Logic and Modelling

— Program Logic —

Jörg Endrullis

VU University Amsterdam

Program Logic

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

Note that:

$$\sum_{i=1}^{x} i = 1 + 2 + 3 + \ldots + x$$
$$= \frac{x(x+1)}{2}$$

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

We would like to express, for example:

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

We would like to express, for example:

- started with input $x \geq 0$, `Sum` yields output $\sum_{i=1}^{x} i$ for z

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

We would like to express, for example:

- started with input $x \geq 0$, `Sum` yields output $\sum_{i=1}^{x} i$ for z
- on every input value, `Sum` terminates

Program `Sum`

Computing the sum of the first x natural numbers.

Program `Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

We would like to express, for example:

- started with input $x \geq 0$, `Sum` yields output $\sum_{i=1}^{x} i$ for z
- on every input value, `Sum` terminates

These are **program specifications** (satisfied by `Sum`).

Simple Core Language (Syntax)

Simple Core Language (Syntax)

Arithmetic expressions:

$$E ::= \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B) \mid (E < E)$$

Commands:

$$C ::= x = E \mid C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}$$

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable
- $*$ indicates multiplication

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable
- $*$ indicates multiplication

- 42

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable
- $*$ indicates multiplication

- 42
- y

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable
- $*$ indicates multiplication

- 42
- y
- $4 + (x - 3)$

Simple Core Language (Syntax)

Arithmetic expressions:

$$E \quad ::= \quad \boldsymbol{n} \mid \boldsymbol{x} \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$$

where

- $\boldsymbol{n}$ is any integer numeral in $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\boldsymbol{x}$ any variable
- $*$ indicates multiplication

- 42
- y
- $4 + (x - 3)$
- $x + (x * (y - (5 + z)))$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,||\, B) \mid (E < E)$$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B) \mid (E < E)$$

- `false`

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,||\, B) \mid (E < E)$$

- `false`
- $x < 0$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B) \mid (E < E)$$

- `false`
- $x < 0$
- $(y * y < x)$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B) \mid (E < E)$$

- $\texttt{false}$
- $x < 0$
- $(y * y < x)$
- $(x < 0) \,\|\, (!(x + 1 < 0))$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,\|\, B) \mid (E < E)$$

- `false`
- $x < 0$
- $(y * y < x)$
- $(x < 0) \,\|\, (!(x + 1 < 0))$

Abbreviations

- $E_1 == E_2$ is short for: $!(E_1 < E_2) \,\&\, !(E_2 < E_1)$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,||\, B) \mid (E < E)$$

- `false`
- $x < 0$
- $(y * y < x)$
- $(x < 0) \,||\, (!(x + 1 < 0))$

Abbreviations

- $E_1 == E_2$ is short for: $!(E_1 < E_2) \,\&\, !(E_2 < E_1)$
- $E_1 \; != E_2$ is short for: $!(E_1 == E_2)$

Simple Core Language (Syntax)

Boolean expressions:

$$B ::= \texttt{true} \mid \texttt{false} \mid (!B) \mid (B \,\&\, B) \mid (B \,||\, B) \mid (E < E)$$

- `false`
- $x < 0$
- $(y * y < x)$
- $(x < 0) \,||\, (!(x + 1 < 0))$

Abbreviations

- $E_1 == E_2$ is short for: $!(E_1 < E_2) \,\&\, !(E_2 < E_1)$
- $E_1 \; != E_2$ is short for: $!(E_1 == E_2)$

Binding priorities

same as for logical operators in predicate logic

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

(b) overwrites current value stored in x by that result

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

(b) overwrites current value stored in x by that result

Sequential composition $C_1 ; C_2$:

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

(b) overwrites current value stored in x by that result

Sequential composition $C_1 ; C_2$:

(a) begins with executing C_1

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

(b) overwrites current value stored in x by that result

Sequential composition $C_1 ; C_2$:

(a) begins with executing C_1

(b) if this does not terminate,
then the run of $C_1 ; C_2$ does not terminate

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

Assignment statement $x = E$:

(a) evaluates integer expression E in current state

(b) overwrites current value stored in x by that result

Sequential composition $C_1 ; C_2$:

(a) begins with executing C_1

(b) if this does not terminate,
then the run of $C_1 ; C_2$ does not terminate

(c) otherwise the run continues by executing C_2
in the (storage) state resulting from the execution of C_1

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

While-do statement $\texttt{while}\ B\ \{C\}$:

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

While-do statement $\texttt{while}\ B\ \{C\}$:

(a) boolean expression B is evaluated in the current state

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

While-do statement $\texttt{while}\ B\ \{C\}$:

(a) boolean expression B is evaluated in the current state

(b) if B evaluates to false, then the command terminates

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

While-do statement $\texttt{while}\ B\ \{C\}$:

(a) boolean expression B is evaluated in the current state

(b) if B evaluates to false, then the command terminates

(c) otherwise, command C will be executed.

Simple Core Language (Syntax, Operational Meaning)

Commands:

$$C ::= \underbrace{x = E}_{\text{atomic command}} \mid \underbrace{C;C \mid \texttt{if}\ B\ \{C\}\ \texttt{else}\ \{C\} \mid \texttt{while}\ B\ \{C\}}_{\text{control structures}}$$

If-then-else statement $\texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}$:

(a) evaluates boolean expression B in the current state

(b) if the result is true, then C_1 is executed;

(c) otherwise C_2 is executed.

While-do statement $\texttt{while}\ B\ \{C\}$:

(a) boolean expression B is evaluated in the current state

(b) if B evaluates to false, then the command terminates

(c) otherwise, command C will be executed.

