COMP4161
Advanced Topics in Software Verification

INV & Exam Prep

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

COMP4161
Advanced Topics in Software Verification

INV

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

Practice with invariants!

Practice with invariants!

Recall:

- ➜ invariants are needed to automate the application of hoare rules
- ➜ they are used by the weakest precondition calculus to deal with loops

Practice with invariants!

Recall:

- ➜ invariants are needed to automate the application of hoare rules
- ➜ they are used by the weakest precondition calculus to deal with loops

Recall:

- ➜ an invariant needs to be "enough" (to prove the postcondition)
- ➜ an invariant needs to be an invariant

Practice with invariants!

Recall:

- ➜ invariants are needed to automate the application of hoare rules
- ➜ they are used by the weakest precondition calculus to deal with loops

Recall:

- ➜ an invariant needs to be "enough" (to prove the postcondition)
- ➜ an invariant needs to be an invariant
  - ➜ "true before the loop"
  - ➜ "if true at the start of an iteration, still true after one iteration"

Weakest precondition - recall

$$\{ \; P \; \} \quad i_0; \; i_1; \; i_2; \quad \{ \; Q \; \}$$

Weakest precondition - recall

$$(P \Longrightarrow pre\ (i_0;\ i_1;\ i_2;)\ Q) \Longrightarrow \{\ P\ \}\ \ i_0;\ i_1;\ i_2;\ \ \{\ Q\ \}$$

Weakest precondition - recall

$(P \Longrightarrow pre\ (i_0;\ i_1;\ i_2;)\ Q) \Longrightarrow\ \{\ P\ \}\ \ i_0;\ i_1;\ i_2;\ \ \{\ Q\ \}$

$\{\ P\ \}$

$i_0;$

$i_1;$

$i_2;$

$\{\ Q\ \}$

Weakest precondition - recall

$(P \Longrightarrow pre\ (i_0;\ i_1;\ i_2;)\ Q) \Longrightarrow \{\ P\ \}\ \ i_0;\ i_1;\ i_2;\ \ \{\ Q\ \}$

$\{\ P\ \}$

$i_0;$

$i_1;$

$pre\ \ i_2\ \ Q$

$i_2;$

$\{\ Q\ \}$

Weakest precondition - recall

$(P \Longrightarrow pre\ (i_0;\ i_1;\ i_2;)\ Q) \Longrightarrow \{\ P\ \}\ \ i_0;\ i_1;\ i_2;\ \ \{\ Q\ \}$

$\{\ P\ \}$

$i_0;$

$pre\ \ i_1\ \ (pre\ \ i_2\ \ Q)$

$i_1;$

$pre\ \ i_2\ \ Q$

$i_2;$

$\{\ Q\ \}$

Weakest precondition - recall

$(P \Longrightarrow pre\ (i_0;\ i_1;\ i_2;)\ Q) \Longrightarrow\ \{\ P\ \}\ \ i_0;\ i_1;\ i_2;\ \ \{\ Q\ \}$

$\{\ P\ \}$

$pre\ i_0\ (pre\ i_1\ (pre\ i_2\ Q)) = pre\ i_1; i_2; i_3;\ Q$

$i_0;$

$pre\ i_1\ (pre\ i_2\ Q)$

$i_1;$

$pre\ i_2\ Q$

$i_2;$

$\{\ Q\ \}$

Invariant - recall

$\{ P \}$

$WHILE\ b$

DO

c

OD

$\{ Q \}$

Invariant - recall

$\{\ P\ \}$

??

WHILE b

DO

c

OD

$\{\ Q\ \}$

Invariant - recall

$\{ P \}$

$WHILE\ b\ INV\ I$

DO

c

OD

$\{ Q \}$

?? $pre\ (WHILE\ b\ INV\ I\ DO\ c\ OD) = I$

Invariant - recall

$\{\ P\ \}$

$P \implies I$ ("true before the loop")

?? $pre\ (WHILE\ b\ INV\ I\ DO\ c\ OD) = I$

WHILE b INV I

DO

c

OD

$\{\ Q\ \}$

Invariant - recall

$\{ P \}$

$P \implies I$ ("true before the loop")

?? $pre\ (WHILE\ b\ INV\ I\ DO\ c\ OD) = I$

$WHILE\ b\ INV\ I$ $I \wedge b \implies pre\ c\ I$

("if true at the start of an iteration,")

DO ("still true after one iteration")

c

OD

$\{ Q \}$

Invariant - recall

$\{ P \}$ | $P \implies I$ ("true before the loop")

?? $pre\ (WHILE\ b\ INV\ I\ DO\ c\ OD) = I$

$WHILE\ b\ INV\ I$ | $I \wedge b \implies pre\ c\ I$

("if true at the start of an iteration,")

DO | ("still true after one iteration")

c

OD

$I \wedge \neg b \implies Q$ ("enough")

$\{ Q \}$

Example 1

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$

WHILE $A \neq a$
DO
    $B := B + b;$
    $A := A + 1$
OD

Example 1

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$

$A =$	0	1	2	3	4	...
$B =$	0	b	b+b	b+b+b	b+b+b+b	...

