Deductive Verification of Programs

6.0 VU Formal Methods in Computer Science

Gernot Salzer

AB Theoretische Informatik und Logik
Institut für Computersprachen

20 November 2013

Topics last time

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions

Structural Operational Semantics (SOS)

The stepwise execution of programs is specified by a transition relation $\Rightarrow$.

$$(\textbf{skip}, \sigma) \Rightarrow \sigma$$

$$\vdots$$

$$(\textbf{while } e \textbf{ do } p \textbf{ od}, \sigma) \Rightarrow \begin{cases} (p; \textbf{while } e \textbf{ do } p \textbf{ od}, \sigma) & \text{if } [e]\,\sigma \neq 0 \\ \sigma & \text{if } [e]\,\sigma = 0 \end{cases}$$

$\Rightarrow$ is deterministic (in the case of TPL).
$\Rightarrow$ allows us to distinguish between aborted runs and endless loops.

The function $[p]\colon \mathcal{S} \mapsto \mathcal{S}$ computed by a TPL program p is defined by

$$[p]\sigma = \sigma' \quad \text{if } (p, \sigma) \stackrel{*}{\Rightarrow} \sigma' \quad \text{(for all states } \sigma\text{)}.$$

$[p]$ is a mathematical function. As such, $[p]\sigma$ can be

- defined: $[p]\sigma$ is a state representing the result of the computation;
- undefined: program p aborts or loops on input σ.

$[p]$ does not distinguish between aborted runs and endless loops.

Natural Semantics

The function computed by a program is defined recursively without reference to a transition relation.

$$[\textbf{skip}]\,\sigma = \sigma$$

$$[v := e]\,\sigma = \sigma' \quad \text{where} \quad \begin{array}{l} \sigma'(v) = [e]\,\sigma \\ \sigma'(x) = \sigma(x) \quad \text{for } x \neq v \end{array}$$

$$[p; q]\,\sigma = [q]\,[p]\,\sigma$$

$$[\textbf{if } e \textbf{ then } p \textbf{ else } q \textbf{ fi}]\,\sigma = \begin{cases} [p]\,\sigma & \text{if } [e]\,\sigma \neq 0 \\ [q]\,\sigma & \text{if } [e]\,\sigma = 0 \end{cases}$$

$$[\textbf{while } e \textbf{ do } p \textbf{ od}]\,\sigma = \begin{cases} [\textbf{while } e \textbf{ do } p \textbf{ od}]\,[p]\,\sigma & \text{if } [e]\,\sigma \neq 0 \\ \sigma & \text{if } [e]\,\sigma = 0 \end{cases}$$

Topics last time

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions

Correctness Assertions

$\{ F \} = \{ \sigma \in \mathcal{S} \mid [F]\,\sigma \neq 0 \}$ ... set of states defined by formula F

"F-states"

$\{ F \}\, p \,\{ G \}$ is true regarding partial correctness (is "partially correct"), if

- ... whenever p starts in an F-state and terminates,
  then p stops in a G-state.
- ... for all states $\sigma \in \mathcal{S}$,
  if $[F]\,\sigma$ is true and $[p]\,\sigma$ is defined,
  then $[G]\,[p]\,\sigma$ is true.

$\{ F \}\, p \,\{ G \}$ is true regarding total correctness (is "totally correct"), if

- ... whenever p starts in an F-state,
  then p terminates and stops in a G-state.
- ... for all states $\sigma \in \mathcal{S}$,
  if $[F]\,\sigma$ is true,
  then $[p]\,\sigma$ is defined and $[G]\,[p]\,\sigma$ is true.

F ... precondition

G ... postcondition

$\{ x = y \}\, x := x + 1 \,\{ x = y + 1 \}$

Totally (and therefore also partially) correct.

$\{ x = y \}\, x := x + 1 \,\{ x > y \}$

Totally (and therefore also partially) correct.

$\{ x \geq y \}\, x := x + 1 \,\{ x = y + 1 \}$

Neither totally nor partially correct. Counterexample: $\sigma(x) = 1$, $\sigma(y) = 0$

$\{ x \geq y \}\, x := x + 1 \,\{ x > y \}$

Totally (and therefore also partially) correct.

$\{ 1 \}\, x := x + 1 \,\{ x > y \}$

Neither totally nor partially correct. Counterexample: $\sigma(x) = 0$, $\sigma(y) = 1$

$\{ 0 \}\, x := x + 1 \,\{ x > y \}$

Totally (and therefore also partially) correct.

$\{ x = y \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x = y + 1 \}$

Totally (and therefore also partially) correct.

$\{ x = y \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x > y \}$

Totally (and therefore also partially) correct.

$\{ x \geq y \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x = y + 1 \}$

Partially but not totally correct. Counterexample: $\sigma(x) = 1$, $\sigma(y) = 0$

$\{ x \geq y \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x > y \}$

Partially but not totally correct. Counterexample: $\sigma(x) = 1$, $\sigma(y) = 0$

$\{ 1 \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x > y \}$

Partially but not totally correct. Counterexample: $\sigma(x) = 1$, $\sigma(y) = 0$

$\{ 0 \}$ **if** $x = y$ **then** $x := x + 1$ **else abort fi** $\{ x > y \}$

Totally (and therefore also partially) correct.

Topics last time

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions

Three ways to prove correctness assertions

Task: Show that $\{ F \}\, p \,\{ G \}$ is partially/totally correct.

Method 1: Hoare calculus.
Decompose correctness assertion into simpler ones (guided by rules) until we obtain true assertions (instances of axioms) and valid formulas.

Method 2: Weakest (liberal) precondition.
Compute the weakest formula F' such that $\{ F' \}\, p \,\{ G \}$ is true, and show that F implies F'.

Method 3: Strongest postcondition.
Compute the strongest formula G' such that $\{ F \}\, p \,\{ G' \}$ is true, and show that G' implies G.

Annotation calculus.
No new method, just combines the above methods for practical usage.

