Name: ....
Student Number ....

Software Engineering 2F04

DAY CLASS — Dr. Mark Lawford
DURATION OF EXAMINATION: 3 Hours
McMaster University Final Examination — December 1999

THIS EXAMINATION PAPER INCLUDES 6 PAGES AND 5 QUESTIONS. YOU ARE RESPONSIBLE FOR ENSURING THAT YOUR COPY OF THE PAPER IS COMPLETE. BRING ANY DISCREPANCY TO THE ATTENTION OF YOUR INVIGILATOR.

Special Instructions: The use of calculators, notes, and text books is not permitted during this exam. Answer all questions in the provided answer booklets. Fill in your name and student number and sign each booklet you use. This paper must be returned with your answers.

Useful Factoids:

Rules Governing Equality

Ia : $\vdash (\forall x)(x = x)$

Ib : $\vdash (\forall x)[x = t \rightarrow (\phi \leftrightarrow \phi[x, t|x])]$

Ic : $\vdash (\forall x)(\forall y)(x = y \rightarrow y = x)$

Id : $\vdash (\forall x)(\forall y)(\forall z)(x = y \wedge y = z \rightarrow x = z)$

1. Propositional & Predicate Logic (35 marks)

a) (4 marks) Show that implication is not associative. i.e. show that

$$((P \rightarrow Q) \rightarrow R) \not\Leftrightarrow (P \rightarrow (Q \rightarrow R))$$

b) (6 marks) Consider the following propositional logic formula

$$(P \wedge Q \rightarrow R) \leftrightarrow (P \rightarrow (Q \rightarrow R))$$

i) Prove that the formula is a tautology.
ii) What two valid rules of inference does it provide?

c) (7 marks) Determine if the following argument is valid or invalid. Justify your answer.
Premises: $(\forall x)(\forall y)(Rxy \rightarrow x = y), (\exists x)Px, (\exists x)\neg Px$
Conclusion: $(\exists x)(\exists y)\neg Rxy$

Continued on page 2

d) (5 marks) Determine if the following argument is valid or invalid. Justify your answer. (Hint: Think.)

Premises: $(\forall x)(\forall y)(Rxy \rightarrow x = y), (\exists x)Px, (\exists x)\neg Px$

Conclusion: $(\forall x)(\forall y)Rxy$

e) (5 marks) Determine if the following set of premises is consistent or inconsistent. Justify your answer. (Hint: Think again.)

Premises: $(\forall x)(\forall y)(Rxy \rightarrow x = y), (\exists x)Px, (\exists x)\neg Px, (\forall x)(\forall y)Rxy$

f) (3 marks) Given a set of premises Γ and a conclusion ϕ, suppose that you find an interpretation structure that provides a counter example that shows that $\Gamma \not\models \phi$. What can you conclude about the consistency of Γ? Why?

g) Consider the interpretation structure $\mathbf{S} := \langle \mathbf{U}, \mathbf{W}, \mathbf{a}, \mathbf{b} \rangle$ where:

$$\mathbf{U} := \{0, 1\}, \mathbf{W} := \{1\}, \mathbf{a} := 0, \mathbf{b} := 1$$

i) (2 marks) Write down a formula ϕ using equality ($=$) that would be satisfied by any interpretation structure with a non-trivial universe (i.e. For any $\mathbf{S}' := \langle \mathbf{U}', \ldots \rangle$ it is the case that $\mathbf{S}' \models \phi$ iff $|\mathbf{U}'| > 1$).

ii) (3 marks) Write down a formula ψ that *does not* use equality such that for $\mathbf{S}$ as defined above $\mathbf{S} \models \psi$ and ψ has the property that $\psi \rightarrow \phi$.

2. Partial Functions and PVS Typechecking (10 marks)

a) (4 marks) Consider the following PVS declaration:

```
U:TYPE
a:U
```

It generates the following Type Correctness Condition (TCC):

```
% Existence TCC generated (at line 13, column 2) for  a: U
  % untried
a_TCC1: OBLIGATION EXISTS (x: U): TRUE;
```

i) Why is the TCC generated?

ii) Assuming PVS doesn't have any inconsistencies, why is it unprovable?

iii) Why would it be provable if PVS had a soundness bug (i.e. it was possible to prove $\perp$)?

b) (4 marks) Consider the function:

$$f(x, y) = \sqrt{x + y}$$

Write down the best PVS definition for f.

c) (2 marks) Assume that $x \neq y$ is an abbreviation for $\neg(x = y)$. In the Parnas/IMPS traditional analysis approach to logic, what is the truth value of the formula $(\exists x)((\sqrt{x})^2 \neq |x|)$? Justify your answer.