WHILE $A \neq a$
DO
  $B := B + b;$
  $A := A + 1$
OD

Example 1

$\{\ a \geq 0\ \land\ b \geq 0\ \}$
$A := 0;$
$B := 0;$

$A =$	0	1	2	3	4	...
$B =$	0	b	b+b	b+b+b	b+b+b+b	...

WHILE $A \neq a$
DO
$B := B + b;$
$A := A + 1$
OD
$\{\ B = b * a\ \}$

Example 1

$\{ \ a \geq 0 \ \land \ b \geq 0 \ \}$
$A := 0;$
$B := 0;$
INV { B = b * A}
WHILE $A \neq a$
DO
  $B := B + b;$
  $A := A + 1$
OD
$\{ \ B = b * a \ \}$

$A =$	0	1	2	3	4	...
$B =$	0	b	b+b	b+b+b	b+b+b+b	...

Example 1

$\{ \ a \geq 0 \ \wedge \ b \geq 0 \ \}$
$A := 0;$
$B := 0;$
INV { B = b * A}
WHILE $A \neq a$
DO
  $B := B + b;$
  $A := A + 1$
OD
$\{ \ B = b * a \ \}$

Example 1

$\{ \ a \geq 0 \ \wedge \ b \geq 0 \ \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0 \quad \checkmark$
INV { B = b * A}
WHILE $A \neq a$
DO
  $B := B + b;$
  $A := A + 1$
OD
$\{ \ B = b * a \ \}$

Example 1

$\{\ a \geq 0 \ \wedge \ b \geq 0\ \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0 \quad ✓$
INV { B = b * A}
WHILE $A \neq a$ $\quad B = b * A \wedge A \neq a \longrightarrow B + b = b * (A + 1)$
DO
  $B := B + b;$
  $A := A + 1$
OD
$\{\ B = b * a\ \}$

Example 1

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0 \quad \checkmark$
INV { B = b * A}
WHILE $A \neq a$ $\quad B = b * A \wedge A \neq a \longrightarrow B + b = b * (A + 1)$
DO $\quad = b * A + b$
$\quad B := B + b;$ $\quad = B + b \quad \checkmark$
$\quad A := A + 1$
OD
$\{\ B = b * a\ \}$

Example 1

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0 \quad \checkmark$
INV { B = b * A}
WHILE $A \neq a$ $\quad B = b * A \wedge A \neq a \longrightarrow B + b = b * (A + 1)$
DO $\quad = b * A + b$
$\quad B := B + b;$ $\quad = B + b \quad \checkmark$
$\quad A := A + 1$
OD $\quad B = b * A \wedge A = a \longrightarrow B = b * a \quad \checkmark$
$\{\ B = b * a\ \}$

Example 2

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$

WHILE $A < a$
DO
    $B := B + b;$
    $A := A + 1$
OD

Example 2

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$

WHILE $A < a$
DO
$B := B + b;$
$A := A + 1$
OD
$\{\ B = b * a\ \}$

Example 2

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$
INV { B = b * A}
WHILE $A < a$
DO
    $B := B + b;$
    $A := A + 1$
OD
$\{\ B = b * a\ \}$

Example 2

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0 \quad \checkmark$
INV { B = b * A}
WHILE $A < a$ $\quad B = b * A \wedge A < a \longrightarrow B + b = b * (A + 1)$
DO $\quad = b * A + b$
$\quad B := B + b;$ $\quad = B + b \quad \checkmark$
$\quad A := A + 1$
OD $\quad B = b * A \wedge A \geq a \longrightarrow B = b * a \quad$???
$\{\ B = b * a\ \}$

Example 2

$\{ a \geq 0 \ \wedge \ b \geq 0 \}$
$A := 0;$
$B := 0;$ $\quad 0 = b * 0$
INV { B = b * A ∧ A≤ a}
WHILE $A < a$ $\quad B = b * A \wedge A < a \longrightarrow B + b = b * (A + 1)$
DO
$\quad B := B + b;$
$\quad A := A + 1$
OD $\quad B = b * A \wedge A \geq a \longrightarrow B = b * a$
$\{ B = b * a \}$

Example 2

$\{\ a \geq 0\ \wedge\ b \geq 0\ \}$
$A := 0;$
$B := 0;$ $\qquad 0 = b * 0 \wedge 0 \leq a$
INV { B = b * A ∧ A≤ a}
WHILE $A < a$ $\qquad B = b * A \wedge A < a \longrightarrow B + b = b * (A + 1)$
DO $\qquad \wedge A \leq a \qquad \wedge A + 1 \leq a$
$\quad B := B + b;$
$\quad A := A + 1$
OD $\qquad B = b * A \wedge A \geq a \longrightarrow B = b * a$
$\{\ B = b * a\ \}$ $\qquad \wedge A \leq a$

Example 2

$\{ \ a \geq 0 \ \wedge \ b \geq 0 \ \}$
$A := 0;$
$B := 0;$ — $0 = b * 0 \wedge 0 \leq a$ ✓
INV { B = b * A ∧ A≤ a}
WHILE $A < a$ — $B = b * A \wedge A < a \wedge A \leq a \longrightarrow B + b = b * (A + 1) \wedge A + 1 \leq a$ ✓
DO
  $B := B + b;$
  $A := A + 1$
OD — $B = b * A \wedge A \geq a \wedge A \leq a \longrightarrow B = b * a$ ✓
$\{ \ B = b * a \ \}$

Example 3

$\{\ a \geq 0 \ \wedge \ b > 0\ \}$
$A := a;$
$B := 1;$

WHILE $A \neq 0$
DO
  $B := B * b;$
  $A := A - 1$
OD

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$

$A := a;$	$A =$	a	a-1	a-2	a-3	...
$B := 1;$	$B =$	1	b	b*b	b*b*b	...

