UNSW SYDNEY

COMP4161

Advanced Topics in Software Verification

INV & Exam Prep

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

T3/2022

UNSW SYDNEY

COMP4161

Advanced Topics in Software Verification

INV

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

T3/2022

Practice with invariants!

Recall:

- invariants are nee[illegible]omate the application of hoare rules
- they are used by t[illegible] 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"

Example 1

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

$A := 0;$

$B := 0;$

INV $\{$ B = b * A$\}$

WHILE $A \neq a$

DO

$\quad B := B + b;$

$\quad A := A + 1$

OD

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

$0 = b * 0$

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

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

Example 2

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

$A := 0;$

$B := 0;$

INV $\{\ B = b * A\}$ $\wedge A \leq a$

WHILE $A < a$

DO

$B := B + b;$

$A := A + 1$

OD

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

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

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

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

Example 3

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

$A := a;$

$B := 1;$

INV $\{\ B = b^{a-A}\}$

WHILE $A \neq 0$

DO

$\quad B := B * b;$

$\quad A := A - 1$

OD

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

$B = b^{a-a}$

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

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

Example 4

$\{\ True\ \}$
$X := x;$
$Y := [];$
$(r = rev\ x$
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
  $Y := (hd\ X\#Y);$
  $X := tl\ X$
OD

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

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

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 $\{\ a^b = C * A^B$

$a^b = 1 * a^b$

WHILE $B \neq 0$

$a^b = C * A^B \wedge B \neq 0 \longrightarrow a^b = (C * A) * A$

DO

INV $\{\ a^b = C * A^B\ \}$

WHILE (B mod 2 = 0)

$a^b = C * A^B \wedge B \bmod 2 = 0 \longrightarrow a^b = C * (A * A)^{B\ div\ 2}$

DO

$A := A * A;$

$B := B\ div\ 2;$

OD

$C := C * A;$

$B := B - 1$

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 $l < length\ A \wedge A!l < piv$ 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 the 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

$p := p \rightarrow nxt;$

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

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

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

$\exists Qs.\ List\ nxt\ (p \rightarrow nxt)\ Qs \wedge X \in Q$

Example 8

What is is Isabelle function doing?

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


Let's write it with linked lists!

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 point

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

$pp := p;$

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

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

OD

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

Demo

UNSW SYDNEY

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 ex as previous years)
- Open book: can u sive 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, Lamb[illegible], natural deduction [1,2]
- Higher Orde[illegible]r (part 1) [2,3[a]]
- Term rewriti[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

UNSW SYDNEY

COMP4161
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, sub
- 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

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

T3/2022

We have learned so far...

- Simply typed lam[illegible]us: $\lambda^{\to}$
- Typing rules for λ[illegible]riables, type contexts
- β-reduction in λ^{-}[illegible]ubject reduction
- β-reduction in $\lambda^{\to}$ always terminates
- Types and terms in Isabelle

UNSW SYDNEY

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, $\wedge$, $\vee$, $\longrightarrow$, $\neg$, iff...
- proof by assumption, 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

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

COMP4161
Advanced Topics in Software Verification

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

T3/2022

We have learned today ...

- ➜ Defining HOL
- ➜ Higher Order Abs[illegible]ax
- ➜ Deriving proof rul[illegible]
- ➜ 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 Te[illegible]ng
- ➜ Confluence and T[illegible] of reduction systems
- ➜ Term Rewriting ir[illegible]

COMP4161
Advanced Topics in Software Verification

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

T3/2022

We have learned today ...

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

COMP4161
Advanced Topics in Software Verification

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

T3/2022

We have learned today ...

- Sets
- Type Definitions
- Inductive Definitio

COMP4161
Advanced Topics in Software Verification

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

T3/2022

We have learned today ...

- Formal backgrour tive definitions
- Definition by inte
- Computation by i
- 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 recursio
- Case distinction
- Structural Induction

UNSW SYDNEY

COMP4161
Advanced Topics in Software Verification

fun

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

T3/2022

We have seen today ...

- General recursion **function**
- Induction over rec ctions
- 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

COMP4161
Advanced Topics in Software Verification

$$\{P\} \; t \; \{Q\}$$

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

T3/2022

We have seen today ...

- ➜ Syntax of a simpl[illegible]e language
- ➜ Operational sema[illegible]
- ➜ Program proof on[illegible]al semantics
- ➜ Hoare logic rules
- ➜ Soundness of Hoare logic

COMP4161
Advanced Topics in Software Verification

$$\{P\} \; t \; \{Q\}$$

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

T3/2022

We have seen today ...

- ➜ Weakest precondi
- ➜ Verification condi
- ➜ Example program
- ➜ Arrays, pointers

COMP4161
Advanced Topics in Software Verification

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

We have seen today

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

COMP4161
Advanced Topics in Software Verification

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