Continued on page 3

3. Mathematical Induction (15 marks)

a) (5 marks) The following rule, call Rule MI, is an *axiom* of Peano Arithmetic (i.e. it can be used on any line of a proof for a formula of Peano Arithmetic):

$$\vdash \phi[0|n] \wedge (\forall m)(\phi[m|n] \rightarrow \phi[m+1|n]) \rightarrow (\forall n)\phi$$

Here ϕ is a formula of Peano Arithmetic, $n \in FV(\phi)$ and m is a "new" variable that does not occur in ϕ.

Informal mathematical induction involves showing (i) $\phi[0|n]$ is true and (ii) assuming that $\phi[m|n]$ is true, you show that $\phi[m+1|n]$ is true. Use formal proof rules to show why this is sufficient to prove $(\forall n)\phi$.

b) (10 marks) Use mathematical induction to informally prove that every odd power of 2 + 1 is divisible by 3. That is, show that for every n, there exists some i such that $2^{(2n+1)} + 1 = 3i$.

4. More Predicate Logic with Equality (15 marks)

Any function $f : U \rightarrow U$ induces an equivalence relation K_f, the *equivalence kernel of* f, given by

$$K_f xy \text{ if and only if } f(x) = f(y).$$

a) (6 marks) In this question you will formally prove that given a function $f : U \rightarrow U$, the equivalence kernel of f is an equivalence relation. To do this, formally prove the following:

i) Reflexivity: $(\forall x)K_f xx$,

ii) Symmetry: $(\forall x)(\forall y)(K_f xy \rightarrow K_f yx)$,

iii) Transitivity: $(\forall x)(\forall y)(\forall z)(K_f xy \wedge K_f yz \rightarrow K_f xz)$.

b) (2 marks) We can define a partial order on equivalence relations as follows: Let E_1 and E_2 be equivalence relations on U. Then we say that E_1 is a refinement of E_2, written $E_1 \preceq E_2$ iff $(\forall x)(\forall y)(E_1 xy \rightarrow E_2 xy)$.

Given functions $f : U \rightarrow U$ and $g : U \rightarrow U$, write down a predicate logic formula involving the function symbols f and g that is true when $K_g \preceq K_f$).

c) (5 marks) Consider the following result from discrete mathematics:

Theorem: Given two functions with the same domain, $f : V_1 \rightarrow V_3$ and $g : V_1 \rightarrow V_2$, then there exists $h : V_2 \rightarrow V_3$ such that the diagram in Figure 1 commutes iff $K_g \preceq K_f$.

$$\begin{tikzcd}
V_1 \arrow[rr, "f"] \arrow[dr, "g"'] & & V_3 \\
& V_2 \arrow[ur, dashed, "h"'] &
\end{tikzcd}$$

Figure 1: Commutative diagram for $(\exists h : V_2 \rightarrow V_3)(\forall v_1 \in V_1)[h(g(v_1)) = f(v_1)]$ iff $K_g \preceq K_f$

The interpretation of this result is that for h to exist, g must retain as much or more information about its domain than f.

Continued on page 4

You will now show that $K_g \preceq K_f$ is a necessary condition for the existence of the function h in the special case when $V_1 = V_2 = V_3$ by formally showing the following result:

$$\vdash (\forall x)[f(x) = h(g(x))] \rightarrow (\forall x)(\forall y)[g(x) = g(y) \rightarrow f(x) = f(y)]$$

d) (2 marks) Use the previous result to show that:

$$\models (\exists x)(\exists y)(g(x) = g(y) \wedge f(x) \neq f(y)) \rightarrow (\exists x)(f(x) \neq h(g(x))$$

5. Software Verification with PVS (25 marks)

In this problem we study the verification of a simplified pressure sensor trip that monitors a pressure sensor and is "tripped" when the sensor value exceeds a normal operating setpoint. As was the case with the power conditioning example of Assignment 5, the specification of the pressure sensor trip makes use of deadbands to eliminate chatter. The proposed specification and the actual implementation for the sensor trip are give in Figure 2 by `f_PressTrip` and `PTRIP`, respectively. In the function definitions, `f_PressTripS1` and `PREV` play corresponding roles as the arguments for the previous value of the state variable computed by the function.

Figure 2 also contains the supporting type and abstraction function definitions for verifying that the implementation meets the specification. The abstraction function `real2AItype` models the A/D (analog to digital) conversion of the pressure sensor value by taking the integer part of its input using the function

```
floor(x:real): {i:int | i <= x & x < i + 1}
```

from the PVS prelude file. It is used to map the real valued specification input `Pressure` to the discrete implementation input `PRES` which has type `AIType`. `AIType` consists of the subrange of natural numbers between 0 and 5000.