WHILE $A \neq 0$
DO
$\quad B := B * b;$
$\quad A := A - 1$
OD

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$
$A := a;$
$B := 1;$

$A =$	a	a-1	a-2	a-3	...
$B =$	1	b	b*b	b*b*b	...

WHILE $A \neq 0$
DO
$B := B * b;$
$A := A - 1$
OD
$\{\ B = b^a\ \}$

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$
$A := a;$
$B := 1;$

$A =$	a	a-1	a-2	a-3	...
$B =$	1	b	b*b	b*b*b	...
				$= b^3 = b^{a-A}$	

WHILE $A \neq 0$
DO
  $B := B * b;$
  $A := A - 1$
OD
$\{\ B = b^a\ \}$

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$
$A := a;$
$B := 1;$

$A =$	a	a-1	a-2	a-3	...
$B =$	1	b	b*b	b*b*b	...
				$= b^3 = b^{a-A}$	

INV $\{\ B = b^{a-A}\}$
WHILE $A \neq 0$
DO
  $B := B * b;$
  $A := A - 1$
OD
$\{\ B = b^a\ \}$

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$
$A := a;$
$B := 1;$

$A =$	a	a-1	a-2	a-3	...
$B =$	1	b	b*b	b*b*b	...

$= b^3 = b^{a-A}$

$1 = b^{a-a}$

INV $\{\ B = b^{a-A} \}$
WHILE $A \neq 0$
DO
  $B := B * b;$
  $A := A - 1$
OD
$\{\ B = b^a\ \}$

$B = b^{a-A} \wedge A \neq 0 \longrightarrow B * b = b^{a-(A-1)}$

$B = b^{a-A} \wedge A = 0 \longrightarrow B = b^a$

Example 3

$\{\ a \geq 0\ \wedge\ b > 0\ \}$
$A := a;$
$B := 1;$

$A =$	a	a-1	a-2	a-3	...
$B =$	1	b	b*b	b*b*b	...
				$= b^3 = b^{a-A}$	

$1 = b^{a-a}$

INV $\{\ B = b^{a-A}\ \wedge$ A$\leq$ a$\}$
WHILE $A \neq 0$
DO
  $B := B * b;$
  $A := A - 1$
OD
$\{\ B = b^a\ \}$

$B = b^{a-A} \wedge A \neq 0 \longrightarrow\ B * b = b^{a-(A-1)}$

$B = b^{a-A} \wedge A = 0\ \longrightarrow\ B = b^a$

Example 4

$\{ \ True \ \}$
$X := x;$
$Y := [];$

WHILE $X \neq []$

DO
  $Y := (hd \ X \# Y);$
  $X := tl \ X$
OD

Example 4

$\{\ \mathit{True}\ \}$
$X := x;$
$Y := [];$

$X =$	$[x_0; x_1; x_2 ...]$	$[x_1; x_2 ...]$	$[x_2 ...]$	...
$Y =$	$[]$	$x_0 \# []$	$x_1 \# x_0 \# []$	...

WHILE $X \neq []$

DO
$\quad Y := (\mathit{hd}\ X \# Y);$
$\quad X := \mathit{tl}\ X$
OD

Example 4

$\{\ \mathit{True}\ \}$
$X := x;$
$Y := [];$

$X =$	$[x_0; x_1; x_2 \ldots]$	$[x_1; x_2 \ldots]$	$[x_2 \ldots]$	$\ldots$
$Y =$	$[]$	$x_0 \# []$	$x_1 \# x_0 \# []$	$\ldots$

WHILE $X \neq []$

DO
  $Y := (\mathit{hd}\ X \# Y);$
  $X := \mathit{tl}\ X$
OD
$\{\ Y = \mathit{rev}\ x\ \}$

Example 4

$\{ \ True \ \}$
$X := x;$
$Y := [];$

$X =$	$[x_0; x_1; x_2 ...]$	$[x_1; x_2 ...]$	$[x_2 ...]$	...
$Y =$	$[]$	$x_0 \# []$	$x_1 \# x_0 \# []$	...