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Hoare Calculus

- finite collection of axioms and rules for deriving correctness assertions
- two calculi: one for partial correctness, one for total correctness
- is sound and complete (with restrictions)

"Sound" means: Every derivable correctness assertion is true.
"Complete" means: Every true correctness assertion is derivable.

Admissible (sound) axiom

Let F and G be formula schemas and p be a program schema.
$\{ F \}\, p \,\{ G \}$ is an admissible axiom if all assertions of this form are true.

"of this form": can be obtained by substitutions, is an instance

$\{ x = 1 \}\, \textbf{skip} \,\{ x = 1 \}$ is an admissible axiom, but very special.

$\{ F \}\, \textbf{skip} \,\{ F \}$ is an admissible axiom (and quite general).

$\{ F \}\, \textbf{skip} \,\{ G \}$ is not admissible. Not true e.g. for $F = 1$ and $G = 0$.

Admissible (sound) rule

Let $X_1, \ldots, X_n$ be schemas for formulas or correctness assertions.

$\frac{X_1 \cdots X_n}{\{\,F\,\}\,p\,\{\,G\,\}}$ is an admissible rule if for all valid/true instances $\hat{X}_1, \ldots, \hat{X}_n, \hat{F}, \hat{p}, \hat{G}$ it holds that:

If $\hat{X}_1, \ldots, \hat{X}_n$ are valid/true then $\{\,\hat{F}\,\}\,\hat{p}\,\{\,\hat{G}\,\}$ is true.

"Whenever the premises hold, the conclusion has to hold."

$\frac{F \Rightarrow G}{\{\,F\,\}\,\mathbf{skip}\,\{\,G\,\}}$ is admissible.

$\frac{F \Rightarrow F' \quad \{\,F'\,\}\,p\,\{\,G'\,\} \quad G' \Rightarrow G}{\{\,F\,\}\,p\,\{\,G\,\}}(\mathrm{lc})$ is admissible.

Logical consequence rule: strengthens pre- and weakens postconditions.

$\frac{F' \Rightarrow F \quad \{\,F'\,\}\,p\,\{\,G\,\}}{\{\,F\,\}\,p\,\{\,G\,\}}$ is not admissible.

Counterexample: $\frac{0 \Rightarrow 1 \text{ is valid} \quad \{\,0\,\}\,\mathbf{skip}\,\{\,0\,\} \text{ is true}}{\{\,1\,\}\,\mathbf{skip}\,\{\,0\,\} \text{ is false}}$

Weakest Precondition

$\mathrm{wp}(p, \mathcal{S}_{\text{out}})$ ... maximal set of states such that
$\mathrm{wp}(p, \mathcal{S}_{\text{out}})\ p\ \mathcal{S}_{\text{out}}$ is totally correct

$$\begin{aligned}\mathrm{wp}(p, \mathcal{S}_{\text{out}}) &= \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ defined and } [p]\,\sigma \in \mathcal{S}_{\text{out}} \,\} \\ &= [p]^{-1}(\mathcal{S}_{\text{out}})\end{aligned}$$

$\mathcal{S}_{\text{in}}\ p\ \mathcal{S}_{\text{out}}$ totally correct $\iff \mathcal{S}_{\text{in}} \subseteq \mathrm{wp}(p, \mathcal{S}_{\text{out}})$

$\{\, F \,\}\ p\ \{\, G \,\}$ totally correct $\iff \{\, F \,\} \subseteq \mathrm{wp}(p, \{\, G \,\})$
$F \Rightarrow \mathrm{wp}(p, G)$

$\mathrm{wp}(\textbf{skip}, G) = G$

wp can be used to define the semantics of TPL.
("axiomatic semantics" for total correctness)

Weakest Liberal Precondition

$\mathsf{wlp}(p, \mathcal{S}_{\text{out}})$ ... maximal set of states such that
$\mathsf{wlp}(p, \mathcal{S}_{\text{out}})\ p\ \mathcal{S}_{\text{out}}$ is partially correct

$$\begin{aligned}
\mathsf{wlp}(p, \mathcal{S}_{\text{out}}) &= \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ undefined or } [p]\,\sigma \in \mathcal{S}_{\text{out}} \,\} \\
&= \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ undefined} \,\} \cup [p]^{-1}(\mathcal{S}_{\text{out}}) \\
&= \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ undefined} \,\} \cup \mathsf{wp}(p, \mathcal{S}_{\text{out}})
\end{aligned}$$

$\mathcal{S}_{\text{in}}\ p\ \mathcal{S}_{\text{out}}$ partially correct $\iff \mathcal{S}_{\text{in}} \subseteq \mathsf{wlp}(p, \mathcal{S}_{\text{out}})$

$\{\, F \,\}\ p\ \{\, G \,\}$ partially correct $\iff \{\, F \,\} \subseteq \mathsf{wlp}(p, \{\, G \,\})$

$$F \Rightarrow \mathsf{wlp}(p,\ G\ )$$

$\mathsf{wlp}(\textbf{skip}, G) = G$

wlp can be used to define the semantics of TPL.
("axiomatic semantics" for partial correctness)

Strongest Postcondition

$\mathrm{sp}(\mathcal{S}_{\mathsf{in}}, p)$ ... minimal set of states such that
$\mathcal{S}_{\mathsf{in}}\ p\ \mathrm{sp}(\mathcal{S}_{\mathsf{in}}, p)$ is partially correct

$$\begin{aligned}\mathrm{sp}(\mathcal{S}_{\mathsf{in}}, p) &= \{\, [p]\,\sigma \mid \sigma \in \mathcal{S}_{\mathsf{in}} \text{ and } [p]\,\sigma \text{ defined} \,\} \\ &= [p](\mathcal{S}_{\mathsf{in}})\end{aligned}$$

$\mathcal{S}_{\mathsf{in}}\, p\, \mathcal{S}_{\mathsf{out}}$ partially correct $\iff \mathrm{sp}(\mathcal{S}_{\mathsf{in}}, p) \subseteq \mathcal{S}_{\mathsf{out}}$

$\{\, F \,\}\, p\, \{\, G \,\}$ partially correct $\iff \mathrm{sp}(\{\, F \,\}, p) \subseteq \{\, G \,\}$
$\mathrm{sp}(\ \ F\ \ , p) \Rightarrow\ \ G$

$\mathrm{sp}(F, \textbf{skip}) = F$

sp can be used to define the semantics of TPL.
("axiomatic semantics" for partial correctness)

Verification of **skip** statements

Hoare calculus

$\{ F \}\, \mathbf{skip}\, \{ F \}$ (sk) is an admissible axiom.