(d) if that execution terminates, then we resume at (a)

Programs `Fac1`, `Fac2`

The **factorial** n of a natural number n is defined inductively by:

$$0! \stackrel{\text{def}}{=} 1$$

$$(n+1)! \stackrel{\text{def}}{=} (n+1) \cdot n!$$

Program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Programs `Fac1`, `Fac2`

The **factorial** n of a natural number n is defined inductively by:

$$0! \stackrel{\text{def}}{=} 1$$

$$(n+1)! \stackrel{\text{def}}{=} (n+1) \cdot n!$$

Program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Program `Fac2`

```
y = 1;
while (x != 0) {
  y = x * y;
  x = x - 1;
}
```

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does `Fac1` always terminate?

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does `Fac1` always terminate? — Not for $x < 0$!

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does `Fac1` always terminate? — Not for $x < 0$!

Therefore `Fac1`:

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does `Fac1` always terminate? — Not for $x < 0$!

Therefore `Fac1`:

- cannot (totally) satisfy the specification (Hoare triple)

$$(\!| \top |\!)\ P\ (\!| y = x! |\!)$$

Hoare Triples (Examples)

Factorial computing program `Fac1`

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does `Fac1` always terminate? — Not for $x < 0$!

Therefore `Fac1`:

- cannot (totally) satisfy the specification (Hoare triple)

$$(\!| \top |\!)\ P\ (\!| y = x! |\!)$$

- but it can satisfy:

$$(\!| x \geq 0 |\!)\ P\ (\!| y = x! |\!)$$

Hoare Triples

A **Hoare triple** is a program specification $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ where:

- ϕ is the **precondition**,
- P a core program,
- ψ is the **postcondition**.

Hoare Triples

A **Hoare triple** is a program specification $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ where:

- ϕ is the **precondition**,
- P a core program,
- ψ is the **postcondition**.

Quantifiers in ϕ and ψ only bind variables not in P !

Hoare Triples

A **Hoare triple** is a program specification $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ where:

- ϕ is the **precondition**,
- P a core program,
- ψ is the **postcondition**.

Quantifiers in ϕ and ψ only bind variables not in P !

A **state** (or **store**) of core programs is a function ℓ that to each variable x assigns an integer $\ell(x)$.

Hoare Triples

A **Hoare triple** is a program specification $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ where:

- ϕ is the **precondition**,
- P a core program,
- ψ is the **postcondition**.

Quantifiers in ϕ and ψ only bind variables not in P !

A **state** (or **store**) of core programs is a function ℓ that to each variable x assigns an integer $\ell(x)$.

The formulas ϕ, ψ are predicate logic formulas over

$$\mathcal{F} = \{\ 0/0,\ 1/0,\ -/2,\ +/2, *\!/2\ \} \qquad \mathcal{P} = \{\ </2,\ =/2\ \}$$

Hoare Triples

A **Hoare triple** is a program specification $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ where:

- ϕ is the **precondition**,
- P a core program,
- ψ is the **postcondition**.

Quantifiers in ϕ and ψ only bind variables not in P !

A **state** (or **store**) of core programs is a function ℓ that to each variable x assigns an integer $\ell(x)$.

The formulas ϕ, ψ are predicate logic formulas over

$$\mathcal{F} = \{\ 0/0,\ 1/0,\ -/2,\ +/2,\ */2\ \} \qquad \mathcal{P} = \{\ </2,\ =/2\ \}$$

we define for all states ℓ :

$$\ell \models \phi \iff \mathcal{Z} \models_\ell \phi$$

where ℓ is viewed as environment, and $\mathcal{Z}$ has domain $\mathbb{Z}$, and interprets function and predicate symbols in standard manner.

Tony Hoare

Tony Hoare (* 1934)

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \models y - x * z < z$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$ since $5 - (-2) \cdot (-1) = 3 \geq -1$.

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$ since $5 - (-2) \cdot (-1) = 3 \geq -1$.
- $\ell \models \forall u\, (y < u \rightarrow y * z < u * z)$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$ since $5 - (-2) \cdot (-1) = 3 \geq -1$.
- $\ell \not\models \forall u\, (y < u \rightarrow y * z < u * z)$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$ since $5 - (-2) \cdot (-1) = 3 \geq -1$.
- $\ell \not\models \forall u\, (y < u \rightarrow y * z < u * z)$
  since: $\ell[u \mapsto 6] \not\models (y < u \rightarrow y * z < u * z)$

Evaluation of Conditions in a State

Example

Let ℓ a state with

$$\ell(x) = -2 \qquad \ell(y) = 5 \qquad \ell(z) = -1$$

Then we find:

- $\ell \models \neg(x + y < z)$ since $(-2) + 5 = 3 \geq (-1)$.
- $\ell \not\models y - x * z < z$ since $5 - (-2) \cdot (-1) = 3 \geq -1$.
- $\ell \not\models \forall u\, (y < u \rightarrow y * z < u * z)$
  since: $\ell[u \mapsto 6] \not\models (y < u \rightarrow y * z < u * z)$
  because: $5 < 6$, but $5 \cdot (-1) = -5 \geq -6 = 6 \cdot (-1)$.

Partial Correctness and Total Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

Partial Correctness and Total Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

Partial Correctness and Total Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

Partial Correctness and Total Correctness

Definition ($\models_{par}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{par}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

Definition ($\models_{tot}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **total correctness** if

Partial Correctness and Total Correctness

Definition ($\models_{par}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{par}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

Definition ($\models_{tot}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

Partial Correctness and Total Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

Definition ($\models_{\text{tot}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

- $\models_{\text{par}}\ (\!|\,x > 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,x = 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,x < 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, x < 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, x < 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, \top \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, x < 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, \top \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

- $\models_{\text{par}}\ (\!|\,x > 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,x = 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\not\models_{\text{par}}\ (\!|\,x < 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\not\models_{\text{par}}\ (\!|\,\top\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,\top\,|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$.

