 `else` S_2`, R) = (e` $\Longrightarrow$ `wp(`S_1`, R))` $\wedge$ $(\neg e$ $\Longrightarrow$ `wp(`S_2`, R)` — Conditional Rule

Example

`wp(if x < y then x := y else skip, x >= y)`
`= ((x < y)` $\Longrightarrow$ `wp(x := y, x >= y))` $\wedge$ `((x >= y)` $\Longrightarrow$
`x >= y)`

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x ← e]` — Assignment Rule
- wp(S_1; S_2, R = wp(S_1, wp(S_2, R)) — Sequence Rule
- wp(if e then S_1 else S_2, R) = (e $\implies$ wp(S_1, R)) $\wedge$ ($\neg e$ $\implies$ wp(S_2, R) — Conditional Rule

Example

`wp(if x < y then x := y else skip, x >= y)`
`= ((x < y)` $\implies$ `y >= y)) ∧ ((x >= y)` $\implies$ `x >= y)`

Weakest Precondition Rules

- `wp(skip, R) = R` — Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` — Sequence Rule
- `wp(if e then` S_1 `else` S_2`, R) = (e` $\Longrightarrow$ `wp(`S_1`, R))` $\wedge$ `(`$\neg e$ $\Longrightarrow$ `wp(`S_2`, R)` — Conditional Rule

Example

`wp(if x < y then x := y else skip, x >= y)`
`= ((x < y)` $\Longrightarrow$ `true))` $\wedge$ `true`

Weakest Precondition Rules

- `wp(skip, R) = R` Skip Rule
- `wp(x := e, R) = R[x ← e]` Assignment Rule
- wp(S_1 ; S_2 , R = wp(S_1 , wp(S_2 , R)) Sequence Rule
- wp(if e then S_1 else S_2 , R) = (e $\Longrightarrow$ wp(S_1 , R)) $\wedge$ ($\neg e$ $\Longrightarrow$ wp(S_2 , R) Conditional Rule

Example

`wp(if x < y then x := y else skip, x >= y)`
`= true`

Precondition Semantics

```
{?} S {R}
```

We want to find a predicate just prior to the execution of S such that:

Precondition Semantics

```
{?} S {R}
```

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.

Precondition Semantics

```
{?} S {R}
```

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.
  completes execution and control reaches the PC just after S

Precondition Semantics

$$\{?\}\ S\ \{R\}$$

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.
- **Option 2**: S terminates and R holds.

Precondition Semantics

`{?} S {R}`

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.
  Written as `{?} S {R}` Hoare triple for partial correctness
- **Option 2**: S terminates and R holds.
  Written as `[?] S [R]` Hoare triple for total correctness

Precondition Semantics

`{?} S {R}`

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.
  Written as `{?} S {R}` Hoare triple for partial correctness
  Weakest Liberal Precondition (wlp)
- **Option 2**: S terminates and R holds.
  Written as `[?] S [R]` Hoare triple for total correctness
  Weakest Precondition (wp)

Precondition Semantics

$$\texttt{\{?\} S \{R\}}$$

We want to find a predicate just prior to the execution of S such that:

- **Option 1**: If S terminates, then R holds.
  Written as `{?} S {R}` Hoare triple for partial correctness
  Weakest Liberal Precondition (wlp)
- **Option 2**: S terminates and R holds.
  Written as `[?] S [R]` Hoare triple for total correctness
  Weakest Precondition (wp)

We can observe that $\texttt{wp} \implies \texttt{wlp}$

Common rules for WP and WLP

- `wp(skip, R) = R` Skip Rule
- `wp(x := e, R) = R[x` $\leftarrow$ `e]` Assignment Rule
- `wp(`S_1`;` S_2`, R = wp(`S_1`, wp(`S_2`, R))` Sequence Rule
- `wp(if e then` S_1 `else` S_2`, R) = (e` $\implies$ `wp(`S_1`, R))` $\wedge$ $(\neg e \implies$ `wp(`S_2`, R))` Conditional Rule

Common rules for WP and WLP

- `wlp(skip, R) = R` — Skip Rule
- `wlp(x := e, R) = R[x` $\leftarrow$ `e]` — Assignment Rule
- `wlp(`S_1`;` S_2`, R = wp(`S_1`, wlp(`S_2`, R))` — Sequence Rule
- `wlp(if e then` S_1 `else` S_2`, R) = (e` $\Longrightarrow$ `wlp(`S_1`, R))` $\wedge$ `(`$\neg$`e` $\Longrightarrow$ `wlp(`S_2`, R))` — Conditional Rule

Abort WP Rule

`abort` aborts the program, so control doesn't reach the PC after it. So by our definition, it is non-terminating.

Abort WP Rule

`abort` aborts the program, so control doesn't reach the PC after it. So by our definition, it is non-terminating.

`wp(abort, R) = false` `wlp(abort, R) = true`

Abort WP Rule

`abort` aborts the program, so control doesn't reach the PC after it. So by our definition, it is non-terminating.

`wp(abort, R) = false` `wlp(abort, R) = true`

Example

`wp(x := y/z, y = x*z) = (z != 0)`
`wlp(x := y/z, y = x*z) = true`
Note that this is assuming division by zero causes abort.

Partial Correctness of while loop

Now, we want to find `wlp(while(e) S, R)`.

In general, both wp and wlp are undecidable.

Partial Correctness of while loop

Now, we want to find `wlp(while(e) S, R)`.

In general, both wp and wlp are undecidable.
Because of the halting problem

Partial Correctness of while loop

Now, we want to find `wlp(while(e) S, R)`.