Unless stated otherwise, the axioms and rules are admissible for partial and total correctness.

Weakest (liberal) precondition

$$\mathrm{wp}(\mathbf{skip}, G) = \mathrm{wlp}(\mathbf{skip}, G) = G$$

Strongest postcondition

$$\mathrm{sp}(F, \mathbf{skip}) = F$$

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Assignment – Forward Reasoning (1)

$\{ F \}\, v := e \,\{ ? \}$

$\{ x = 2 \}\, y := 3 \,\{ x = 2 \wedge y = 3 \}$ ✓

Let's generalise this observation:
$\{ F \}\, v := e \,\{ F \wedge v = e \}$

$\{ x = 2 \}\, x := 3 \,\{ x = 2 \wedge x = 3 \}$???
$\Rightarrow$ v must not occur in F!

$\{ x = 2 \}\, y := y + 1 \,\{ x = 2 \wedge y = y + 1 \}$???
$\Rightarrow$ v must not occur in e!

$\{ F \}\, v := e \,\{ F \wedge v = e \}$ . . . axiom, if v occurs neither in F nor in e.

Useful for initialising local variables.
But how to verify the other assignments?

Assignment – Forward Reasoning (2)

$$\{\, x = 2 \,\} \qquad x := x + 5 \qquad \{\, x - 5 = 2 \,\}$$

If $\sigma(x) = 2$ and $\sigma'(x) = \sigma(x) + 5$ then $\sigma'(x) - 5 = 2$.

$$\sigma'(x) - 5 = \sigma(x)$$

Let's generalise this observation:

$$\{\, F \,\}\, v := e(v)\, \{\, F[v/e^{-1}(v)] \,\}$$

e^{-1} ... 'inverse' of expression e

$F[v/e^{-1}(v)]$ ... formula F with every v replaced by $e^{-1}(v)$

$\{\, y = 4x \,\}\, x := 2x\, \{\, y = 4(x/2) \,\}$ $\quad e(x) = 2x \quad e^{-1}(x) = x/2$

$\{\, y = 4x \,\}\, x := 2x\, \{\, y = 2x \,\}$ ✓

$\{\, y = 4x \,\}\, x := x/2\, \{ \qquad \}$

What is the inverse of $x/2$? Is it $2x$? Or $2x + 1$?

What if the inverse cannot be expressed concisely by an expression?

What is the inverse of a constant?

Assignment – Forward Reasoning (3)

$\{ F \}\, v := e \,\{ ? \}$

What do we know about the state after the assignment?

- There is an old value of v (now lost). $\exists v'$
- F is true, but for the old value of v. $F[v/v']$
- The new value of v equals the value of e (but evaluated with old value of v). $v = e[v/v']$

$\{ F \}\, v := e \,\{ \exists v' \,(F[v/v'] \wedge v = e[v/v']) \}$ ... axiom
(v' is a fresh variable not occurring in F and e)

$\{ y = 4x \}\, x := 2x \,\{ \exists x' \,(y = 4x' \wedge x = 2x') \}$
$\exists x' \,(y = 2x \wedge x = 2x')$
$y = 2x \wedge \exists x' \, x = 2x'$
$y = 2x \wedge x$ is even

$\{ y = 4x \}\, x := x/2 \,\{ \exists x' \,(y = 4x' \wedge x = x'/2) \}$

Assignment – Backward Reasoning

$$\{\, x + 5 = 7 \,\} \qquad x := x + 5 \qquad \{\, x = 7 \,\}$$

If $\sigma(x) + 5 = 7$ and $\sigma'(x) = \sigma(x) + 5$ then $\sigma'(x) = 7$.

Let's generalise this observation:
$\{\, F[v/e] \,\}\, v := e \,\{\, F \,\}$

$F[v/e]$ ... formula F, where every v has been replaced by e
No inverse of e needed!

$\{\, y = 2(2x) \,\}\, x := 2x \,\{\, y = 2x \,\}$ ✓

$\{\, y = 8(x/2) \,\}\, x := x/2 \,\{\, y = 8x \,\}$ ✓

$$\frac{x = 2 \Rightarrow 3 = 3 \quad \{\, 3 = 3 \,\}\, x := 3 \,\{\, x = 3 \,\}}{\{\, x = 2 \,\}\, x := 3 \,\{\, x = 3 \,\}}\ (\mathrm{lc})$$ ✓

$\{\, F[v/e] \,\}\, v := e \,\{\, F \,\}$ ... axiom

Verification of assignments

Hoare calculus

$\{ G[v/e] \}\, v := e \,\{ G \}$ (as) is an admissible axiom.

$\{ F \}\, v := e \,\{ \exists v' \, (F[v/v'] \wedge v = e[v/v']) \}$ (as′) is an admissible axiom provided v' does not occur in F and e.

$\{ F \}\, v := e \,\{ F \wedge v = e \}$ (as″) is an admissible axiom provided v does not occur in F and e.

Weakest (liberal) precondition

$\mathsf{wp}(v := e, G) = \mathsf{wlp}(v := e, G) = G[v/e]$

Strongest postcondition

$\mathsf{sp}(F, v := e) = \exists v' \, (F[v/v'] \wedge v = e[v/v'])$
where v' does not occur in F and e.

Forward vs. Backward Reasoning (1)

Precondition: Weak condition on input variables, like $x > 0$.

Postcondition: Complex formula describing the property we are interested in, like $z = x * y_0$. Ignores auxiliary variables.

