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

Software Engineering 2F03

DAY CLASS — Dr. Mark Lawford
DURATION OF EXAMINATION: 3 Hours
McMaster University Final Examination — December 1998

THIS EXAMINATION PAPER INCLUDES 5 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.

1. Propositional & Predicate Logic (25 Marks Total)

a) Proofs and Universal Specification

i) (2 marks) Verify that the following propositional formula is a tautology:

$$(P \to Q_1) \wedge (\neg P \to Q_2) \leftrightarrow (P \wedge Q_1) \vee (\neg P \wedge Q_2)$$

ii) (4 marks) Prove that the following formula is a theorem of predicate logic:

$$(\forall x)(\forall y)[(Px \to Qf(x)y) \wedge (\neg Px \to Qg(x)y) \leftrightarrow (Px \wedge Qf(x)y) \vee (\neg Px \wedge Qg(x)y)]$$

iii) (2 marks) The postcondition of the program statement

```
if (x>=3) y=x+1;
else y=1-x;
```

can be written as $(x \geq 3 \to y = x + 1) \wedge (x < 3 \to y = 1 - x)$. Find a logically equivalent formula ϕ that is a disjunction of two subformulas (i.e. ϕ is of the form $\phi_1 \vee \phi_2$).

b) (6 marks) Validity of Arguments: Determine whether the following is a valid argument. Justify your answer.
Premises: $(\forall x)(Ax \to Cx), (\forall x)(Bx \to Dx), (\exists x)Ax, (\exists x)\neg Dx$
Conclusion: $(\exists x)(Cx \wedge \neg Bx)$

c) In the following you will deduce some properties of a world where there are winners and losers (those who did not win).

i) (7 marks) Formally prove:

$$(\exists x)Wx, (\exists x)\neg Wx \vdash (\forall x)(\exists y)x \neq y$$

ii) (2 marks) What do you conclude about the cardinality of the universe for any interpretation structure satisfying the two premises?

iii) (2 marks) Find the simplest interpretation structure (i.e. having the smallest universe) that is a model for the formula:

$$(\exists x)Wx \wedge (\exists x)\neg Wx \to (\forall x)(\exists y)x \neq y$$

Continued on page 2

2. Tabular Specification I: Weakening conditions on tabular definitions (15 Marks Total)

Consider the tabular definition:

$$f(x,y) = \begin{array}{|c|c|c|} \hline C_1xy & C_2xy & C_3xy \\ \hline f_1(x,y) & f_2(x,y) & f_3(x,y) \\ \hline \end{array}$$

a) (7 marks) Assume functions f_1, f_2 and f_3 are defined when C_1, C_2 and C_3 are respectively true. Using our standard notation for predicate calculus, write down the two formulas, one for Disjointness and one for Completeness, that PVS would require a user to prove for the above table. In this case the Disjointness and Completeness proof obligations (TCCs) are sufficient to guarantee that the table defines a (total) function.

b) (6 marks) The PVS example in Figure 1 provides some insight as to why the Disjointness conditions generated by PVS are overly restrictive. The theorem "same" can be easily proved using the (GRIND) command but when a file containing the definitions is type checked, the disjointness condition fails for `f2`. For the table used to define f in part (a) above, create a weaker "disjointness" condition that together with the completeness condition provides necessary and sufficient conditions for the table defining f to be a total function.

.

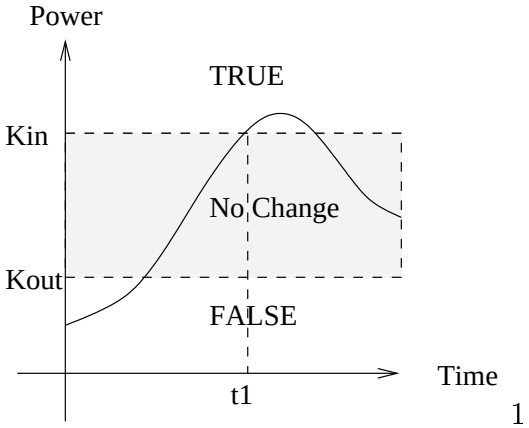
```
x: VAR real

                        %--------------%
f1(x): real = TABLE     |[ x<0 | x>=0 ]|
                        %--------------%
                        |  x   | 2*x   ||
              ENDTABLE  %--------------%

                        %---------------%
f2(x): real = TABLE     |[ x<=0 | x>=0 ]|
                        %---------------%
                        |   x   | 2*x   ||
              ENDTABLE  %---------------%

same: THEOREM FORALL (x:real): f1(x)=f2(x)
```

Figure 1: Disjointness condition counter example

c) (2 marks) Although the weakened "disjointness" condition together with the usual completeness condition provides necessary and sufficient conditions for a table to define a function. Why is it preferable for software engineers to use the more strict disjointness condition when using tables to specify the functional requirements of software?

