COMP4161
Advanced Topics in Software Verification

INV & Exam Prep

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

COMP4161
Advanced Topics in Software Verification

INV

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

Practice with invariants!

Practice with invariants!

Recall:

- invariants are neede[illegible]ate the application of hoare rules
- they are used by th[illegible]recondition calculus to deal with loops

Practice with invariants!

Recall:

- ➜ invariants are neede[illegible]ate the application of hoare rules
- ➜ they are used by th[illegible]recondition 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 neede[illegible]ate the application of hoare rules
- they are used by th[illegible]recondition 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"

Example 1

$A := 0;$
$B := 0;$

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

Example 1

$A := 0;$
$B := 0;$

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

Example 1

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

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

Example 1

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

Example 1

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

$$B = b * A \wedge A \neq a \longrightarrow B + b = b * (A + 1)$$

$$B = b * A \wedge A = a \longrightarrow B = b * a$$

Example 2

$A := 0;$
$B := 0;$

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

Example 2

$A := 0;$
$B := 0;$

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

Example 2

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

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

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

$$0 = b * 0$$

$$B = b * A \wedge A < a \;\; \longrightarrow \;\; B + b = b * (A + 1)$$

$$B = b * A \wedge A \geq a \;\; \longrightarrow \;\; B = b * a$$

Example 2

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

$0 = b * 0 \wedge 0 \leq a$

$B = b * A \wedge A < a \longrightarrow B + b = b * (A + 1)$
$\wedge A \leq a \qquad \wedge A + 1 \leq a$

$B = b * A \wedge A \geq a \longrightarrow B = b * a$
$\wedge A \leq a$

Example 3

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

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

Example 3

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

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

Example 3

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

$A := a;$

$B := 1;$

WHILE $A \neq 0$

DO

  $B := B * b;$

  $A := A - 1$

OD

$\{\ B = b^a\ \}$

Example 3

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

Example 3

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

$^{a-a}$

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

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

Example 4

$X := x;$
$Y := [];$

WHILE $X \neq []$

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

Example 4

$X := x;$
$Y := [];$

WHILE $X \neq []$

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

Example 4

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

WHILE $X \neq []$

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

Example 4

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

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

Example 4

$\{ \ True \ \}$
$X := x;$
$Y := [];$ $\qquad\qquad (\ldots = rev\ x$
INV $\{ \ (rev\ X)@Y = rev\ x \ \}$
WHILE $X \neq []$

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

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

Example 5

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

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)

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

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)

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

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

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)

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

$\{ \ a \geq 0 \wedge b \geq 0 \ \}$
$A := a;\ B := b;\ C := 1$ $\quad = 1 * a^b$
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 $l \le u$
DO

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

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

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

Example 6

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

WHILE $l \leq u$
DO

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

  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 6

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

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

WHILE $l \leq u$
DO

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

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 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→[illegible]

Definition:

"*List nxt p Ps*" is a [illegible] starting at pointer p following the next pointer through the [illegible]xt, 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

$\quad p := p \rightarrow nxt;$

OD

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→

Definition:

"*List nxt p Ps*" is a starting at pointer p following the next pointer through the xt, 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

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

OD

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

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→

Definition:

"*List nxt p Ps*" is a starting at pointer p following the next pointer through the *xt*, 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

$\quad p := p \rightarrow nxt;$

OD

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

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p$\rightarrow$field

Pointer update: p$\rightarrow$

Definition:

"*List nxt p Ps*" is a starting at pointer *p* following the next pointer through the *xt*, 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

$\quad p := p \rightarrow nxt;$

OD

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

Example 7

Reminder:

datatype ref = Ref int | Null

Pointer access: p→field

Pointer update: p→

Definition:

"*List nxt p Ps*" is a … starting at pointer *p* following the next pointer through the …*xt*, 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

$\quad p := p \rightarrow nxt;$

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?

```
fun f :: 'a list ⇒ 'a list ⇒ 'a list where
  f [] ys = ys |
  f xs [] = xs |
  f (x#xs) (y#ys) = x#y# f xs ys
```


Example 8

What is is Isabelle function doing?

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

Example 8

What is is Isabelle function doing?

fun *splice* :: *'a list* ⇒ [illegible] *a list where*
  splice [] *ys* = *ys*|
  splice *xs* [] = *xs*|
  splice (*x*#*xs*) (*y*#*ys*) = *x*#*y*# *f* *xs* *ys*

Let's write it with linked lists!