Forward reasoning: Given a precondition F and a program p, compute a (strongest) postcondition G' and show that it implies the desired property G.

$$\frac{\{\, F \,\}\, p \,\{\, G' \,\} \quad G' \Rightarrow G}{\{\, F \,\}\, p \,\{\, G \,\}}\ (\mathsf{lc})$$

Disadvantage: For deriving G' we do not use the postcondition. Hence G' usually specifies much more of the program's behaviour than is really needed to prove G.

Forward vs. Backward Reasoning (2)

Backward reasoning: Given a program property G and a program p, compute a (weakest) precondition F' and show that it is implied by the given precondition F.

$$\frac{F \Rightarrow F' \quad \{\, F' \,\}\, p \,\{\, G \,\}}{\{\, F \,\}\, p \,\{\, G \,\}}\ \text{(lc)}$$

Alternatively, given the postcondition G and the program p, characterise the input data for which the program works.

$$\{\, ??? \,\}\, p \,\{\, G \,\} \Rightarrow \{\, F' \,\}\, p \,\{\, G \,\}$$

Advantages:

- Goal-oriented: We only care about variables and properties necessary for proving the postcondition.
- Meshes well with the assignment axiom (as) and the weakest precondition of assignments.

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Verifying sequential compositions – Hoare calculus

Suppose we know that

- $\{ F \}\, p \,\{ G \}$ is (partially or totally) correct,
  i.e., executing p in an F-state yields a G-state, and
- $\{ G \}\, q \,\{ H \}$ is (partially or totally) correct,
  i.e., executing q in a G-state yields an H-state.

Then we have that

- $\{ F \}\, p; q \,\{ H \}$ is (partially or totally) correct,
  i.e., executing p followed by q in an F-state yields an H-state.

$$\frac{\{ F \}\, p \,\{ G \} \quad \{ G \}\, q \,\{ H \}}{\{ F \}\, p; q \,\{ H \}}\ (\mathrm{sc}) \quad \text{is an admissible rule.}$$

Problem: How can we determine the interpolant G?

Example: Swapping values (Hoare calculus)

$$\dfrac{\dfrac{F \Rightarrow G[x/y] \quad \{\, G[x/y] \,\}\, x := y \,\{\, G \,\}}{\{\, F \,\}\, x := y \,\{\, G \,\}}\,(\mathrm{lc}) \qquad \dfrac{G \Rightarrow Q[y/t] \quad \{\, Q[y/t] \,\}\, y := t \,\{\, Q \,\}}{\{\, G \,\}\, y := t \,\{\, Q \,\}}\,(\mathrm{lc})}{\{\, F \,\}\, x := y; y := t \,\{\, Q \,\}}\,(\mathrm{sc})$$

(as) (as)

$$\dfrac{\dfrac{P \Rightarrow F[t/x] \quad \{\, F[t/x] \,\}\, t := x \,\{\, F \,\}}{\{\, P \,\}\, t := x \,\{\, F \,\}}\,(\mathrm{lc}) \qquad \{\, F \,\}\, x := y; y := t \,\{\, Q \,\}}{\{\, P \colon x = x_0 \wedge y = y_0 \,\}\, t := x; x := y; y := t \,\{\, Q \colon x = y_0 \wedge y = x_0 \,\}}\,(\mathrm{sc})$$

(as)

To do: Find F and G such that the formulas $P \Rightarrow F[t/x]$, $F \Rightarrow G[x/y]$, and $G \Rightarrow Q[y/t]$ become valid.

We choose $F \equiv G[x/y]$ and $G \equiv Q[y/t]$.

It remains to prove $P \Rightarrow F[t/x]$, i.e., $P \Rightarrow Q[y/t][x/y][t/x]$.

$$P \Rightarrow Q[y/t][x/y][t/x]$$

$$(x = x_0 \wedge y = y_0) \Rightarrow (x = y_0 \wedge y = x_0)[y/t][x/y][t/x]$$

$$(x = x_0 \wedge y = y_0) \Rightarrow (x = y_0 \wedge t = x_0)[x/y][t/x]$$

$$(x = x_0 \wedge y = y_0) \Rightarrow (y = y_0 \wedge t = x_0)[t/x]$$

$$(x = x_0 \wedge y = y_0) \Rightarrow (y = y_0 \wedge x = x_0)$$

This implication is valid, therefore the assertion

$$\{\, P \,\}\, t := x; x := y; y := t \,\{\, Q \,\}$$

is partially/totally correct.

Example: Swapping values (Hoare simplified)

- Decompose sequential compositions in reverse order.
- "Guess" suitable interpolants leading directly to axioms.

$$\dfrac{\dfrac{\dfrac{P \Rightarrow F[t/x] \quad \{\, F[t/x] \,\}\, t := x \,\{\, F \,\}}{\{\, P \,\}\, t := x \,\{\, F \,\}}\ (\mathsf{lc}) \qquad \{\, F \,\}\, x := y \,\{\, G \,\}}{\{\, P \,\}\, t := x; x := y \,\{\, G \,\}}\ (\mathsf{sc}) \qquad \{\, G \,\}\, y := t \,\{\, Q \,\}}{\{\, P : x = x_0 \wedge y = y_0 \,\}\, t := x; x := y; y := t \,\{\, Q : x = y_0 \wedge y = x_0 \,\}}\ (\mathsf{sc})$$

(as) over $P \Rightarrow F[t/x]$ and $\{\, F[t/x] \,\}\, t := x \,\{\, F \,\}$; Choose $F = G[x/y]$; (as) over $\{\, F \,\}\, x := y \,\{\, G \,\}$; Choose $G = Q[y/t]$; (as) over $\{\, G \,\}\, y := t \,\{\, Q \,\}$.

- Prove $P \Rightarrow F[t/x]$

  $P \Rightarrow G[x/y][t/x]$

  $P \Rightarrow Q[y/t][x/y][t/x]$

  (Already done above.)

Still: 3 assignments in the program, but 9 in the proof.

