Exam-1 Program Verification 2019/2020

BBG-169, 7th Oct 2019, 8:30 - 10:30

Lecturers: Wishnu Prasetya, Anna-Lena Lamprecht

1 Predicate Transformer [2pt]

1. [0.6 pt] Give the weakest pre-conditions of the following statements. **Show** your calculation.

   (a) $\{*\ ?\ *\}$ **if** $\texttt{x}{\geq}\texttt{10}$ **then** $\{\texttt{x} := \texttt{x} - \texttt{10}\ ;\ \texttt{y} := \texttt{y} * \texttt{x}\}$ **else** $\texttt{y} := -\texttt{1}$ $\{*\ \texttt{y} > \texttt{0}\ *\}$

   (b) $\{*\ ?\ *\}$ $a[i{-}1] := a[0] + 1\ ;\ a[0] := 0$ $\{*\ \texttt{a[i}{-}\texttt{1]} \geq \texttt{a[0]}\ *\}$
   Do reduce/remove all **repby** sub-expressions from your result.

2. [0.3 pt] Suppose we have a programming language that has a concept of statements, for which we can define Hoare triples: if S is a statement of this programming language, $\{P\}\ S\ \{Q\}$ means that if we execute S on a state satisfying the pre-condition P, and if the statement terminates, it will terminate in a state satisfying Q.

   Suppose we define how to calculate **wlp** over the statements of this programming language, and claim that this **wlp** is <u>complete</u>. What does this mean?

3. [0.5 pt] The calculation of **wlp** over $a[e_1] := e_2$ as defined in the PV Lecture Notes assumes that the array a is of infinite size. Suppose we now want to reason over realistic arrays that must have a finite size, and moreover a should not be null. Propose how to calculate the **wlp** of such an assignment when a is a realistic array.

   You can assume that e_1 and e_2 do not crash/throw any exception.

4. [0.5 pt] Suppose we want to have N-dimensional arrays, $N{>}0$. E.g. we can now write $a[0][1][0]$. Give a rule to calculate the **wlp** over an assignmentr $target := expr$ where the $target$ is a 3-dimensional array expression. For example, an assignment such as $a[0][1][0] := 0$.

   You can assume that arrays have infinite size.

2 Dealing with Loop [2 pt]

1. [1.8 pt] Consider the specifications of the programs below. Give for each a loop-invariant that would be good enough to prove the validity of the corresponding specification. It should be an invariant that is consistent, strong enough to realize the asked post-condition, and realistic to be established by the pre-condition or initialization of the loop. Use the *partial* correctness interpretation of Hoare triples.

   Unless stated otherwise, all variables are of type int.

   (a)
   $$\{*\ \mathtt{x}{\in}\{\mathtt{1,2,3}\}\ *\}$$
   $$\textbf{while}\ \mathtt{x{>}0}\ \textbf{do}\ \{\ \textbf{if}\ \mathtt{x{=}1}\ \textbf{then}\ \mathtt{x} := \mathtt{x{+}1}\ \textbf{else}\ \mathtt{x} := \mathtt{x{-}2}\ \}$$
   $$\{*\ \mathtt{x} = \mathtt{0}\ *\}$$

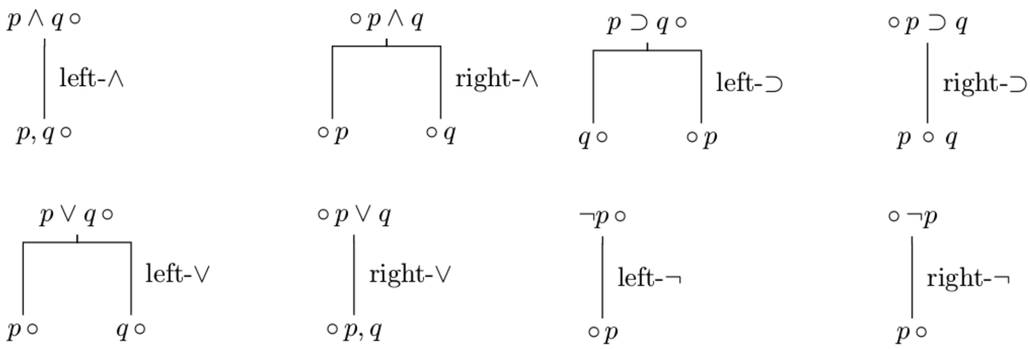
   (b)
   $$\{*\ \mathtt{x{=}10} \land \mathtt{y{=}0}\ *\}$$
   $$\textbf{while}\ \mathtt{x{>}0}\ \textbf{do}\ \{\mathtt{x} := \mathtt{x{-}2}\ ;\ \mathtt{y} := \mathtt{y{+}20}\ \}$$
   $$\{*\ \mathtt{x{+}y} = \mathtt{100}\ *\}$$