- $\models_{\text{par}}\ (\!|\, x > 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\models_{\text{par}}\ (\!|\, x = 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, x < 0 \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, \top \,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$
- $\not\models_{\text{par}}\ (\!|\, \top \,|\!)\ \texttt{while}\ \texttt{false}\,\{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{par}}\ (\!|x > 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|x = 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|x < 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{false}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

- $\models_{\text{par}}\ (\!|\,x > 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,x = 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\not\models_{\text{par}}\ (\!|\,x < 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\not\models_{\text{par}}\ (\!|\,\top\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\not\models_{\text{par}}\ (\!|\,\top\,|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$
- $\models_{\text{par}}\ (\!|\,\top\,|\!)\ \texttt{while}\,\texttt{true}\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{par}}\ (\!|x > 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|x = 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|x < 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{false}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|\phi|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\psi|\!)$

Partial Correctness

Definition ($\models_{\text{par}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **partial correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- **if** P terminates when started on ℓ,
- **then** the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{par}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{par}}\ (\!|x > 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|x = 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|x < 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{false}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{par}}\ (\!|\phi|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\psi|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\,\phi\,|\!)\ P\ (\!|\,\psi\,|\!)$.

- $\models_{\text{tot}}\ (\!|\,x > 0\,|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|\,x = 0\,|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{true}\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{true}\,\{x = x - 1\}\ (\!|x = 0|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,(x > 0)\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{false}\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\,\texttt{true}\,\{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|\phi|\!)\ \texttt{while}\,\texttt{true}\,\{x = x - 1\}\ (\!|\psi|\!)$

Total Correctness

Definition ($\models_{\text{tot}}$)

The triple $(\!|\phi|\!)\ P\ (\!|\psi|\!)$ is satisfied under **total correctness** if

for all states ℓ that satisfy the pre-condition ϕ:

- P terminates when started on ℓ, and
- the reached state ℓ' satisfies the postcondition ψ.

In this case we write: $\models_{\text{tot}}\ (\!|\phi|\!)\ P\ (\!|\psi|\!)$.

- $\models_{\text{tot}}\ (\!|x > 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\models_{\text{tot}}\ (\!|x = 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|x < 0|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\ (x > 0)\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{false}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$
- $\not\models_{\text{tot}}\ (\!|\phi|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\psi|\!)$

Partial and Total Correctness

A roundabout implementation of the successor function:

Succ

```
a = x + 1;
if (a - 1 == 0) {
    y = 1;
} else {
    y = a;
}
```

Partial and Total Correctness

A roundabout implementation of the successor function:

Succ

```
a = x + 1;
if (a - 1 == 0) {
   y = 1;
} else {
   y = a;
}
```

We should be able to prove:

Partial and Total Correctness

A roundabout implementation of the successor function:

Succ

```
a = x + 1;
if (a - 1 == 0) {
    y = 1;
} else {
    y = a;
}
```

We should be able to prove:

- $\vDash_{\mathsf{par}} (\!| \top |\!)$ Succ $(\!| y = x + 1 |\!)$

Partial and Total Correctness

A roundabout implementation of the successor function:

Succ

```
a = x + 1;
if (a - 1 == 0) {
    y = 1;
} else {
    y = a;
}
```

We should be able to prove:

- $\vDash_{\text{par}} \{\!|\, \top \,|\!\}\ \texttt{Succ}\ \{\!|\, y = x + 1 \,|\!\}$
- $\vDash_{\text{tot}} \{\!|\, \top \,|\!\}\ \texttt{Succ}\ \{\!|\, y = x + 1 \,|\!\}$

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

- $\not\vdash_{\text{tot}} (\!| \top |\!)$ Fac1 $(\!| y = x! |\!)$

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

- $\not\vdash_{\text{tot}} \{\!|\, \top \,|\!\}$ Fac1 $\{\!|\, y = x! \,|\!\}$
- $\vdash_{\text{tot}} \{\!|\, x \geq 0 \,|\!\}$ Fac1 $\{\!|\, y = x! \,|\!\}$

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

- $\not\vdash_{\text{tot}} \{\!|\, \top \,|\!\}$ Fac1 $\{\!|\, y = x! \,|\!\}$
- $\vdash_{\text{tot}} \{\!|\, x \geq 0 \,|\!\}$ Fac1 $\{\!|\, y = x! \,|\!\}$

And for partial correctness we expect:

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

- $\not\vdash_{\text{tot}} \{\!|\top|\!\}$ `Fac1` $\{\!|y = x!|\!\}$
- $\vdash_{\text{tot}} \{\!|x \geq 0|\!\}$ `Fac1` $\{\!|y = x!|\!\}$

And for partial correctness we expect:

- $\vdash_{\text{par}} \{\!|x \geq 0|\!\}$ `Fac1` $\{\!|y = x!|\!\}$

Partial and Total Correctness

Fac1

```
y = 1;
z = 0;
while (z != x) {
  z = z + 1;
  y = y * z;
}
```

Does not terminate for $x < 0$!