Verifying sequential compositions – wp, wlp, and sp

Weakest (liberal) precondition

$$\mathrm{wp}(p; q, G) = \mathrm{wp}(p, \mathrm{wp}(q, G))$$
$$\mathrm{wlp}(p; q, G) = \mathrm{wlp}(p, \mathrm{wlp}(q, G))$$

- $\mathrm{wp}(q, G)$ is the maximal set of states such that q yields an G-state.
- $\mathrm{wp}(p, \mathrm{wp}(q, G))$ is the maximal set of states such that p yields a $\mathrm{wp}(q, G)$-state.

Strongest postcondition

$$\mathrm{sp}(F, p; q) = \mathrm{sp}(\mathrm{sp}(F, p), q)$$

- $\mathrm{sp}(F, p)$ is the minimal set of states obtained by executing p in an F-state.
- $\mathrm{sp}(\mathrm{sp}(F, p), q)$ is the minimal set of states obtained by executing q in an $\mathrm{sp}(F, p)$-state.

Example: Swapping values (proof by wp)

To show: $\{ P \}\, t := x; x := y; y := t \,\{ Q \}$ is totally correct, where

P: $x = x_0 \wedge y = y_0$
Q: $x = y_0 \wedge y = x_0$

$$\begin{aligned}
\mathrm{wp}(t := x; x := y; y := t,\ Q) &= \mathrm{wp}(t := x; x := y,\ \mathrm{wp}(y := t,\ Q)) \\
&= \mathrm{wp}(t := x; x := y,\ Q[y/t]) \\
&= \mathrm{wp}(t := x,\ \mathrm{wp}(x := y,\ Q[y/t])) \\
&= \mathrm{wp}(t := x,\ Q[y/t][x/y]) \\
&= Q[y/t][x/y][t/x]
\end{aligned}$$

It remains to prove $P \Rightarrow Q[y/t][x/y][t/x]$. (Already done above.)

$\{ F \}\, p \,\{ G \}$ totally correct iff $F \Rightarrow \mathrm{wp}(p, G)$.

$\mathrm{wp}(v := e, G) = G[v/e]$
$\mathrm{wp}(p; q,\ G) = \mathrm{wp}(p,\ \mathrm{wp}(q,\ G))$

Example: Swapping values (proof by sp)

To show: $\{ P \}\, t := x; x := y; y := t \,\{ Q \}$ is partially correct, where

P: $x = x_0 \wedge y = y_0$
Q: $x = y_0 \wedge y = x_0$

$$
\begin{aligned}
&\mathsf{sp}(P,\ t := x; x := y; y := t) \\
&= \mathsf{sp}(\mathsf{sp}(P,\ t := x),\ x := y; y := t) \\
&= \mathsf{sp}(\exists t' \,(P[t/t'] \wedge t = x[t/t']),\ x := y; y := t) \\
&= \mathsf{sp}(\exists t' \,(P \wedge t = x),\ x := y; y := t) \\
&= \mathsf{sp}(P \wedge t = x,\ x := y; y := t)
\end{aligned}
$$

$\{ F \}\, p \,\{ G \}$ partially correct iff $\mathsf{sp}(F, p) \Rightarrow G$.

$\mathsf{sp}(F, v := e) = \exists v' \,(F[v/v'] \wedge v = e[v/v'])$
$\mathsf{sp}(F,\ p; q) = \mathsf{sp}(\mathsf{sp}(F,\ p),\ q)$

Example: Swapping values (proof by sp)

To show: $\{ P \}\, t := x; x := y; y := t \,\{ Q \}$ is partially correct, where

P: $x = x_0 \wedge y = y_0$
Q: $x = y_0 \wedge y = x_0$

$$
\begin{aligned}
&\mathrm{sp}(P,\ t := x; x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P,\ t := x),\ x := y; y := t) \\
&= \mathrm{sp}(P \wedge t = x,\ x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P \wedge t = x,\ x := y),\ y := t) \\
&= \mathrm{sp}(\exists x' \,((P \wedge t = x)[x/x'] \wedge x = y[x/x']),\ y := t) \\
&= \mathrm{sp}(\exists x' \,(x' = x_0 \wedge y = y_0 \wedge t = x' \wedge x = y),\ y := t) \\
&= \mathrm{sp}(\exists x' \,(x' = x_0 \wedge y = y_0 \wedge t = x_0 \wedge x = y),\ y := t) \\
&= \mathrm{sp}(y = y_0 \wedge t = x_0 \wedge x = y \wedge \exists x' \,(x' = x_0),\ y := t) \\
&= \mathrm{sp}(y = y_0 \wedge t = x_0 \wedge x = y,\ y := t)
\end{aligned}
$$

$\{ F \}\, p \,\{ G \}$ partially correct iff $\mathrm{sp}(F, p) \Rightarrow G$.

$\mathrm{sp}(F, v := e) = \exists v' \,(F[v/v'] \wedge v = e[v/v'])$
$\mathrm{sp}(F,\ p; q) = \mathrm{sp}(\mathrm{sp}(F,\ p),\ q)$

Example: Swapping values (proof by sp)

To show: $\{ P \}\, t := x; x := y; y := t \,\{ Q \}$ is partially correct, where

P: $x = x_0 \wedge y = y_0$
Q: $x = y_0 \wedge y = x_0$

$$
\begin{aligned}
&\mathrm{sp}(P,\ t := x; x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P,\ t := x),\ x := y; y := t) \\
&= \mathrm{sp}(P \wedge t = x,\ x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P \wedge t = x,\ x := y),\ y := t) \\
&= \mathrm{sp}(y = y_0 \wedge t = x_0 \wedge x = y,\ y := t) \\
&= \exists y' \,((y = y_0 \wedge t = x_0 \wedge x = y)[y/y'] \wedge y = t[y/y']) \\
&= \exists y' \,(y' = y_0 \wedge t = x_0 \wedge x = y' \wedge y = t) \\
&= \exists y' \,(y' = y_0 \wedge t = x_0 \wedge x = y_0 \wedge y = t) \\
&= (t = x_0 \wedge x = y_0 \wedge y = t \wedge \exists y' \,(y' = y_0)) \\
&= (t = x_0 \wedge x = y_0 \wedge y = t)
\end{aligned}
$$

$\{ F \}\, p \,\{ G \}$ partially correct iff $\mathrm{sp}(F, p) \Rightarrow G$.

