UNIVERSITY OF DUBLIN
TRINITY COLLEGE

Faculty of Engineering, Mathematics, and Science

School of Computer Science and Statistics

B.A. (Mod.) Computer Science
Junior Sophister Examination

3BA31: Formal Methods

Wednesday, 20th May
Arts Building 3051

14:00–16:00

Dr. Andrew Butterfield

Instructions to Candidates:

Attempt three questions.
(all questions carry equal marks)

Materials permitted for this examination:

Q1.

(a) What role does an “Environment” play in giving meanings to both expressions and predicates? Illustrate your answer by giving two environments, one falsifying, the other “truthifying”, the following predicate,

$$\exists x : \mathbb{N} \bullet x < y \land y - z > x$$

given that $$y, z \in \mathbb{N}$$. (9 marks)

(b) Prove the following law:

$$\left( \forall x \bullet P \Rightarrow N \right) \equiv \left( \exists x \bullet P \Rightarrow N \right), \quad x \notin \mathbb{N}$$

You may use any laws in the reference at the end of this paper, or in the Handbook of Mathematics. (12 marks)

(c) Prove the following laws of programming:

$$x := e ; P = P[e/x]$$
$$x := e ; (P \triangleright c \triangleright Q) = (x := e ; P) \triangleright c[e/x] \triangleright (x := e ; Q)$$

using the first to help in proving the second. (12 marks)

Q2. The factorial function, $$n!$$ or $$\text{fac}(n)$$ satisfies the following laws:

$$\begin{align*}
\text{fac}(0) &= 1 \\
\text{fac}(n) &= n \ast \text{fac}(n - 1), \quad n > 0
\end{align*}$$

We have the following specification ($$FSpec$$) and proposed program ($$FProg$$) and invariant $$inv$$:

$$\begin{align*}
FSpec &\triangleq f, x : [f' = \text{fac}(n)] \\
FProg &\triangleq f, x := 1, n; (x > 1) \ast (f, x := f \ast x, x - 1) \\
inv &\triangleq f \ast \text{fac}(x) = \text{fac}(n)
\end{align*}$$

both with alphabet $$\{n, n', f, f', x, x'\}$$. Assume all variables are integer.

(a) Prove: $$FSpec \sqsubseteq f, x : [inv' \land x' \leq 1]$$ (9 marks)

(b) Prove: $$f, x : [inv'] \sqsubseteq f, x := 1, n$$ (9 marks)

(c) Prove: $$f, x : [x > 1 \land inv \Rightarrow inv'] \sqsubseteq f, x := f \ast x, x - 1$$ (9 marks)

(d) What refinement laws in the Reference (p4) can then be used to justify that $$FProg$$ refines $$FSpec$$ (at least with regard to partial correctness)? (6 marks)
Q3.

(a) What are the key principles that a system of refinement should satisfy? (7 marks)

(b) Which of the following predicate-transformer functions are monotonic?

\[ F(P) \equiv Q \lor \neg(R \land \neg P) \]
\[ G(P) \equiv Q \lor \neg(P \land \neg R) \]
\[ H(P) \equiv P \lor \neg(P \land \neg R) \]
\[ K(P) \equiv P \land (P \equiv Q) \]

Justify your answer in each case (8 marks)

(c) In UTP, why do healthiness predicate-transformers have to be idempotent? (6 marks)

(d) Given the following definitions of \( H_1 \) and \( H_2 \),

\[ H_1(P) \equiv \text{ok} \Rightarrow P \]
\[ H_2(P) \equiv P ; (\text{ok} \Rightarrow \text{ok}') \land \nu' = \nu, \quad \alpha P = \{\text{ok}, \text{ok}', \nu, \nu'\} \]

prove that they commute: \( H_1 \circ H_2 = H_2 \circ H_1 \) (12 marks)

Q4.