INV $\{ \ (rev \ X) @ Y = rev \ x \}$
WHILE $X \neq []$

DO
$\quad Y := (hd \ X \# Y);$
$\quad X := tl \ X$
OD
$\{ \ Y = rev \ x \ \}$

Example 4

$\{\ True\ \}$
$X := x;$ $\quad X = [x_0; x_1; x_2 ...] \quad [x_1; x_2 ...] \quad [x_2 ...] \quad ...$
$Y := [];$ $\quad Y = [] \quad x_0 \# [] \quad x_1 \# x_0 \# [] \quad ...$

$(rev\ x) @ [] = rev\ x$

INV $\{\ (rev\ X) @ Y = rev\ x \}$
WHILE $X \neq []$

$(rev\ X) @ Y = rev\ x\ \wedge X \neq [] \longrightarrow$
$(rev\ (tl\ X)) @ ((hd\ X) \# Y) = rev\ x$

DO
$\quad Y := (hd\ X \# Y);$
$\quad X := tl\ X$
OD $\quad (rev\ X) @ Y = rev\ x\ \wedge X = []\ \longrightarrow Y = rev\ x$
$\{\ Y = rev\ x\ \}$

Example 4

$\{ \; True \; \}$
$X := x;$ $\quad X = \quad [x_0; x_1; x_2 ...] \quad [x_1; x_2 ...] \quad [x_2 ...] \quad ...$
$Y := [];$ $\quad Y = \quad [] \quad x_0 \# [] \quad x_1 \# x_0 \# [] \quad ...$

$(rev\ x) @ [] = rev\ x$

$\text{INV} \; \{ \; (rev\ X) @ Y = rev\ x \}$
$\text{WHILE}\ X \neq []$

$(rev\ X) @ Y = rev\ x \; \wedge X \neq [] \longrightarrow$
$(rev\ (tl\ X)) @ ((hd\ X) \# Y) = rev\ x$
$= (rev\ X) @ Y$
$= (rev\ ((hd\ X) \# (tl\ X))) @ Y$

DO
$\quad Y := (hd\ X \# Y);$
$\quad X := tl\ X$
OD

$(rev\ X) @ Y = rev\ x \; \wedge X = [] \; \longrightarrow Y = rev\ x$

$\{ \; Y = rev\ x \; \}$

Example 5

$A := a$; $B := b$; $C := 1$;

WHILE $B \neq 0$
DO

WHILE (B mod 2 = 0)

DO
$A := A * A$;
$B := B$ *div* 2;
OD
$C := C * A$;
$B := B - 1$
OD

Example 5

Try with $b = 10 = 2^1 + 2^3$ or $b = 12 = 2^2 + 2^3$ (and e.g. a=3)

$A := a$; $B := b$; $C := 1$;

WHILE $B \neq 0$
DO

WHILE (B mod 2 = 0)

DO
$A := A * A$;
$B := B$ *div* 2;
OD
$C := C * A$;
$B := B - 1$
OD

Example 5

Try with $b = 10 = 2^1 + 2^3$ or $b = 12 = 2^2 + 2^3$ (and e.g. a=3)

$\{ a \geq 0 \land b \geq 0 \}$
$A := a;\ B := b;\ C := 1;$

WHILE $B \neq 0$
DO

    WHILE (B mod 2 = 0)

        DO
        $A := A * A;$
        $B := B\ div\ 2;$
        OD
    $C := C * A;$
    $B := B - 1$
OD
$\{ C = a^b \}$

Example 5

Try with $b = 10 = 2^1 + 2^3$ or $b = 12 = 2^2 + 2^3$ (and e.g. a=3)

$\{ a \geq 0 \wedge b \geq 0 \}$
$A := a;\ B := b;\ C := 1;$
INV $\{ a^b = C * A^B \}$
WHILE $B \neq 0$
DO
INV $\{ a^b = C * A^B \}$
  WHILE (B mod 2 = 0)

    DO
    $A := A * A;$
    $B := B\ div\ 2;$
    OD
  $C := C * A;$
  $B := B - 1$
OD
$\{ C = a^b \}$

Example 5

Try with $b = 10 = 2^1 + 2^3$ or $b = 12 = 2^2 + 2^3$ (and e.g. a=3)

$\{ a \geq 0 \wedge b \geq 0 \}$
$A := a;\ B := b;\ C := 1;$ $\quad a^b = 1 * a^b$
INV $\{ a^b = C * A^B \}$
WHILE $B \neq 0$ $\quad a^b = C * A^B \wedge B \neq 0 \longrightarrow a^b = (C * A) * A^{B-1}$
DO
INV $\{ a^b = C * A^B \}$
WHILE (B mod 2 = 0)
$a^b = C * A^B \wedge B\ mod\ 2 = 0 \longrightarrow a^b = C * (A * A)^{B\ div\ 2}$
DO
$A := A * A;$
$B := B\ div\ 2;$
OD
$C := C * A;$
$B := B - 1$
OD $\quad a^b = C * A^B \wedge B = 0 \longrightarrow C = a^b$
$\{ C = a^b \}$

Example 6

$l := 0; u := \mathit{length}\ A - 1; A := a$

WHILE $l \leq u$
DO

WHILE $l < \mathit{length}\ A \wedge A!l \leq \mathit{piv}$ DO $l := l + 1$ OD;

WHILE $0 < u \wedge \mathit{piv} \leq A!u$ DO $u := u - 1$ OD;

IF $l \leq u$ THEN $A := A[l := A!u, u := A!l]$ ELSE SKIP FI
OD

Example 6

$l := 0; u := \mathit{length}\ A - 1; A := a$

WHILE $l \leq u$
DO

WHILE $l < \mathit{length}\ A \wedge A!l \leq \mathit{piv}$ DO $l := l + 1$ OD;

WHILE $0 < u \wedge \mathit{piv} \leq A!u$ DO $u := u - 1$ OD;