$\mathrm{sp}(F, v := e) = \exists v' \,(F[v/v'] \wedge v = e[v/v'])$
$\mathrm{sp}(F,\ p; q) = \mathrm{sp}(\mathrm{sp}(F,\ p),\ q)$

Example: Swapping values (proof by sp)

To show: $\{ P \}\, t := x; x := y; y := t \,\{ Q \}$ is partially correct, where

P: $x = x_0 \wedge y = y_0$
Q: $x = y_0 \wedge y = x_0$

$$
\begin{aligned}
&\mathrm{sp}(P,\ t := x; x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P,\ t := x),\ x := y; y := t) \\
&= \mathrm{sp}(P \wedge t = x,\ x := y; y := t) \\
&= \mathrm{sp}(\mathrm{sp}(P \wedge t = x,\ x := y),\ y := t) \\
&= \mathrm{sp}(y = y_0 \wedge t = x_0 \wedge x = y,\ y := t) \\
&= (t = x_0 \wedge x = y_0 \wedge y = t)
\end{aligned}
$$

It remains to prove $(t = x_0 \wedge x = y_0 \wedge y = t) \Rightarrow Q$

$\{ F \}\, p \,\{ G \}$ partially correct iff $\mathrm{sp}(F, p) \Rightarrow G$.

$\mathrm{sp}(F, v := e) = \exists v' \, (F[v/v'] \wedge v = e[v/v'])$
$\mathrm{sp}(F,\ p; q) = \mathrm{sp}(\mathrm{sp}(F,\ p),\ q)$

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Hoare calculus:

- clear separation between premises and conclusions
- cumbersome for manual proofs:
  programs and formulas have to be copied many times

wp/wlp/sp:

- recursive evaluation leads to duplications

Idea:

- use only one copy of program
- add assertions to the program step by step,
  which characterise the states at each point
- collect all useful ideas from wp, wlp, sp, and Hoare calculus

$\Rightarrow$ Annotation calculus

Nothing new, just a practical tool!

Example: Swapping values (proof by annotations)

$$\{\, P\colon x = x_0 \wedge y = y_0 \,\}$$
$$\{\, 3\colon Q[y/t][x/y][t/x] \,\} \quad \text{as} \uparrow$$
$$t := x;$$
$$\{\, 2\colon Q[y/t][x/y] \,\} \quad \text{as} \uparrow$$
$$x := y;$$
$$\{\, 1\colon Q[y/t] \,\} \quad \text{as} \uparrow$$
$$y := t$$
$$\{\, Q\colon x = y_0 \wedge y = x_0 \,\}$$

It remains to prove the implication

$$P \Rightarrow 3\colon \quad (x = x_0 \wedge y = y_0) \Rightarrow Q[y/t][x/y][t/x]$$

Annotation rules so far

Assignments

$$
\begin{array}{lcl}
 & & \{\, F[v/e] \,\} \quad \text{as} \uparrow \\
v := e & \mapsto & v := e \\
\{\, F \,\} & & \{\, F \,\}
\end{array}
$$

$$
\begin{array}{lcl}
\{\, F \,\} & & \{\, F \,\} \\
v := e & \mapsto & v := e \\
 & & \{\, \exists v'(F[v/v'] \wedge v = e[v/v']) \,\} \quad \text{as} \downarrow
\end{array}
$$

$$
\begin{array}{lcll}
\{\, F \,\} & & \{\, F \,\} & \\
v := e & \mapsto & v := e & \text{(if } v \text{ does not occur free in } F \text{ and } e\text{)} \\
 & & \{\, F \wedge v = e \,\} \quad \text{as}' \downarrow &
\end{array}
$$

Annotation rules so far

Skip statements

$$\begin{array}{lcl} & & \{\,F\,\} \quad \mathsf{sk}\uparrow \\ \textbf{skip} & \mapsto & \textbf{skip} \\ \{\,F\,\} & & \{\,F\,\} \end{array} \qquad \begin{array}{lcl} \{\,F\,\} & & \{\,F\,\} \\ \textbf{skip} & \mapsto & \textbf{skip} \\ & & \{\,F\,\} \quad \mathsf{sk}\downarrow \end{array}$$

Logical consequence

$$\begin{array}{l} \{\,F\,\} \\ \{\,G\,\} \end{array} \quad \mapsto \quad \begin{array}{l} \{\,F\,\} \\ \{\,G\,\} \end{array} \quad \text{Prove } F \Rightarrow G \quad \mathsf{lc}$$

Sequential composition

$$\begin{array}{lcl} & & \{\,F\,\} \quad \mathsf{sc}\uparrow \\ ; & \mapsto & ; \\ \{\,F\,\} & & \{\,F\,\} \end{array} \qquad \begin{array}{lcl} \{\,F\,\} & & \{\,F\,\} \\ ; & \mapsto & ; \\ & & \{\,F\,\} \quad \mathsf{sc}\downarrow \end{array}$$

These rules are applied implicitly.

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Conditionals (1)

Hoare calculus

$$\frac{\{\,F \wedge e\,\}\,p\,\{\,G\,\} \quad \{\,F \wedge \neg e\,\}\,q\,\{\,G\,\}}{\{\,F\,\}\,\textbf{if } e \textbf{ then } p \textbf{ else } q \textbf{ fi}\,\{\,G\,\}}\ \text{(if)}$$

is an admissible rule.

Annotation rules

$$\begin{array}{l} \{\,F\,\} \\ \textbf{if } e \textbf{ then} \\ \quad \ldots \\ \textbf{else} \end{array} \quad \mapsto \quad \begin{array}{l} \{\,F\,\} \\ \textbf{if } e \textbf{ then} \\ \quad \{\,F \wedge e\,\} \quad \text{if}\downarrow \\ \quad \ldots \\ \textbf{else} \\ \quad \{\,F \wedge \neg e\,\} \quad \text{if}\downarrow \end{array}$$

$$\begin{array}{l} \textbf{else} \\ \quad \ldots \\ \textbf{fi} \\ \{\,G\,\} \end{array} \quad \mapsto \quad \begin{array}{l} \textbf{else} \\ \quad \ldots \\ \quad \{\,G\,\} \quad \text{fi}\uparrow \\ \textbf{fi} \\ \{\,G\,\} \end{array}$$

What about rules "if $\uparrow$" and "fi $\downarrow$" as duals of "if $\downarrow$" and "fi $\uparrow$"?