(a) Prove the following sequence property:

\[ s \neq \langle \rangle \land t \neq \langle \rangle \Rightarrow (\text{front}(s) \sim t)(\#s) = \text{head}(t) \]

(hint: induction on non-nil \( s \) should work) (16 marks)

(b) Sequence subtraction \((s - t)\) is partial, only being defined when its second argument is a prefix of its first \((t \leq s)\). Given the following definitions of \( R_1 \) and \( R_2 \),

\[ R_1(P) \equiv \text{tr} \leq \text{tr}' \]
\[ R_2(P) \equiv \exists s \cdot P[s, s \sim (\text{tr}' - \text{tr})/\text{tr}, \text{tr}'] \]

prove that they commute (i.e., \( R_1 \circ R_2 = R_2 \circ R_1 \)), and highlight how the partiality of sequence subtraction plays a critical role in the proof. (17 marks)
Reference

\( \forall \forall \text{- distr} \)
\( N \lor (\forall x \bullet P) \equiv (\forall x \bullet N \lor P), \ x \not\in N \)
\( \land \exists \text{- distr} \)
\( N \land (\exists x \bullet P) \equiv (\exists x \bullet N \land P), \ x \not\in N \)
\( \text{gen-deMorgan} \)
\( (\exists x \bullet P) \equiv \neg(\forall x \bullet \neg P) \)
\( \exists 1 \text{- pt} \)
\( (\exists x \bullet x = e \land P) \equiv P[e/x] \)
\( \exists \exists \text{- def} \)
\( (\exists b : \exists \bullet P) \equiv P[False/b] \lor P[True/b] \)
\( \text{cnsq-} \lor \text{- distr} \)
\( P \Rightarrow Q \lor R \equiv (P \Rightarrow Q) \lor (P \Rightarrow R) \)
\( := \text{- def} \)
\( x :=_A e \triangleq x' = e \land \nu' = \nu, \ A = \{x, x', \nu, \nu'\} \)
\( \vdash \text{- def} \)
\( P; Q \triangleq (\exists \nu_m \bullet P[\nu_m/\nu'] \land Q[\nu_m/\nu]) \)
\( \text{subst-comp} \)
\( P[e/x][f/x] = P[e/f/x]/x \)
\( \text{subst-inv} \)
\( P[x/y][y/x] = P, \ x \not\in P \)
\( \text{subseq-} \land \text{- def} \)
\( P[e_1, \ldots, e_n/x_1, \ldots, x_n][f/y] = P[e_1, \ldots, e_n, f/x_1, \ldots, x_n/y], \ y \not\in e_1, \ldots, e_n \)
\( \vdash \Rightarrow \text{- subst} \)
\( (P \vdash c \Rightarrow Q)[e/x] = P[e/x] \vdash c[e/x] \Rightarrow Q[e/x] \)
\( \text{frame-def} \)
\( \bar{x} : [P] \triangleq P \land \nu' = \nu \)
\( \text{prog-strengthen} \)
\( P \sqsubseteq P \land Q \)
\( \text{sim-} := \text{- def} \)
\( \bar{x} :=_A e \bar{e} \triangleq x'_1 = e_1 \land \ldots \land x'_n = e_n \land \nu' = \nu \)
\( \text{=} \text{- subst} \)
\( x = e \land P \equiv x = e \land P[e/x] \)
\( \text{=} \Rightarrow \text{- subst} \)
\( (x = e \land P \Rightarrow Q) \equiv (x = e \land P \Rightarrow Q[e/x]) \)
\( \text{=} \text{- subst} \)
\( S \sqsubseteq P \land x = e \equiv (S[e/x] \sqsubseteq P \land x = e) \)
\( \text{=} \text{- abort} \)
\( \text{true} \subseteq P \)
\( \text{=} \text{- trans} \)
\( (S \sqsubseteq Q) \land (Q \sqsubseteq P) \Rightarrow (S \sqsubseteq P) \)
\( \text{rel-init-loop-} \sqsubseteq \)
\( \bar{v} : [\text{pre} \Rightarrow \text{inv}' \land \neg c'] \sqsubseteq \bar{v} : [\text{pre} \Rightarrow \text{inv}'] ; c \star \bar{v} : [c \land \text{inv} \Rightarrow \text{inv}'] \)
\( \text{front-def-one} \)
\( \text{front}(x) \triangleq \langle \rangle \)
\( \text{front-def-cons} \)
\( \text{front}(xs) \triangleq xs(\text{front}(s)) \)
\( \# (\langle \rangle) \triangleq 0 \)
\( \# (xs) \triangleq 1 + \# s \)
\( \text{index-def-one} \)
\( (xs)(1) \triangleq x \)
\( \text{index-def-cons} \)
\( (xs)(n) \triangleq s(n-1) \)
\( \text{index-1-is-head} \)
\( s(1) \triangleq \text{head}(s), \ s \neq \langle \rangle \)
\( \text{seq-} \delta \text{- subst} \)
\( \delta(s - t) \triangleq t \leq s \)
\( \text{seq-} \leq \text{- seq} \)
\( s \leq s \sim t, \ \delta s \land \delta t \)