IF $l \leq u$ THEN $A := A[l := A!u, u := A!l]$ ELSE SKIP FI
OD
$\{\ \mathit{LEQ}\ A\ u \wedge \mathit{EQ}\ A\ u\ l \wedge \mathit{GEQ}\ A\ l \wedge A$ permutes $a\ \}$

Example 6

$$LEQ\ A\ n = \forall k.\ k < n \longrightarrow\ A!k \le piv$$
$$GEQ\ A\ n = \forall k.\ n < k < length\ A \longrightarrow A!k \ge piv$$
$$EQ\ A\ n\ m = \forall k.\ n \le k \le m \longrightarrow A!k = piv$$

$\{\ 0 < length\ A\ \}$
$l := 0; u := length\ A - 1; A := a$

WHILE $l \le u$
DO

WHILE $l < length\ A \wedge A!l \le piv$ DO $l := l + 1$ OD;

WHILE $0 < u \wedge piv \le A!u$ DO $u := u - 1$ OD;

IF $l \le u$ THEN $A := A[l := A!u, u := A!l]$ ELSE SKIP FI
OD
$\{\ LEQ\ A\ u \wedge EQ\ A\ u\ l \wedge GEQ\ A\ l \wedge A$ permutes $a\ \}$

Example 6

$LEQ\ A\ n = \forall k.\ k < n \longrightarrow\ A!k \leq piv$
$GEQ\ A\ n = \forall k.\ n < k < length\ A \longrightarrow A!k \geq piv$
$EQ\ A\ n\ m = \forall k.\ n \leq k \leq m \longrightarrow A!k = piv$

$\{\ 0 < length\ A\ \}$
$l := 0; u := length\ A - 1; A := a$
INV $\{\ LEQ\ A\ l \wedge GEQ\ A\ u \wedge u < length\ A \wedge l \leq length\ A \wedge A$ permutes $a\}$
WHILE $l \leq u$
DO
  INV $\{\ LEQ\ A\ l \wedge GEQ\ A\ u \wedge u < length\ A \wedge l \leq length\ A \wedge A$ permutes $a\}$
  WHILE $l < length\ A \wedge A!l \leq piv$ DO $l := l + 1$ OD;

  INV $\{\ LEQ\ A\ l \wedge GEQ\ A\ u \wedge u < length\ A \wedge l \leq length\ A \wedge A$ permutes $a\}$
  WHILE $0 < u \wedge piv \leq A!u$ DO $u := u - 1$ OD;

  IF $l \leq u$ THEN $A := A[l := A!u, u := A!l]$ ELSE SKIP FI
OD
$\{\ LEQ\ A\ u \wedge EQ\ A\ u\ l \wedge GEQ\ A\ l \wedge A$ permutes $a\ \}$

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→field :== v

Definition:

"*List nxt p Ps*" is a linked list, starting at pointer *p* following the next pointer through the function *nxt*, and where *Ps* contains the list of the pointers of the linked list.

$\{\ List\ nxt\ p\ Ps \wedge X \in Ps\ \}$

WHILE $p \neq Null \wedge p \neq Ref\ X$

DO

$p := p \rightarrow nxt;$

OD

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→field :== v

Definition:

"*List nxt p Ps*" is a linked list, starting at pointer *p* following the next pointer through the function *nxt*, and where *Ps* contains the list of the pointers of the linked list.

$\{ \mathit{List}\ \mathit{nxt}\ p\ \mathit{Ps} \wedge X \in \mathit{Ps} \}$

WHILE $p \neq \mathit{Null} \wedge p \neq \mathit{Ref}\ X$

DO

$p := p \rightarrow \mathit{nxt};$

OD

$\{ p = \mathit{Ref}\ X \}$

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→field :== v

Definition:

"*List nxt p Ps*" is a linked list, starting at pointer *p* following the next pointer through the function *nxt*, and where *Ps* contains the list of the pointers of the linked list.

$\{\ \mathit{List}\ \mathit{nxt}\ p\ \mathit{Ps} \wedge X \in \mathit{Ps}\ \}$

WHILE $p \neq \mathit{Null} \wedge p \neq \mathit{Ref}\ X$

DO

$p := p \rightarrow \mathit{nxt};$

OD

$\{\ p = \mathit{Ref}\ X\ \}$

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→field :== v

Definition:

"*List nxt p Ps*" is a linked list, starting at pointer *p* following the next pointer through the function *nxt*, and where *Ps* contains the list of the pointers of the linked list.

$\{ \mathit{List}\ \mathit{nxt}\ p\ \mathit{Ps} \wedge X \in \mathit{Ps} \}$

INV $\{ \exists \mathit{Qs}.\ \mathit{List}\ \mathit{nxt}\ p\ \mathit{Qs} \wedge X \in \mathit{Qs} \}$

WHILE $p \neq \mathit{Null} \wedge p \neq \mathit{Ref}\ X$

DO

$p := p \rightarrow \mathit{nxt};$

OD

$\{ p = \mathit{Ref}\ X \}$

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→field :== v

Definition:

"*List nxt p Ps*" is a linked list, starting at pointer *p* following the next pointer through the function *nxt*, and where *Ps* contains the list of the pointers of the linked list.