Example: Maximum (1)

$\{\,\mathsf{true}\,\}$
if $x \leq y$ **then**
  $\{\,\mathsf{true} \wedge x \leq y\,\}$ if $\downarrow$
  $\{\,y = \max(x, y)\,\}$ as $\uparrow$
  $z := y$
  $\{\,z = \max(x, y)\,\}$ fi $\uparrow$
else
  $\{\,\mathsf{true} \wedge x \not\leq y\,\}$ if $\downarrow$
  $\{\,x = \max(x, y)\,\}$ as $\uparrow$
  $z := x$
  $\{\,z = \max(x, y)\,\}$ fi $\uparrow$
fi
$\{\,z = \max(x, y)\,\}$

It remains to prove

$\mathsf{true} \wedge x \leq y$
  $\Rightarrow y = \max(x, y)$ lc

and

$\mathsf{true} \wedge x \not\leq y$
  $\Rightarrow x = \max(x, y)$ lc

Conditionals (2)

Strongest postcondition

$$\mathrm{sp}(F, \textbf{if } e \textbf{ then } p \textbf{ else } q \textbf{ fi}) = \mathrm{sp}(F \wedge e,\ p) \vee \mathrm{sp}(F \wedge \neg e,\ q)$$

Annotation rules

$$\begin{array}{l} \{\, F \,\} \\ \textbf{if } e \textbf{ then} \\ \quad \ldots \\ \textbf{else} \end{array} \quad \mapsto \quad \begin{array}{l} \{\, F \,\} \\ \textbf{if } e \textbf{ then} \\ \quad \{\, F \wedge e \,\} \quad \mathrm{if}\downarrow \\ \quad \ldots \\ \textbf{else} \\ \quad \{\, F \wedge \neg e \,\} \quad \mathrm{if}\downarrow \end{array}$$

$$\begin{array}{l} \{\, G_1 \,\} \\ \textbf{else} \\ \quad \ldots \\ \quad \{\, G_2 \,\} \\ \textbf{fi} \end{array} \quad \mapsto \quad \begin{array}{l} \{\, G_1 \,\} \\ \textbf{else} \\ \quad \ldots \\ \quad \{\, G_2 \,\} \\ \textbf{fi} \\ \{\, G_1 \vee G_2 \,\} \quad \mathrm{fi}\downarrow \end{array}$$

Conditionals (3)

Hoare calculus (an alternative)

$$\frac{\{\,F\,\}\,p\,\{\,H\,\} \quad \{\,G\,\}\,q\,\{\,H\,\}}{\{\,(e \Rightarrow F) \wedge (\neg e \Rightarrow G)\,\}\,\textbf{if}\ e\ \textbf{then}\ p\ \textbf{else}\ q\ \textbf{fi}\,\{\,H\,\}}\ (\text{if}')$$ is an admissible rule.

Weakest (liberal) precondition

$$\text{wp}(\textbf{if}\ e\ \textbf{then}\ p\ \textbf{else}\ q\ \textbf{fi}, H) = (e \Rightarrow \text{wp}(p, H)) \wedge (\neg e \Rightarrow \text{wp}(q, H))$$
$$\text{wlp}(\textbf{if}\ e\ \textbf{then}\ p\ \textbf{else}\ q\ \textbf{fi}, H) = (e \Rightarrow \text{wlp}(p, H)) \wedge (\neg e \Rightarrow \text{wlp}(q, H))$$

Annotation rules

$$\begin{array}{l} \textbf{if}\ e\ \textbf{then} \\ \quad \{\,F\,\} \\ \quad \ldots \\ \textbf{else} \\ \quad \{\,G\,\} \end{array} \quad \mapsto \quad \begin{array}{l} \{\,(e \Rightarrow F) \wedge (\neg e \Rightarrow G)\,\} \quad \text{if}\uparrow \\ \textbf{if}\ e\ \textbf{then} \\ \quad \{\,F\,\} \\ \quad \ldots \\ \textbf{else} \\ \quad \{\,G\,\} \end{array}$$

$$\begin{array}{l} \textbf{else} \\ \quad \ldots \\ \textbf{fi} \\ \{\,G\,\} \end{array} \quad \mapsto \quad \begin{array}{l} \textbf{else} \\ \quad \ldots \\ \quad \{\,G\,\} \quad \text{fi}\uparrow \\ \textbf{fi} \\ \{\,G\,\} \end{array}$$

Note: $(e \Rightarrow F) \wedge (\neg e \Rightarrow G)$ is logically equivalent to $(e \wedge F) \vee (\neg e \wedge G)$.

Example: Maximum (2)

$\{\,\mathsf{true}\,\}$
$\{\,(x \le y \Rightarrow y = \max(x,y)) \wedge (x \not\le y \Rightarrow x = \max(x,y))\,\}$ if $\uparrow$
if $x \le y$ **then**
  $\{\,y = \max(x,y)\,\}$ as $\uparrow$
  $z := y$
  $\{\,z = \max(x,y)\,\}$ fi $\uparrow$
else
  $\{\,x = \max(x,y)\,\}$ as $\uparrow$
  $z := x$
  $\{\,z = \max(x,y)\,\}$ fi $\uparrow$
fi
$\{\,z = \max(x,y)\,\}$

It remains to prove

$$\mathsf{true} \Rightarrow ((x \le y \Rightarrow y = \max(x,y)) \wedge (x \not\le y \Rightarrow x = \max(x,y)))$$

(logic consequence rule).

$$\text{true} \Rightarrow (x \leq y \Rightarrow y = \max(x, y)) \wedge (x \not\leq y \Rightarrow x = \max(x, y))$$

is of the form

$$A \Rightarrow ((B \Rightarrow C) \wedge (D \Rightarrow E))$$

which can be rewritten as

$$(A \Rightarrow (B \Rightarrow C)) \wedge$$
$$(A \Rightarrow (D \Rightarrow E))$$

$$(A \wedge B \Rightarrow C) \wedge$$
$$(A \wedge D \Rightarrow E)$$

Hence proving the single implication above is the same as proving the two implications from before.