At the bottom of the specification in Figure 2, the theorem `Sentrip1` is an example of a *block comparison theorem* that could be used to prove that the implementation `PTRIP` will produce the same output as the specification `f_PressTrip` for all possible inputs. Attempting to prove the theorem `Sentrip` results in the following unprovable sequent:

```
[-1]  Pressure!1 < 2450
[-2]  floor(Pressure!1) <= 2400
  |-------
[1]   Pressure!1 <= 2400
[2]   NotTripped?(f_PressTripS1!1)

Rule?
```

a) (5 marks) Write down the characteristic equation for the sequent.

b) (5 marks) Find all the values of `Pressure!1` and `f_PressTripS1!1` that provide counter examples for the equation.

c) (2 marks) Pick specific values for `Pressure!1` and `f_PressTripS1!1` that provide a counter example and confirm that it provides a counter example to theorem `Sentrip` by evaluating `f_PressTrip(Pressure!1,f_PressTripS1!1)` and `bool2Trip(PTRIP(real2AItype(Pressure!1),Trip2bool(f_PressTripS1!1)` and comparing the results.

Continued on page 5

sentrip : THEORY
 BEGIN

Trip : TYPE = {Tripped, NotTripped}

AItype : TYPE = $\{i : \text{nat} \mid 0 \leq i \wedge i \leq 5000\}$

f_PressTrip(Pressure : real, f_PressTripS1 : Trip) : Trip = TABLE

$\text{Pressure} \leq 2400$	$2400 < \text{Pressure} \wedge \text{Pressure} < 2450$	$\text{Pressure} \geq 2450$
NotTripped	f_PressTripS1	Tripped

ENDTABLE

PTRIP(PRES : AItype, PREV : bool) : bool = TABLE

$\text{PRES} \leq 2400$	$2400 < \text{PRES} \wedge \text{PRES} < 2450$	$\text{PRES} \geq 2450$
FALSE	PREV	TRUE

ENDTABLE

Trip2bool(TripVal : Trip) : bool = TABLE

TripVal = Tripped	TripVal = NotTripped
TRUE	FALSE

ENDTABLE

bool2Trip(BoolVal : bool) : Trip = TABLE

BoolVal = TRUE	BoolVal = FALSE
Tripped	NotTripped

ENDTABLE

real2AItype(x : real) : AItype = TABLE

$x \leq 0$	$0 < x \wedge x < 5000$	$x \geq 5000$
0	floor(x)	5000

ENDTABLE

Sentrip1 : THEOREM
 ($\forall$ (Pressure : real, f_PressTripS1 : Trip) :
  f_PressTrip(Pressure, f_PressTripS1) =
   bool2Trip(PTRIP(real2AItype(Pressure), Trip2bool(f_PressTripS1))))

END sentrip

Figure 2: Formatted PVS specification for pressure sensor trip example

Continued on page 6

d) (2 marks) State a theorem that could be used to prove that the implementation does not meet the specification. This would provide confirmation that the unprovable sequent for theorem `Sentrip1` results from inconsistencies between the specification and implementation and not from a poor choice of PVS prover commands by the verifier. This is an example of refutation theorem proving where a software engineer tries to prove that the implementation is *NOT* equivalent to the specification.

e) (3 marks) In fact it is currently impossible to change the definition of `PTRIP` so that it will satisfy the specification `f_PressTrip`. Why? (Hint: See question 4 part (d).)

f) (5 marks) How could the original specification `f_PressTrip` be modified so that it is possible to find a new implementation of `PTRIP` that would make `Sentrip1` true? (HINT: You only need to make minor modifications to the inequalities of `f_PressTrip`.)

g) (3 marks) Suppose the A/D conversion hardware has a tolerance of ± 5 associated with it. The interpretation is that an input `Pressure` value of say 17.3 could produce the same results as any value in the range $12.3 \leq \text{Pressure} \leq 22.3$. This being the case, strict functional equivalence of an implementation with a specification might be considered too restrictive. Suppose that it is now acceptable for the original implementation of `PTRIP` to merely produce an output that is within tolerance. This means that for every value of `Pressure` there is some other value, say `P`, such that

$$\texttt{Pressure} - 5 \leq \text{P} \leq \texttt{Pressure} + 5$$

and when `f_PressTrip` is evaluated at the real value of `Pressure`, applying `PTRIP` to the converted version of `P` produces the same result.

Restate theorem `Sentrip1` to take this tolerance into account. (HINT: One way to do it involves the use of existential quantification over a dependent type.)

"S.A.L.T. - 'Students Against Lawford's Tenure.' Join today!" - 2F04 student

The End