Therefore for total correctness, we expect to be able to prove:

- $\not\vdash_{\text{tot}} \langle\!| \top |\!\rangle$ Fac1 $\langle\!| y = x! |\!\rangle$
- $\vdash_{\text{tot}} \langle\!| x \geq 0 |\!\rangle$ Fac1 $\langle\!| y = x! |\!\rangle$

And for partial correctness we expect:

- $\vdash_{\text{par}} \langle\!| x \geq 0 |\!\rangle$ Fac1 $\langle\!| y = x! |\!\rangle$
- $\vdash_{\text{par}} \langle\!| \top |\!\rangle$ Fac1 $\langle\!| y = x! |\!\rangle$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

- $\vDash_{\text{par}} \langle\!| x \geq 0 |\!\rangle$ `Fac2` $\langle\!| y = x! |\!\rangle$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that `Fac2` 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

- $\vDash_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that `Fac2` 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although `Fac2` computes the factorial function, we have:

- $\vDash_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that `Fac2` 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although `Fac2` computes the factorial function, we have:

- $\not\vDash_{\text{par}} \{\!|\, x = n \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$ (for $n = 2, 3, \ldots$).
- $\vDash_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that `Fac2` 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although `Fac2` computes the factorial function, we have:

- $\not\vDash_{\text{par}} \{\!|\, x = n \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$ (for $n = 2, 3, \ldots$).
- $\not\vDash_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that `Fac2` 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although `Fac2` computes the factorial function, we have:

- $\not\models_{\text{par}} \{\!|\, x = n \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$ (for $n = 2, 3, \ldots$).
- $\not\models_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ `Fac2` $\{\!|\, y = x! \,|\!\}$

Remedy: use of a logical variable x_0 .

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that Fac2 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although Fac2 computes the factorial function, we have:

- $\not\vDash_{\text{par}} \{\!|\, x = n \,|\!\}$ Fac2 $\{\!|\, y = x! \,|\!\}$ (for $n = 2, 3, \ldots$).
- $\not\vDash_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ Fac2 $\{\!|\, y = x! \,|\!\}$

Remedy: use of a logical variable x_0. Then we expect:

- $\vDash_{\text{par}} \{\!|\, (x \geq 0) \wedge x_0 = x \,|\!\}$ Fac2 $\{\!|\, y = x_0! \,|\!\}$

Program Variables and Logical Variables

Fac2

```
y = 1;
while (x != 0) {
  y = y * x;
  x = x - 1;
}
```

Note that Fac2 'consumes' its input:

- If $x > 0$ before execution, we have $x = 0$ afterwards.

Although Fac2 computes the factorial function, we have:

- $\not\models_{\text{par}} \{\!|\, x = n \,|\!\}$ Fac2 $\{\!|\, y = x! \,|\!\}$ (for $n = 2, 3, \ldots$).
- $\not\models_{\text{par}} \{\!|\, x \geq 0 \,|\!\}$ Fac2 $\{\!|\, y = x! \,|\!\}$

Remedy: use of a logical variable x_0. Then we expect:

- $\models_{\text{par}} \{\!|\, (x \geq 0) \wedge x_0 = x \,|\!\}$ Fac2 $\{\!|\, y = x_0! \,|\!\}$
- $\models_{\text{tot}} \{\!|\, (x \geq 0) \wedge x_0 = x \,|\!\}$ Fac2 $\{\!|\, y = x_0! \,|\!\}$

Program Variables and Logical Variables

`Sum` adds up the first x natural numbers and stores them in z :

`Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

Program Variables and Logical Variables

`Sum` adds up the first x natural numbers and stores them in z :

`Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

During execution x is consumed.

Program Variables and Logical Variables

Sum adds up the first x natural numbers and stores them in z :

Sum

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

During execution x is consumed. So while we have e.g.:

- $\vDash_{\text{tot}} \{\!|\, x = 2 \,|\!\}$ Sum $\{\!|\, z = 3 \,|\!\}$,

Program Variables and Logical Variables

`Sum` adds up the first x natural numbers and stores them in z:

`Sum`

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

During execution x is consumed. So while we have e.g.:

- $\vDash_{\text{tot}} \{\!|\, x = 2 \,|\!\}$ `Sum` $\{\!|\, z = 3 \,|\!\}$,
- $\vDash_{\text{tot}} \{\!|\, x = 3 \,|\!\}$ `Sum` $\{\!|\, z = 6 \,|\!\}$,

Program Variables and Logical Variables

Sum adds up the first x natural numbers and stores them in z:

Sum

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

During execution x is consumed. So while we have e.g.:

- $\vDash_{\text{tot}} \{\!|\, x = 2 \,|\!\}$ Sum $\{\!|\, z = 3 \,|\!\}$,
- $\vDash_{\text{tot}} \{\!|\, x = 3 \,|\!\}$ Sum $\{\!|\, z = 6 \,|\!\}$,

we find:

- $\not\vDash_{\text{tot}} \{\!|\, x \geq 0 \,|\!\}$ Sum $\{\!|\, z = \frac{x(x+1)}{2} \,|\!\}$, (note: $\sum_{i=1}^{x} i = \frac{x(x+1)}{2}$).

Program Variables and Logical Variables

Sum adds up the first x natural numbers and stores them in z:

Sum

```
z = 0;
while (x > 0) {
  z = z + x;
  x = x - 1;
}
```

During execution x is consumed. So while we have e.g.:

- $\vDash_{\text{tot}} \{\!|\, x = 2 \,|\!\}$ Sum $\{\!|\, z = 3 \,|\!\}$,
- $\vDash_{\text{tot}} \{\!|\, x = 3 \,|\!\}$ Sum $\{\!|\, z = 6 \,|\!\}$,

we find:

- $\not\vDash_{\text{tot}} \{\!|\, x \geq 0 \,|\!\}$ Sum $\{\!|\, z = \frac{x(x+1)}{2} \,|\!\}$, (note: $\sum_{i=1}^{x} i = \frac{x(x+1)}{2}$).