In general, both wp and wlp are undecidable.
Because we want the weakest such condition, we have to find the loop invariant, which is undecidable.

Partial Correctness of while loop

We need a loop invariant I such that

- I should hold in the beginning
- {I $\wedge$ e} S {I} holds
- {I $\wedge$ $\neg e$} skip {R} holds

Partial Correctness of while loop

We need a loop invariant I such that

- I should hold in the beginning (Initial state x)
- {I ∧ e} S {I} holds
- {I ∧ ¬e} skip {R} holds

} (For all possible states y)

Partial Correctness of while loop

We need a loop invariant I such that

- I should hold in the beginning (Initial state x)
- {I ∧ e} S {I} holds
- {I ∧ $\neg e$} skip {R} holds

} (For all possible states y)

wlp(while(e) S, R)
= I
∧ $\forall y$ ((I ∧ e) $\implies$ wlp(S, I))[x ← y]
∧ $\forall y$ ((I ∧ $\neg e$) $\implies$ R)[x ← y]
We are interested in the weakest such I.

Partial Correctness of while loop

Example

```
wlp(while(x > 0) x--, {x == 0}) == ?
```

Partial Correctness of while loop

Example

```
wlp(while(x > 0) x--, {x == 0}) == ?
```

Candidate I: x = -1

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x = -1
Not a precondition because does not statisfy:
$\forall y \ ((\mathtt{I} \wedge \neg e) \implies \mathtt{R})[\mathtt{x} \leftarrow \mathtt{y}]$

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x = -1

Not a precondition because does not statisfy:

$\forall y$ ((`x = -1` $\wedge$ `x <= 0`) $\implies$ `x = 0`)[x $\leftarrow$ y]

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x = -1

Not a precondition because does not statisfy:

$\forall y$ `((y = -1` $\wedge$ `y <= 0)` $\implies$ `y = 0)`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x = -1

Not a precondition because does not statisfy:

$\forall y$ `((y = -1` $\wedge$ `y <= 0)` $\implies$ `y = 0)` ✗Not provable

Partial Correctness of while loop

Example

```
wlp(while(x > 0) x--, {x == 0}) == ?
```

Candidate I: $x > 1$

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: $x > 1$

Does not satisfy:

$\forall y$ `((I` $\wedge$ `e)` $\implies$ `wlp(S, I))[x` $\leftarrow$ `y]`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x > 1
Does not satisfy:
$\forall y$ `((x > 1` $\wedge$ `x > 0)` $\implies$ `wlp(x := x - 1, x > 1))[x` $\leftarrow$ `y]`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x > 1

Does not satisfy:

$\forall y$ `(x > 1` $\wedge$ `x > 0)` $\implies$ `x-1 > 1)[x` $\leftarrow$ `y]`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: $x > 1$

Does not satisfy:

$\forall y\ (y > 1 \land y > 0) \implies y-1 > 1)$

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x > 1

Does not satisfy:

$\forall y$ `(y > 1` $\wedge$ `y > 0)` $\Longrightarrow$ `y-1 > 1)` ✗Not provable

Partial Correctness of while loop

Example

```
wlp(while(x > 0) x--, {x == 0}) == ?
```

Candidate I: x >= 0

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x >= 0

Satisfies both:

- $\forall y$ `((I` $\wedge$ `e)` $\implies$ `wlp(S, I))[x` $\leftarrow$ `y]`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x $>=$ 0

Satisfies both:

- $\forall y$ `((I ∧ e)` $\implies$ `wlp(S, I))[x ← y]`
  $\forall y$ `(y >= 0 ∧ y > 0)` $\implies$ `y - 1 >= 0`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x >= 0

Satisfies both:

- $\forall y$ `((I` $\wedge$ `e)` $\implies$ `wlp(S, I))[x` $\leftarrow$ `y]`
  $\forall y$ `(y >= 0` $\wedge$ `y > 0)` $\implies$ `y - 1 >= 0`
  Provable

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x >= 0

Satisfies both:

- $\forall y$ `((I` $\wedge$ `e)` $\implies$ `wlp(S, I))[x` $\leftarrow$ `y]`
  $\forall y$ `(y >= 0` $\wedge$ `y > 0)` $\implies$ `y - 1 >= 0`
  Provable
- $\forall y$ `((I` $\wedge$ $\neg e$`)` $\implies$ `R)[x` $\leftarrow$ `y]`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x $>=$ 0

Satisfies both:

- $\forall y$ `((I` $\wedge$ `e)` $\implies$ `wlp(S, I))[x` $\leftarrow$ `y]`
  $\forall y$ `(y >= 0` $\wedge$ `y > 0)` $\implies$ `y - 1 >= 0`
  Provable
- $\forall y$ `((I` $\wedge$ $\neg e$`)` $\implies$ `R)[x` $\leftarrow$ `y]`
  $\forall y$ `(y >= 0` $\wedge$ `y <= 0)` $\implies$ `y = 0`

Partial Correctness of while loop

Example

`wlp(while(x > 0) x--, {x == 0}) == ?`

Candidate I: x >= 0

Satisfies both:

- $\forall y$ ((I $\wedge$ e) $\implies$ wlp(S, I))[x $\leftarrow$ y]
  $\forall y$ (y >= 0 $\wedge$ y > 0) $\implies$ y - 1 >= 0
  Provable
- $\forall y$ ((I $\wedge$ $\neg e$) $\implies$ R)[x $\leftarrow$ y]
  $\forall y$ (y >= 0 $\wedge$ y <= 0) $\implies$ y = 0
  Provable