Topics today

1. Why formal methods?
2. Syntax of TPL(toy programming language)
3. Operational Semantics of TPL
4. Correctness assertions
5. How to prove correctness assertions
6. Assignments
7. Sequential Composition
8. Annotation calculus
9. Conditionals
10. Abort statements

Abort-Statement – Partial Correctness

$\{ F \}\, \textbf{abort}\, \{ G \}$ being partially correct means:

- For all states σ,
  If $[F]\,\sigma$ is true and $[\textbf{abort}]\,\sigma$ is defined, then $[G]\,[\textbf{abort}]\,\sigma$ is true.
- Since $[\textbf{abort}]\,\sigma$ is undefined for all $\sigma \in \mathcal{S}$,
  the implication is always true, regardless of F and G.

Hoare calculus

$\{ F \}\, \textbf{abort}\, \{ G \}$ (ab) is an axiom admissible for partial correctness.

Weakest liberal precondition

$\text{wlp}(\textbf{abort}, G) = \text{true}$

Strongest postcondition

$\text{sp}(F, \textbf{abort}) = \text{false}$

Annotation rule

$$\textbf{abort} \quad \mapsto \quad \begin{array}{l} \{\, \text{true} \,\} \quad \text{ab} \\ \textbf{abort} \\ \{\, \text{false} \,\} \quad \text{ab} \end{array}$$

Abort-Statement – Total Correctness

$\{\, F \,\}\, \textbf{abort}\, \{\, G \,\}$ being totally correct means:

- For all states σ,
  If $[F]\,\sigma$ is true, then $[\textbf{abort}]\,\sigma$ is defined and $[G]\,[\textbf{abort}]\,\sigma$ is true.
- Since $[\textbf{abort}]\,\sigma$ is always undefined,
  the implication is true only if $[F]\,\sigma$ is always false.

Hoare calculus

$\{\, \text{false} \,\}\, \textbf{abort}\, \{\, G \,\}$ (abt) is an admissible axiom.

Weakest precondition

$\text{wp}(\textbf{abort}, G) = \text{false}$

Annotation rule

$$\textbf{abort} \quad \mapsto \quad \begin{array}{ll} \{\, \text{false} \,\} & \text{abt} \\ \textbf{abort} & \\ \{\, \text{false} \,\} & \text{abt} \end{array}$$

wp(**if** $x \geq 0$ **then skip else abort fi**, $x > 2$)

$= (x \geq 0 \wedge \mathsf{wp}(\textbf{skip}, x > 2)) \vee (x < 0 \wedge \mathsf{wp}(\textbf{abort}, x > 2)$
$= (x \geq 0 \wedge x > 2) \vee (x < 0 \wedge \mathsf{false})$
$= x > 2 \vee \mathsf{false}$
$= x > 2$

wlp(**if** $x \geq 0$ **then skip else abort fi**, $x > 2$)

$= (x \geq 0 \wedge \mathsf{wlp}(\textbf{skip}, x > 2)) \vee (x < 0 \wedge \mathsf{wlp}(\textbf{abort}, x > 2))$
$= (x \geq 0 \wedge x > 2) \vee (x < 0 \wedge \mathsf{true})$
$= x > 2 \vee x < 0$

sp(true, **if** $x \geq 0$ **then skip else abort fi**)

$= \mathsf{sp}(\mathsf{true} \wedge x \geq 0, \textbf{skip}) \vee \mathsf{sp}(\mathsf{true} \wedge x < 0, \textbf{abort})$
$= (\mathsf{true} \wedge x \geq 0) \vee \mathsf{false}$
$= x \geq 0$

$\{ x > 2 \}$ **if** $x \geq 0$ **then skip else abort fi** $\{ x > 2 \}$ is totally correct.

$\{ x > 2 \}$
if $x \geq 0$ **then**
  $\{ x > 2 \wedge x \geq 0 \}$ if$\downarrow$
  $\{ x > 2 \}$ sk$\uparrow$
  skip
  $\{ x > 2 \}$ fi$\uparrow$
else
  $\{ x > 2 \wedge x \not\geq 0 \}$ if$\downarrow$
  $\{ \text{false} \}$ abt
  abort
  $\{ \text{false} \}$ abt
  $\{ x > 2 \}$ fi$\uparrow$
fi
$\{ x > 2 \}$

It remains to prove

$x > 2 \wedge x \geq 0$
  $\Rightarrow x > 2$ lc

and

$x > 2 \wedge x \not\geq 0$
  $\Rightarrow$ false lc

and

false
  $\Rightarrow x > 2$ lc

$\{ x > 2 \}$ **if** $x \geq 0$ **then skip else abort fi** $\{ x > 2 \}$ is partially correct.

$\{ x > 2 \}$
if $x \geq 0$ **then**
  $\{ x > 2 \wedge x \geq 0 \}$ if$\downarrow$
  $\{ x > 2 \}$ sk$\uparrow$
  skip
  $\{ x > 2 \}$ fi$\uparrow$
else
  $\{ x > 2 \wedge x \not\geq 0 \}$ if$\downarrow$
  $\{ \text{true} \}$ ab
  abort
  $\{ \text{false} \}$ ab
  $\{ x > 2 \}$ fi$\uparrow$
fi
$\{ x > 2 \}$

It remains to prove

$x > 2 \wedge x \geq 0$
  $\Rightarrow x > 2$ lc

and

$x > 2 \wedge x \not\geq 0$
  $\Rightarrow$ true lc

and

false
  $\Rightarrow x > 2$ lc