$\{\ List\ nxt\ p\ Ps \wedge X \in Ps\ \}$

INV $\{\ \exists Qs.\ List\ nxt\ p\ Qs \wedge X \in Qs\}$

WHILE $p \neq Null \wedge p \neq Ref\ X$

DO

$p := p \rightarrow nxt;$

OD

$\{\ p = Ref\ X\ \}$

$\exists Qs.\ List\ nxt\ p\ Qs \wedge X \in Qs$

$\exists Qs.\ List\ nxt\ p\ Qs \wedge X \in Qs$
$\wedge p \neq Null \wedge p \neq Ref\ X \longrightarrow$
$\exists Qs.\ List\ nxt\ (p \rightarrow nxt)\ Qs \wedge X \in Qs$

$\exists Qs.\ List\ nxt\ p\ Qs \wedge X \in Qs$
$\wedge (p = Null \vee p = Ref\ X) \longrightarrow\ p = Ref\ X$

Example 8

What is is Isabelle function doing?

$$\begin{array}{l} \textit{fun } f \ :: \ 'a \ list \Rightarrow' a \ list \Rightarrow' a \ list \ where \\ \quad f \ [] \ ys = ys| \\ \quad f \ xs \ [] = xs| \\ \quad f \ (x\#xs) \ (y\#ys) = x\#y\# \ f \ xs \ ys \end{array}$$

Example 8

What is is Isabelle function doing?

$fun\ \textcolor{red}{splice}\ ::\ 'a\ list \Rightarrow' a\ list \Rightarrow' a\ list\ where$
$\textcolor{red}{splice}\ []\ ys = ys|$
$\textcolor{red}{splice}\ xs\ [] = xs|$
$\textcolor{red}{splice}\ (x\#xs)\ (y\#ys) = x\#y\#\ f\ xs\ ys$

Example 8

What is is Isabelle function doing?

$$\begin{array}{l} \textit{fun}\ \textit{splice}\ ::\ {}'a\ \textit{list} \Rightarrow' a\ \textit{list} \Rightarrow' a\ \textit{list}\ \textit{where} \\ \quad \textit{splice}\ []\ \textit{ys} = \textit{ys} | \\ \quad \textit{splice}\ \textit{xs}\ [] = \textit{xs} | \\ \quad \textit{splice}\ (x \# \textit{xs})\ (y \# \textit{ys}) = x \# y \#\ f\ \textit{xs}\ \textit{ys} \end{array}$$

Let's write it with linked lists!

Example 8

$$\{\ \mathit{List}\ \mathit{nxt}\ p\ \mathit{Ps} \wedge \mathit{List}\ \mathit{nxt}\ q\ \mathit{Qs} \wedge (\mathit{set}\ \mathit{Ps} \cap \mathit{set}\ \mathit{Qs}) = \{\} \wedge \mathit{size}\ \mathit{Qs} \leq \mathit{size}\ \mathit{Ps}\ \}$$

$$\{\ \mathit{List}\ \mathit{nxt}\ p\ (\mathit{splice}\ \mathit{Ps}\ \mathit{Qs})\ \}$$

Example 8

$\{\ \mathit{List\ nxt\ p\ Ps} \wedge \mathit{List\ nxt\ q\ Qs} \wedge (\mathit{set\ Ps} \cap \mathit{set\ Qs}) = \{\} \wedge \mathit{size\ Qs} \leq \mathit{size\ Ps}\ \}$

$pp := p;$

WHILE $q \neq \mathit{Null}$

DO

$qq := q \rightarrow nxt;\ q \rightarrow nxt := pp \rightarrow nxt;\ pp \rightarrow nxt = q;\ pp := q \rightarrow nxt;\ q := qq;$

OD

$\{\ \mathit{List\ nxt\ p\ (splice\ Ps\ Qs)}\ \}$

Example 8

$List\ nxt\ p\ Ps = Path\ nxt\ p\ Ps\ Null$

$Path\ nxt\ p\ Ps\ Null$ is a linked list from p to q following function nxt and containing list of pointers Ps

$\{ List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps \}$

$pp := p;$

INV $\{$

$\}$

WHILE $q \neq Null$

DO

$qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{ List\ nxt\ p\ (splice\ Ps\ Qs) \}$

Example 8

List nxt p Ps = *Path nxt p Ps Null*

Path nxt p Ps Null is a linked list from *p* to *q* following function *nxt* and containing list of pointers *Ps*

$\{ \; List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps \; \}$

$pp := p;$

INV $\{ \; \exists PPs$

$\quad List\ nxt\ pp\ PPs$

$\}$

WHILE $q \neq Null$

DO

$\quad qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{ \; List\ nxt\ p\ (splice\ Ps\ Qs) \; \}$

Example 8

$List\ nxt\ p\ Ps = Path\ nxt\ p\ Ps\ Null$

$Path\ nxt\ p\ Ps\ Null$ is a linked list from p to q following function nxt and containing list of pointers Ps

$\{\ List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps\ \}$

$pp := p;$

INV $\{\ \exists PPs\ QQs$

$\quad List\ nxt\ pp\ PPs\ \wedge List\ nxt\ q\ QQs$

$\}$

WHILE $q \neq Null$

DO

$\quad qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{\ List\ nxt\ p\ (splice\ Ps\ Qs)\ \}$

