COMPUTER SCIENCE TRIPOS Part II – 2025 – Paper 8

6 Hoare Logic and Model Checking (cp526)

Consider a programming language with commands C consisting of the `skip` no-op command, sequential composition C_1`;`C_2, loops `while` B `do` C for Boolean expressions B, conditionals `if` B `then` C_1 `else` C_2, assignment X `:=` E for program variables X and arithmetic expressions E, heap allocation X `:= alloc(`E_1`,...,`E_n`)`, heap assignment `[`E_1`]` `:=` E_2, heap dereference X `:= [`E`]`, and heap location disposal `dispose(`E`)`. Assume `null` $= 0$, and predicates for lists and partial lists:

$$\begin{aligned}
&\text{list}(t, []) = (t = \text{null}) \wedge emp \\
&\text{list}(t, h :: \alpha) = \exists y.(t \mapsto h) * ((t+1) \mapsto y) * \text{list}(y, \alpha) \\
&\text{plist}(t_1, [], t_2) = (t_1 = t_2) \wedge emp \\
&\text{plist}(t_1, h :: \alpha, t_2) = \exists y.\ (t_1 \mapsto h) * ((t_1+1) \mapsto y) * \text{plist}(y, \alpha, t_2)
\end{aligned}$$

In the following, all triples are linear separation logic triples.

(a) Give a proof outline for the following triple or explain why a proof must fail.
$\{P = p \wedge p \mapsto v\}$ `Q:=P; dispose(Q); [P]:=1;` $\{p \mapsto 1\}$ [2 marks]

(b) Define a separation logic predicate array(t, α) for pointers t and mathematical lists α that, unlike the list predicate, represents the values of α in consecutive memory cells starting from t. [3 marks]

(c) Define and explain a partial correctness rule for a new command `nondet`. For a given command C, `nondet`(C) nondeterministically either executes C or does nothing. Maintain soundness of the proof system, and ensure the rule accurately reflects the behaviour of the new command. [3 marks]

(d) Give a loop invariant that would prove the following triple, for a command that interleaves two lists of the same length (no proof required). The zip function interleaves mathematical lists, for example zip $[1,3,5]$ $[2,4,6] = [1,2,3,4,5,6]$.
$\{(\text{list}(X, \alpha) * \text{list}(Y, \beta)) \wedge \text{length}\ \alpha = \text{length}\ \beta\}$
```
P:=X; Q:=Y;
while P ≠ null do
  (N1:=[P+1]; N2:=[Q+1]; [P+1]:=Q; [Q+1]:=N1; P:=N1; Q:=N2)
```
$\{\text{list}(X, \text{zip}\ \alpha\ \beta)\}$ [5 marks]

(e) Give a loop invariant that would prove the following triple, for a command that splits a non-empty list of length $2n$ (for some n) into two lists of alternating elements (no proof required).
$\{\text{list}(X, \alpha) \wedge \alpha \neq [] \wedge \exists n.\ \text{length}\ \alpha = 2n\}$
```
Y:=[X+1]; LX:=X; LY:=Y; N:=[LY+1];
while N ≠ null do
  ([LX+1]:=N; LX:=N; N:=[N+1]; [LY+1]:=N; LY:=N; N:=[N+1]);
[LX+1]:=null; [LY+1]:=null;
```
$\{\exists \beta, \gamma.\ (\text{list}(X, \beta) * \text{list}(Y, \gamma)) \wedge \alpha = \text{zip}\ \beta\ \gamma\}$ [7 marks]