   (c) Below, the variable a is an array of int, and sorted is a boolean. The program checks if the array segment $\mathtt{a}[\mathtt{0}..\mathtt{N})$ is sorted.
   $$\{*\ \mathtt{k{=}1}\ \land \mathtt{N{>}0}\ \land\ \mathtt{sorted}{=}\textbf{true}\ *\}$$
   $$\textbf{while}\ \mathtt{k{<}N} \land \mathtt{sorted}\ \ \textbf{do}\ \{\ \mathtt{sorted} := (\mathtt{a}[\mathtt{k}-\mathtt{1}]{\le}\mathtt{a}[\mathtt{k}])\ ;\ \mathtt{k}{:=}\mathtt{k{+}1}\ \}$$
   $$\{*\ \mathtt{sorted}\ =\ (\forall \mathtt{i} : \mathtt{1}{\le}\mathtt{i}{<}\mathtt{N} : \mathtt{a}[\mathtt{i}-\mathtt{1}] \le \mathtt{a}[\mathtt{i}])\ *\}$$

2. [challenging 0.2pt] Suppose this specification is valid under partial correctness:
   $$\{*\ \mathtt{I}\ *\}\ \ \textbf{while}\ g\ \textbf{do}\ S\ \ \{*\ \mathtt{Q}\ *\}$$
   Furthermore I is also an invariant of the loop above. Prove (it does not have to be a formal proof) that $(g \land I) \lor (\neg g \land Q)$ is also invariant.

3 Propositional Theorem Proving [1 pt]

[1 pt] Construct refutation trees for the sequents below. Specify a counterexample if the sequent turns out to be invalid. Use the standard analytic refutation rules from the lecture:



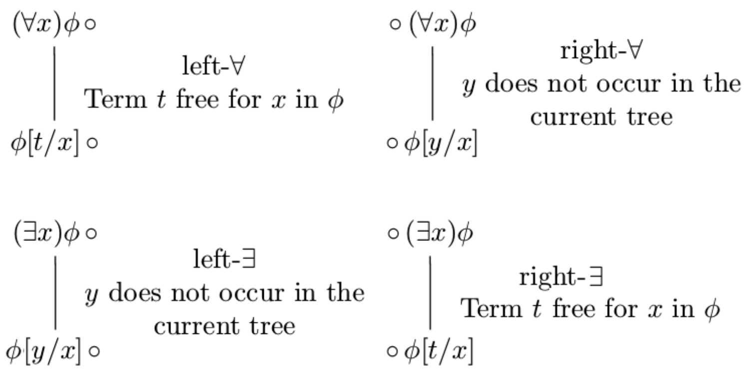
1. $k \supset (e \wedge o)\ ;\ o \supset (b \wedge t) \quad \vdash \quad \neg k \wedge t$
2. $b \wedge \neg d\ ;\ a \supset (b \supset (c \supset d)) \quad \vdash \quad c \supset \neg a$

4 First-Order Theorem Proving [2.5 pt]

1. [0.5 pt] Construct a refutation tree for the following sequent:

$$(\exists x)((p \supset qx)) \quad \vdash \quad (p \supset (\exists y)(qy))$$

Specify a counterexample if the sequent turns out to be invalid. Use the standard analytic refutation rules (see above) and the additional rules for first-order logic:



2. [1 pt] The sequent $\vdash (\forall x)(\exists y)(\forall z)(\exists w)(rx, y \vee \neg rw, z)$ is valid. After a number of steps, we can close the refutation tree as follows (because $rv_2, \varsigma_2(v_1)$ is unifiable with $r\varsigma_1, v_3$):

$$\begin{array}{c} \vdots \\ | \\ rv_2, \varsigma_2(v_1); rv_4, \varsigma_3(v_3) \circ \phi; \psi; r\varsigma_1, v_1; r\varsigma_1, v_3 \\ \times \text{ with } \sigma = \{\varsigma_1/v_1, \varsigma_1/v_2, \varsigma_2(\varsigma_1)/v_3\} \end{array}$$

   Can it also be closed with another substitution? (explain)

3. [1 pt] Does the unification algorithm always halt? Explain your answer. (Hint: Consider the total number of variables in t_1 and t_2.)

5 Clause Sets and Resolution [2.5 pt]

1. [0.5 pt] Describe a polynomial-time algorithm for determining wether a CNF is a tautology.

2. [0.5 pt] Use (binary) resolution to prove that the following proposition is unsatisfiable. Convert to clause sets first.

$$(r \vee \neg u) \wedge (\neg r \vee s) \wedge (u \vee s) \wedge (\neg s \vee v) \wedge (\neg v \vee \neg s)$$

3. [1.5 pt] Solve the following problem with binary resolution and demodulation.

   - Alice's and Betty's spouses are friends.
   - If two persons are friends, then the first person is also a friend of the spouse of the second person.
   - Friendship is symmetrical.
   - Married persons are the spouse of their spouse.

   Are Alice and Betty friends?