Example 8

$List\ nxt\ p\ Ps = Path\ nxt\ p\ Ps\ Null$

$Path\ nxt\ p\ Ps\ Null$ is a linked list from p to q following function nxt and containing list of pointers Ps

$\{\ List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps\ \}$

$pp := p;$

INV $\{\ \exists PPs\ QQs\ PPPs.$

$List\ nxt\ pp\ PPs\ \wedge List\ nxt\ q\ QQs\ \wedge Path\ nxt\ p\ PPPs\ pp$

$\}$

WHILE $q \neq Null$

DO

$qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{\ List\ nxt\ p\ (splice\ Ps\ Qs)\ \}$

Example 8

$List\ nxt\ p\ Ps = Path\ nxt\ p\ Ps\ Null$

$Path\ nxt\ p\ Ps\ Null$ is a linked list from p to q following function nxt and containing list of pointers Ps

$$\{\ List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps\ \}$$

$pp := p;$

INV $\{\ \exists PPs\ QQs\ PPPs.$

$\quad List\ nxt\ pp\ PPs\ \wedge List\ nxt\ q\ QQs\ \wedge Path\ nxt\ p\ PPPs\ pp$

$\quad \wedge PPPs@splice\ PPs\ QQs\ =\ splice\ Ps\ Qs$

$\}$

WHILE $q \neq Null$

DO

$\quad qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{\ List\ nxt\ p\ (splice\ Ps\ Qs)\ \}$

Example 8

$List\ nxt\ p\ Ps = Path\ nxt\ p\ Ps\ Null$

$Path\ nxt\ p\ Ps\ Null$ is a linked list from p to q following function nxt and containing list of pointers Ps

$\{\ List\ nxt\ p\ Ps \wedge List\ nxt\ q\ Qs \wedge (set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps\ \}$

$pp := p;$

$\mathrm{INV}\ \{\ \exists PPs\ QQs\ PPPs.\ \ size\ QQs \leq size\ PPs\ \wedge$

$List\ nxt\ pp\ PPs\ \wedge List\ nxt\ q\ QQs\ \wedge Path\ nxt\ p\ PPPs\ pp$

$\wedge\ PPPs@splice\ PPs\ QQs\ =\ splice\ Ps\ Qs$

$\}$

$\mathrm{WHILE}\ q \neq Null$

DO

$qq := q \rightarrow nxt; q \rightarrow nxt := pp \rightarrow nxt; pp \rightarrow nxt = q; pp := q \rightarrow nxt; q := qq;$

OD

$\{\ List\ nxt\ p\ (splice\ Ps\ Qs)\ \}$

Example 8

List nxt p Ps = *Path nxt p Ps Null*

Path nxt p Ps Null is a linked list from *p* to *q* following function *nxt* and containing list of pointers *Ps*

$\{$ *List nxt p Ps* $\wedge$ *List nxt q Qs* $\wedge$ (*set Ps* $\cap$ *set Qs*) $= \{\} \wedge$ *size Qs* $\leq$ *size Ps* $\}$
pp := *p*;
INV $\{$ $\exists$*PPs QQs PPPs.* *size QQs* $\leq$ *size PPs* $\wedge$
List nxt pp PPs $\wedge$ *List nxt q QQs* $\wedge$ *Path nxt p PPPs pp*
$\wedge$ *PPPs*@*splice PPs QQs* = *splice Ps Qs* $\wedge$
set PPs $\cap$ *set QQs* $= \{\} \wedge$ *distinct PPPs* $\wedge$ *set PPPs* $\cap$ (*set PPs* $\cup$ *set QQs*) $= \{\}$
$\}$
WHILE $q \neq$ *Null*
DO
qq := *q* $\rightarrow$ *nxt*; *q* $\rightarrow$ *nxt* := *pp* $\rightarrow$ *nxt*; *pp* $\rightarrow$ *nxt* = *q*; *pp* := *q* $\rightarrow$ *nxt*; *q* := *qq*;
OD
$\{$ *List nxt p* (*splice Ps Qs*) $\}$

Demo

COMP4161
Advanced Topics in Software Verification

Exam Prep

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

Last Time

- ➜ The automated proof method **wp**
- ➜ The C Parser and translating C into Simpl
- ➜ AutoCorres and translating Simpl into monadic form
- ➜ The option and exception monads

Exam

- 24h take-home exam (same as previous years)

Exam

- ➜ 24h take-home exam (same as previous years)

- ➜ Open book: can use any passive resource (books, slides, google, etc)
- ➜ **Not** allowed to ask for help from anyone
- ➜ **Not** allowed AI assistance for technical support (e.g. ChatGPT).
- ➜ starts 8am AEST, Monday 4th Dec 2023, ends 7:59am AEST, Tuesday 5nd Dec 2023

Exam

- 24h take-home exam (same as previous years)

- Open book: can use any passive resource (books, slides, google, etc)
- **Not** allowed to ask for help from anyone
- **Not** allowed AI assistance for technical support (e.g. ChatGPT).
- starts 8am AEST, Monday 4th Dec 2023, ends 7:59am AEST, Tuesday 5nd Dec 2023

- Should be doable in about 4-6 hours.
  The 24h are for flexibility not for you to stay awake actual 24 hours.
- Recommend to start early, finish the easy questions first.
- Take breaks. Don't forget to eat :-)
- If there are clarification questions, make **private** threads on Ed.