Again, we 'store' the initial value of x in a logical variable x_0:

- $\vDash_{\text{tot}} \{\!|\, x \geq 0 \wedge x_0 = x \,|\!\}$ Sum $\{\!|\, z = \frac{x_0(x_0+1)}{2} \,|\!\}$

Program Variables and Logical Variables

Definition

For a Hoare triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$, its set of **logical variables** are those variables that are free in ϕ and ψ, but do not occur in P.

Program Variables and Logical Variables

Definition

For a Hoare triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$, its set of **logical variables** are those variables that are free in ϕ and ψ, but do not occur in P.

Examples

- $\vDash_{\text{tot}}\ (\!|\, (x \geq 0) \wedge x_0 = x \,|\!)\ \texttt{Fac2}(x, y)\ (\!|\, y = x_0! \,|\!)$

Program Variables and Logical Variables

Definition

For a Hoare triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$, its set of **logical variables** are those variables that are free in ϕ and ψ, but do not occur in P.

Examples

- $\vDash_{\text{tot}}\ (\!|\, (x \geq 0) \wedge x_0 = x \,|\!)\ \texttt{Fac2}(x, y)\ (\!|\, y = x_0! \,|\!)$
- $\vDash_{\text{tot}}\ (\!|\, x \geq 0 \wedge x' = x \,|\!)\ \texttt{Sum}(x, z)\ (\!|\, z = \frac{x'(x'+1)}{2} \,|\!)$

Program Variables and Logical Variables

Definition

For a Hoare triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$, its set of **logical variables** are those variables that are free in ϕ and ψ, but do not occur in P.

Examples

- $\models_{\text{tot}}\ (\!|\, (x \geq 0) \wedge x_0 = x \,|\!)\ \texttt{Fac2}(x, y)\ (\!|\, y = x_0! \,|\!)$
- $\models_{\text{tot}}\ (\!|\, x \geq 0 \wedge x' = x \,|\!)\ \texttt{Sum}(x, z)\ (\!|\, z = \frac{x'(x'+1)}{2} \,|\!)$

- called **logical**: because occur only in logical formulas

Program Variables and Logical Variables

Definition

For a Hoare triple $(\!|\, \phi \,|\!)\ P\ (\!|\, \psi \,|\!)$, its set of **logical variables** are those variables that are free in ϕ and ψ, but do not occur in P.

Examples

- $\models_{\text{tot}}\ (\!|\, (x \geq 0) \wedge x_0 = x \,|\!)\ \texttt{Fac2}(x, y)\ (\!|\, y = x_0! \,|\!)$
- $\models_{\text{tot}}\ (\!|\, x \geq 0 \wedge x' = x \,|\!)\ \texttt{Sum}(x, z)\ (\!|\, z = \frac{x'(x'+1)}{2} \,|\!)$

- called **logical**: because occur only in logical formulas
- state of program assigns a value to program variables, but not to logical variables

Proof Rules for Partial Correctness

$$\frac{(\!|\phi|\!)\ C_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ C_1;C_2\ (\!|\psi|\!)}\ \text{Composition}$$

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

Proof Rules for Partial Correctness

$$\frac{(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)}{(\!|\,\phi\,|\!)\ C_1 ; C_2\ (\!|\,\psi\,|\!)}\ \text{Composition}$$

Proof Rules for Partial Correctness

$$\frac{(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)}{(\!|\,\phi\,|\!)\ C_1 ; C_2\ (\!|\,\psi\,|\!)}\ \text{Composition}$$

Convenient to read bottom up (also for the other rules):

Proof Rules for Partial Correctness

$$\frac{(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)}{(\!|\,\phi\,|\!)\ C_1 ; C_2\ (\!|\,\psi\,|\!)}\ \text{Composition}$$

Convenient to read bottom up (also for the other rules):

In order to prove

$$(\!|\,\phi\,|\!)\ C_1 ; C_2\ (\!|\,\psi\,|\!)$$

Proof Rules for Partial Correctness

$$\frac{(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)}{(\!|\,\phi\,|\!)\ C_1;C_2\ (\!|\,\psi\,|\!)}\ \text{Composition}$$

Convenient to read bottom up (also for the other rules):

In order to prove

$$(\!|\,\phi\,|\!)\ C_1;C_2\ (\!|\,\psi\,|\!)$$

we need to find an **appropriate midcondition** η, and prove

$$(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad \text{and} \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)$$

Proof Rules for Partial Correctness

$$\frac{(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)}{(\!|\,\phi\,|\!)\ C_1;C_2\ (\!|\,\psi\,|\!)}\ \text{Composition}$$

Convenient to read bottom up (also for the other rules):

In order to prove

$$(\!|\,\phi\,|\!)\ C_1;C_2\ (\!|\,\psi\,|\!)$$

we need to find an **appropriate midcondition** η, and prove

$$(\!|\,\phi\,|\!)\ C_1\ (\!|\,\eta\,|\!) \qquad \text{and} \qquad (\!|\,\eta\,|\!)\ C_2\ (\!|\,\psi\,|\!)$$

We thereby split the goal into two subgoals:

- execution of C_1 in a state satisfying ϕ, we get to a state satisfying η
- execution of C_2 in a state satisfying η, we get to a state satisfying ψ

Proof Rules for Partial Correctness

$$\frac{\{\!|\, \phi \wedge B \,|\!\}\ C_1\ \{\!|\, \psi \,|\!\} \qquad \{\!|\, \phi \wedge \neg B \,|\!\}\ C_2\ \{\!|\, \psi \,|\!\}}{\{\!|\, \phi \,|\!\}\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ \{\!|\, \psi \,|\!\}}\ \text{If-statement}$$

Proof Rules for Partial Correctness

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

The goal (conclusion) is decomposed into two subgoals:

Proof Rules for Partial Correctness

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

The goal (conclusion) is decomposed into two subgoals:

- if B evaluates to true, the then-part C_1 is executed

Proof Rules for Partial Correctness

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

The goal (conclusion) is decomposed into two subgoals:

- if B evaluates to true, the then-part C_1 is executed
- if B evaluates to false, the else-part C_2 is executed

Proof Rules for Partial Correctness

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

The goal (conclusion) is decomposed into two subgoals:

- if B evaluates to true, the then-part C_1 is executed
- if B evaluates to false, the else-part C_2 is executed

In both cases, ψ has to hold after the execution.