Example 8

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

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

Example 8

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

pp := *p*;

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* $\land$ *List* [illegible] $(set\ Ps \cap set\ Qs) = \{\} \land size\ Qs \le size\ Ps$ $\}$

$pp := p;$

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 pointer

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

$pp := p;$

INV $\{\ \exists PPs$

$List\ nxt\ pp\ P$

$\}$

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 $\{\ \exists PPs\ QQs$

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

$\}$

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

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

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*.

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

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

$\}$

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* [illegible] $(set\ Ps \cap set\ Qs) = \{\} \wedge size\ Qs \leq size\ Ps$ $\}$

$pp := p;$

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

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

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

$\}$

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.

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

$pp := p;$

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, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

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 exa[illegible] previous years)

Exam

- 24h take-home exa previous years)
- Open book: can us e resource (books, slides, google, etc)
- **Not** allowed to ask for help from anyone
- starts 8am AEST, Thu 1st Dec 2022, ends 7:59am AEST, Fri 2nd Dec 2022

Exam

- 24h take-home exa … previous years)
- Open book: can us … e resource (books, slides, google, etc)
- **Not** allowed to ask for help from anyone
- starts 8am AEST, Thu 1st Dec 2022, ends 7:59am AEST, Fri 2nd Dec 2022
- 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 email the lecturers.

Content

- ➜ Foundations & Principles
  - Intro, Lambda [illegible]atural deduction [1,2]
  - Higher Order [illegible] part 1) [2,3[a]]
  - Term rewriting [illegible] [3,4]

- ➜ Proof & Specification Techniques
  - Inductively defined sets, rule induction [4,5]
  - Datatype induction, primitive recursion [5,7]
  - General recursive functions, termination proofs [7[b]]
  - Proof automation, Isar (part 2) [8]
  - 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

MP4161
Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned so far...

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

Proofcraft

UNSW SYDNEY

MP4161

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned so far...

- ➜ Simply typed lamb[illegible] $\lambda^{\rightarrow}$
- ➜ Typing rules for $\lambda^{\rightarrow}$[illegible]bles, type contexts
- ➜ β-reduction in $\lambda^{\rightarrow}$ s[illegible]ject reduction
- ➜ β-reduction in $\lambda^{\rightarrow}$ always terminates
- ➜ Types and terms in Isabelle

MP4161

Advanced Topics in Software Verification

$\lambda^{\rightarrow}$ and HOL

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

What we have learned so far...

- natural deduction r… /, $\longrightarrow$, $\neg$, iff…
- proof by assumptio… ule, elim rule
- safe and unsafe rul…
- indent your proofs! (one space per subgoal)
- prefer implicit backtracking (chaining) or *rule_tac*, instead of *back*
- *prefer* and *defer*
- *oops* and *sorry*

Proofcraft

UNSW SYDNEY

COMP4161
Advanced Topics in Software Verification

HOL

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned so far...

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

Proofcraft

UNSW SYDNEY

MP4161

Advanced Topics in Software Verification

HOL

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned today ...

- Defining HOL
- Higher Order Abstr
- Deriving proof rules
- More automation
- Equations and Term Rewriting

COMP4161
Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today...

- Equations and Tern
- Confluence and Ter reduction systems
- Term Rewriting in I

Proofcraft

UNSW SYDNEY

MP4161

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned today ...

- Conditional term re
- Congruence rules
- AC rules
- More on confluence

MP4161
Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned today ...

- ➜ Sets
- ➜ Type Definitions
- ➜ Inductive Definition

COMP4161
Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have learned today ...

- Formal background [illegible] definitions
- Definition by interse[illegible]
- Computation by ite[illegible]
- Formalisation in Isabelle

COMP4161

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today ...

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

Proofcraft

UNSW SYDNEY

MP4161
Advanced Topics in Software Verification

fun

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen 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, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today ...

- ➜ sledgehammer
- ➜ nitpick
- ➜ quickcheck

MP4161
Advanced Topics in Software Verification

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

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today ...

- Syntax of a simple anguage
- Operational semant
- Program proof on c emantics
- Hoare logic rules
- Soundness of Hoare logic

Proofcraft

UNSW SYDNEY

MP4161
Advanced Topics in Software Verification

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

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today ...

- Weakest preconditi
- Verification conditi
- Example program p
- Arrays, pointers

MP4161
Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

We have seen today

- Deep and shallow e
- Isabelle records
- Nondeterministic St with Failure
- Monadic Weakest P Rules

COMP4161
Advanced Topics in Software Verification

C

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

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