Content

- ➜ Foundations & Principles
  - Intro, Lambda calculus, natural deduction [1,2]
  - Higher Order Logic, Isar (part 1) [2,3[a]]
  - Term rewriting [3,4]
- ➜ Proof & Specification Techniques
  - Inductively defined sets, rule induction [4,5]
  - Datatype induction, primitive recursion [5,7]
  - General recursive functions, termination proofs [7]
  - Proof automation, Isar (part 2) [8[b]]
  - Hoare logic, proofs about programs, invariants [8,9]
  - C verification [9,10]
  - Practice, questions, exam prep [10[c]]

[a] a1 due; [b] a2 due; [c] a3 due

COMP4161
Advanced Topics in Software Verification

λ

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned so far...

- ➜ λ calculus syntax
- ➜ free variables, substitution
- ➜ β reduction
- ➜ α and η conversion
- ➜ β reduction is confluent
- ➜ λ calculus is very expressive (turing complete)
- ➜ λ calculus results in an inconsistent logic

COMP4161
Advanced Topics in Software Verification

$$\lambda^{\rightarrow}$$

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned so far...

- Simply typed lambda calculus: $\lambda^{\rightarrow}$
- Typing rules for $\lambda^{\rightarrow}$, type variables, type contexts
- β-reduction in $\lambda^{\rightarrow}$ satisfies subject reduction
- β-reduction in $\lambda^{\rightarrow}$ always terminates
- Types and terms in Isabelle

COMP4161
Advanced Topics in Software Verification

$\lambda^{\rightarrow}$ and HOL

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

What we have learned so far...

- natural deduction rules for $\wedge$, $\vee$, $\longrightarrow$, $\neg$, iff...
- proof by assumption, by intro rule, elim rule
- safe and unsafe rules

- indent your proofs! (one space per subgoal)
- prefer implicit backtracking (chaining) or *rule_tac*, instead of *back*
- *prefer* and *defer*
- *oops* and *sorry*

COMP4161
Advanced Topics in Software Verification

HOL

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned so far...

- Isar style proofs
- proof, qed
- assumes, shows
- fix, obtain
- moreover, ultimately
- forward, backward
- mixing proof styles

COMP4161
Advanced Topics in Software Verification

HOL

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned today ...

- ➜ Defining HOL
- ➜ Higher Order Abstract Syntax
- ➜ Deriving proof rules
- ➜ More automation
- ➜ Equations and Term Rewriting

COMP4161
Advanced Topics in Software Verification

→

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today...

- ➜ Equations and Term Rewriting
- ➜ Confluence and Termination of reduction systems
- ➜ Term Rewriting in Isabelle

COMP4161
Advanced Topics in Software Verification

$\longrightarrow$

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned today ...

- ➜ Conditional term rewriting
- ➜ Congruence rules
- ➜ AC rules
- ➜ More on confluence

COMP4161
Advanced Topics in Software Verification

{}

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned today ...

- ➜ Sets
- ➜ Type Definitions
- ➜ Inductive Definitions

COMP4161
Advanced Topics in Software Verification

{}

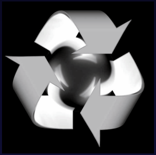
Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have learned today ...

➜ Formal background of inductive definitions
➜ Definition by intersection
➜ Computation by iteration
➜ Formalisation in Isabelle

COMP4161
Advanced Topics in Software Verification



Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today ...

- ➜ Datatypes
- ➜ Primitive recursion
- ➜ Case distinction
- ➜ Structural Induction

COMP4161
Advanced Topics in Software Verification

fun

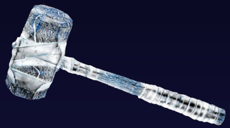
Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have see today ...

- ➜ General recursion with **fun**/**function**
- ➜ Induction over recursive functions
- ➜ How **fun** works
- ➜ Termination, partial functions, congruence rules

COMP4161
Advanced Topics in Software Verification



Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today ...

- ➜ sledgehammer
- ➜ nitpick
- ➜ quickcheck

COMP4161
Advanced Topics in Software Verification

$$\{P\} \ldots \{Q\}$$

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today ...

- ➜ Syntax of a simple imperative language
- ➜ Operational semantics
- ➜ Program proof on operational semantics
- ➜ Hoare logic rules
- ➜ Soundness of Hoare logic

COMP4161
Advanced Topics in Software Verification

$$\{P\} \ldots \{Q\}$$

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today ...

- ➜ Weakest precondition
- ➜ Verification conditions
- ➜ Example program proofs
- ➜ Arrays, pointers

COMP4161
Advanced Topics in Software Verification

$\gg=$

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

We have seen today

- ➜ Deep and shallow embeddings
- ➜ Isabelle records
- ➜ Nondeterministic State Monad with Failure
- ➜ Monadic Weakest Precondition Rules

COMP4161
Advanced Topics in Software Verification

C

Gerwin Klein, Miki Tanaka, Johannes Åman Pohjola, Rob Sison

T3/2023

Today we have seen

- ➜ The automated proof method **wp**
- ➜ The C Parser and translating C into Simpl
- ➜ AutoCorres and translating Simpl into monadic form
- ➜ The option and exception monads