Proof Rules for Partial Correctness

$$\frac{}{(\!|\, \psi[E/x] \,|\!)\ x = E\ (\!|\, \psi \,|\!)}\ \text{Assignment}$$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\, \psi[E/x] \,|\!)\ x = E\ (\!|\, \psi \,|\!)}\ \text{Assignment}$$

For ψ is $x = 5$ we get:

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

For ψ is $x = 5$ we get:

$$(\!|6 = 5|\!)\ x = 6\ (\!|x = 5|\!)$$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

For ψ is $x = 5$ we get:

$$(\!|6 = 5|\!)\ x = 6\ (\!|x = 5|\!)$$

For ψ' is $x = 6$ we get:

$$(\!|6 = 6|\!)\ x = 6\ (\!|x = 6|\!)$$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

For ψ is $x = 5$ we get:

$$(\!|6 = 5|\!)\ x = 6\ (\!|x = 5|\!)$$

For ψ' is $x = 6$ we get:

$$(\!|6 = 6|\!)\ x = 6\ (\!|x = 6|\!)$$

The reverse form with conclusion

$$(\!|\psi|\!)\ x = E\ (\!|\psi[E/x]|\!)$$

does not make sense!

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

For ψ is $x = 5$ we get:

$$(\!|6 = 5|\!)\ x = 6\ (\!|x = 5|\!)$$

For ψ' is $x = 6$ we get:

$$(\!|6 = 6|\!)\ x = 6\ (\!|x = 6|\!)$$

The reverse form with conclusion

$$(\!|\psi|\!)\ x = E\ (\!|\psi[E/x]|\!)$$

does not make sense! E.g. for ϕ is $x = 5$ it would entail:

$$(\!|x = 5|\!)\ x = 6\ (\!|6 = 5|\!) \quad \times$$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\, \psi[E/x] \,|\!)\ x = E\ (\!|\, \psi \,|\!)}\ \text{Assignment}$$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\, \psi[E/x] \,|\!)\ x = E\ (\!|\, \psi \,|\!)}\ \text{Assignment}$$

- is best applied backwards

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $(\!|2 = 4|\!)\ x = 2\ (\!|x = 4|\!)$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $(\!|2 = 4|\!)\ x = 2\ (\!|x = 4|\!)$
- $(\!|2 > 2|\!)\ x = 2\ (\!|2 > x|\!)$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $(\!|2 = 4|\!)\ x = 2\ (\!|x = 4|\!)$
- $(\!|2 > 2|\!)\ x = 2\ (\!|2 > x|\!)$
- $(\!|x + 1 = y|\!)\ x = x + 1\ (\!|x = y|\!)$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $(\!|2 = 4|\!)\ x = 2\ (\!|x = 4|\!)$
- $(\!|2 > 2|\!)\ x = 2\ (\!|2 > x|\!)$
- $(\!|x + 1 = y|\!)\ x = x + 1\ (\!|x = y|\!)$
- $(\!|x + 1 + 5 = y|\!)\ x = x + 1\ (\!|x + 5 = y|\!)$

Proof Rules for Partial Correctness

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

- is best applied backwards
- note the necessity of capture avoiding in substitution!

Hint for understanding: if ϕ holds when replacing x by E, then after the assignment $x = E$, the formula ϕ holds.

- $(\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $(\!|2 = 4|\!)\ x = 2\ (\!|x = 4|\!)$
- $(\!|2 > 2|\!)\ x = 2\ (\!|2 > x|\!)$
- $(\!|x + 1 = y|\!)\ x = x + 1\ (\!|x = y|\!)$
- $(\!|x + 1 + 5 = y|\!)\ x = x + 1\ (\!|x + 5 = y|\!)$
- $(\!|x + 1 > 0 \wedge y > 0|\!)\ x = x + 1\ (\!|x > 0 \wedge y > 0|\!)$

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

- the body C of the loop usually changes the variables

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

- the body C of the loop usually changes the variables
- ψ expresses a relationship between values that is preserved by any execution of C

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

- the body C of the loop usually changes the variables
- ψ expresses a relationship between values that is preserved by any execution of C
- the **premise** expresses:
  - if ψ and B are true before executing the body C, then ψ will again be true afterwards

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

- the body C of the loop usually changes the variables
- ψ expresses a relationship between values that is preserved by any execution of C
- the **premise** expresses:
  - if ψ and B are true before executing the body C, then ψ will again be true afterwards
- the **conclusion** expresses:
  - if ψ is true before the execution, then no matter how often the while-loop is executed, ψ will also be true at the end

Proof Rules for Partial Correctness

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

The key part is the **loop invariant** ψ:

- the body C of the loop usually changes the variables
- ψ expresses a relationship between values that is preserved by any execution of C
- the **premise** expresses:
  - if ψ and B are true before executing the body C, then ψ will again be true afterwards
- the **conclusion** expresses:
  - if ψ is true before the execution, then no matter how often the while-loop is executed, ψ will also be true at the end
  - and since the while-statement only terminates if B does not hold any more, B will be false in the final state

Proof Rules for Partial Correctness

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!| \phi |\!)\ C\ (\!| \psi |\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!| \phi' |\!)\ C\ (\!| \psi' |\!)}\ \text{Implied}$$

Proof Rules for Partial Correctness

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\, \phi \,|\!)\ C\ (\!|\, \psi \,|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\, \phi' \,|\!)\ C\ (\!|\, \psi' \,|\!)}\ \text{Implied}$$

$\vdash_{AR} \phi$ is valid $\iff$

there is a natural deduction proof of ϕ where standard laws of arithmetic may be used as premises

Proof Rules for Partial Correctness

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\, \phi \,|\!)\ C\ (\!|\, \psi \,|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\, \phi' \,|\!)\ C\ (\!|\, \psi' \,|\!)}\ \text{Implied}$$

$\vdash_{AR} \phi$ is valid $\iff$

there is a natural deduction proof of ϕ where standard laws of arithmetic may be used as premises

When **applied top-down**, this rule **allows** to:

- **strengthen** the precondition (assume more than we need)
- **weaken** the postcondition (conclude less than we could)

Proof Rules for Partial Correctness

$$\frac{(\!|\phi|\!)\ C_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ C_1;C_2\ (\!|\psi|\!)}\ \text{Composition}$$

$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

$$\frac{(\!|\phi \wedge B|\!)\ C_1\ (\!|\psi|\!) \qquad (\!|\phi \wedge \neg B|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ \texttt{if}\ B\ \{C_1\}\ \texttt{else}\ \{C_2\}\ (\!|\psi|\!)}\ \text{If-statement}$$

$$\frac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$\vDash_{\mathsf{par}} \left(\!\left| \top \right|\!\right) \texttt{while}\, \texttt{true}\, \{x = x - 1\} \left(\!\left| x = 0 \right|\!\right)$

$\vDash_{\mathsf{par}} \left(\!\left| \top \right|\!\right) \texttt{while}\, \texttt{true}\, \{x = x - 1\} \left(\!\left| x = 0 \right|\!\right)$

$$\left(\!\left| \top \right|\!\right) \texttt{while}\, \texttt{true}\, \{x = x - 1\} \left(\!\left| x = 0 \right|\!\right)$$

$\vDash_{\mathsf{par}} \left(\!\left| \top \right|\!\right) \texttt{while}\ \texttt{true}\ \{x = x - 1\} \left(\!\left| x = 0 \right|\!\right)$

$$\frac{}{\left(\!\left| \top \right|\!\right) \texttt{while}\ \texttt{true}\ \{x = x - 1\} \left(\!\left| x = 0 \right|\!\right)}\ \text{Implied}$$

$\vDash_{\text{par}} \ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\frac{}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$\vDash_{\text{par}} \ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\frac{\vdash_{AR} \top \wedge \neg\top \rightarrow x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

$$\frac{\vdash_{AR} \phi' \rightarrow \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \rightarrow \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$\vDash_{\text{par}} \ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)} \qquad \vdash_{AR} \top \wedge \neg\top \rightarrow x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

$$\dfrac{\vdash_{AR} \phi' \rightarrow \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \rightarrow \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$\vDash_{\text{par}} \langle\!| \top |\!\rangle \ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!| x = 0 |\!\rangle$

$$\dfrac{\dfrac{}{\langle\!| \top |\!\rangle \ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!| \top \wedge \neg \top |\!\rangle} \qquad \vdash_{AR} \top \wedge \neg \top \rightarrow x = 0}{\langle\!| \top |\!\rangle \ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!| x = 0 |\!\rangle}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \rightarrow \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \rightarrow \phi \qquad \langle\!| \phi |\!\rangle \ C \ \langle\!| \psi |\!\rangle \qquad \vdash_{AR} \psi \rightarrow \psi'}{\langle\!| \phi' |\!\rangle \ C \ \langle\!| \psi' |\!\rangle}\ \text{Implied}$$

$\vDash_{\text{par}} \; (\!|\, \top \,|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)$

$$\dfrac{\dfrac{}{(\!|\, \top \,|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\, \top \wedge \neg \top \,|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg \top \rightarrow x = 0}{(\!|\, \top \,|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\, x = 0 \,|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \rightarrow \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \rightarrow \phi \qquad (\!|\, \phi \,|\!)\ C\ (\!|\, \psi \,|\!) \qquad \vdash_{AR} \psi \rightarrow \psi'}{(\!|\, \phi' \,|\!)\ C\ (\!|\, \psi' \,|\!)}\ \text{Implied}$$

$$\dfrac{\dfrac{}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \; (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \rightarrow x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \rightarrow \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \rightarrow \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \rightarrow \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \; (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Implied}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \; (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Implied}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \langle\!|\, \top \,|\!\rangle \ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!|\, x = 0 \,|\!\rangle$

$$\dfrac{\dfrac{\dfrac{\vdash_{AR} \top \wedge \top \to \top}{\langle\!|\, \top \wedge \top \,|\!\rangle\ x = x - 1\ \langle\!|\, \top \,|\!\rangle}\ \text{Implied}}{\langle\!|\, \top \,|\!\rangle\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!|\, \top \wedge \neg \top \,|\!\rangle}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg \top \to x = 0}{\langle\!|\, \top \,|\!\rangle\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ \langle\!|\, x = 0 \,|\!\rangle}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad \langle\!|\, \phi \,|\!\rangle\ C\ \langle\!|\, \psi \,|\!\rangle \qquad \vdash_{AR} \psi \to \psi'}{\langle\!|\, \phi' \,|\!\rangle\ C\ \langle\!|\, \psi' \,|\!\rangle}\ \text{Implied}$$

$$\dfrac{\langle\!|\, \psi \wedge B \,|\!\rangle\ C\ \langle\!|\, \psi \,|\!\rangle}{\langle\!|\, \psi \,|\!\rangle\ \texttt{while}\ B\ \{C\}\ \langle\!|\, \psi \wedge \neg B \,|\!\rangle}\ \text{Partial-while}$$

$\vDash_{\text{par}} \ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{\vdash_{AR} \top \wedge \top \to \top \qquad \overline{(\!|\top|\!)\ x = x - 1\ (\!|\top|\!)}}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Implied}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \ (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{\vdash_{AR} \top \wedge \top \to \top \qquad \overline{(\!|\top|\!)\ x = x - 1\ (\!|\top|\!)}}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Implied}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$\vDash_{\text{par}} \; (\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)$

$$\dfrac{\dfrac{\dfrac{\vdash_{AR} \top \wedge \top \to \top \qquad \dfrac{}{(\!|\top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Assign}}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\ \text{Implied}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\ \text{Partial-while} \qquad \vdash_{AR} \top \wedge \neg\top \to x = 0}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|x = 0|\!)}\ \text{Implied}$$

Note: We ignored trivial premises $\vdash_{AR} \chi \to \chi$ of Implied

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\dfrac{(\!|\psi \wedge B|\!)\ C\ (\!|\psi|\!)}{(\!|\psi|\!)\ \texttt{while}\ B\ \{C\}\ (\!|\psi \wedge \neg B|\!)}\ \text{Partial-while}$$

$$\dfrac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}\ \text{Assignment}$$

$\vDash_{\text{par}} \; (\!|\phi|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\psi|\!)$

$$\dfrac{\vdash_{AR} \phi \to \top \qquad \dfrac{\dfrac{\vdash_{AR} \top \wedge \top \to \top \qquad \dfrac{}{(\!|\top|\!)\ x = x - 1\ (\!|\top|\!)}\,\text{a}}{(\!|\top \wedge \top|\!)\ x = x - 1\ (\!|\top|\!)}\,\text{i}}{(\!|\top|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\top \wedge \neg\top|\!)}\,\text{w} \qquad \vdash_{AR} \top \wedge \neg\top \to \psi}{(\!|\phi|\!)\ \texttt{while}\ \texttt{true}\ \{x = x - 1\}\ (\!|\psi|\!)}\,\text{i}$$

$$\dfrac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

Towards Proving Partial Correctness of `Fac2`

$$\frac{}{(\!|\psi[E/x]|\!)\ x := E\ (\!|\psi|\!)}\ \text{Assignment}$$

$$\frac{\vdash_{AR} \phi' \to \phi \qquad (\!|\phi|\!)\ C\ (\!|\psi|\!) \qquad \vdash_{AR} \psi \to \psi'}{(\!|\phi'|\!)\ C\ (\!|\psi'|\!)}\ \text{Implied}$$

$$\frac{(\!|\phi|\!)\ C_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ C_2\ (\!|\psi|\!)}{(\!|\phi|\!)\ C_1; C_2\ (\!|\psi|\!)}\ \text{Composition}$$

$$\dfrac{\dfrac{\dfrac{}{(\!|1=1|\!)\ y := 1\ (\!|y=1|\!)}\,a}{(\!|\top|\!)\ y := 1\ (\!|y=1|\!)}\,i \qquad \dfrac{\dfrac{}{(\!|y=1 \wedge 0=0|\!)\ z := 0\ (\!|y=1 \wedge z=0|\!)}\,a}{(\!|y=1|\!)\ z := 0\ (\!|y=1 \wedge z=0|\!)}\,i}{(\!|\top|\!)\ y := 1; z := 0\ (\!|y=1 \wedge z=0|\!)}\,c$$

Proving Partial Correctness of `Fac2`

$$\dfrac{\dots \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\{\!|y \cdot (z+1) = (z+1)!|\!\}\ z = z+1\ \{\!|y \cdot z = z!|\!\}}\,a}{\{\!|y = z! \wedge z \neq x|\!\}\ z = z+1\ \{\!|y \cdot z = z!|\!\}}\,i \quad \dfrac{}{\{\!|y \cdot z = z!|\!\}\ y = y * z\ \{\!|y = z!|\!\}}\,a}{\{\!|y = z! \wedge z \neq x|\!\}\ z = z+1; y = y * z\ \{\!|y = z!|\!\}}\,c}{\{\!|y = z!|\!\}\ \texttt{while}\ (z\ \texttt{!=}\ x)\ \{z = z+1; y = y * z\}\ \{\!|y = z! \wedge z = x|\!\}}\,w}{\{\!|y = 1 \wedge z = 0|\!\}\ \texttt{while}\ (z\ \texttt{!=}\ x)\ \{z = z+1; y = y * z\}\ \{\!|y = x!|\!\}}\,i}{\{\!|\top|\!\}\ \underbrace{y = 1; z = 0; \texttt{while}\ (z\ \texttt{!=}\ x)\ \{z = z+1; y = y * z\}}_{\texttt{Fac2}}\ \{\!|y = x!|\!\}}\,c$$

$$\dfrac{\{\!|\psi \wedge B|\!\}\ C\ \{\!|\psi|\!\}}{\{\!|\psi|\!\}\ \texttt{while}\ B\ \{C\}\ \{\!|\psi \wedge \neg B|\!\}}\ \text{Partial-while}$$

(loop invariant ψ)