Continued on page 3

3. Tabular Specifications II: Code Reuse Caveats (20 Marks Total)



The figure to the left illustrates a power conditioning function that is used in an industrial control system. When the Power level drops below $Kout$, a sensor becomes unreliable so it is "conditioned out" by setting $PwrCond$ to FALSE. When the power exceeds Kin, the sensor is "conditioned in" and is used to evaluate the system. While $Power$ is between $Kout$ and Kin, the value of $PowerCond$ is left unchanged by setting it to its previous value, $Prev$.

E.g. For the graph of $Power$ above, $PwrCond$ would start out FALSE, then become TRUE at time $t1$ and remain TRUE.

Since different sensors might have different conditioning in and conditioning out values Kin and $Kout$, a general power conditioning function is proposed. The PVS description of the proposed general power conditioning function is shown in Figure 2. Typechecking the definition generates an

```
PwrCond(Prev:bool, Power, Kin, Kout:posreal):bool = TABLE
  %-------------------------------------------------%
  |[Power<=Kout | Power>Kout & Power<Kin | Power>=Kin]|
  %-------------------------------------------------%
  |   FALSE      |        Prev            |   TRUE   ||
  %-------------------------------------------------%
ENDTABLE
```

Figure 2: PVS Specification of general $PwrCond$ function

unprovable disjointness TCC. The PwrCond_TCC1 disjointness TCC and the unprovable sequent that results from trying to prove PwrCond_TCC1 are shown in Figure 3.

a) (6 marks) Write down the characteristic formula for the unprovable sequent in Figure 3. Find a counter example that makes the characteristic formula false. (NOTE: Your counter examples values for Kin, $Kout$ and $Power$ must be of the correct type - $posreal = \{x : real | x > 0\}$.)

b) (4 marks) Verify that the counter example satisfies the conditions of two or more columns of the table for PwrCond, thereby proving the table as defined does not properly specify a function.

c) (4 marks) What implicit assumption did the designers make regarding input arguments Kin and Kout that led them to omit the counter example case from the table? Why is such an undocumented assumption dangerous in a setting where code may be reused by other developers?

d) (6 marks) Use dependent typing to create a new version of the PwrCond table that makes the assumed relation between Kin and Kout explicit and thereby rules out any counter examples like those from (i). The columns of the resulting table should be disjoint.

Continued on page 4

```
% Disjointness TCC generated (at line 19, column 54) for
    % TABLE
    %   |[ Power <= Kout | Power > Kout & Power < Kin | Power >= Kin ]|
    %   |    TRUE        |      Prev                  | FALSE        ||
    %   ENDTABLE
  % unfinished
PwrCond_TCC1: OBLIGATION
      (FORALL (Kin: posreal, Kout: posreal, Power: posreal):
         NOT (Power <= Kout AND Power > Kout & Power < Kin)
             AND NOT (Power <= Kout AND Power >= Kin)
               AND NOT ((Power > Kout & Power < Kin) AND Power >= Kin));

PwrCond_TCC1 :

[-1]    Kin!1 > 0
[-2]    Kout!1 > 0
[-3]    Power!1 > 0
[-4]    Power!1 <= Kout!1
[-5]    (Kin!1 <= Power!1)
  |-------
[1]     FALSE

Rule?
```

Figure 3: TCC and resulting unprovable sequent for $PwrCond$

4. Proof by Induction (15 Marks Total)

a) (3 marks) Write down the mathematical induction postulate (axiom schema for rule MI) for the theory of Peano Arithmetic.

b) (5 marks) Explain why Rule MI is a valid rule of inference by showing how the base step and inductive step can be used to formally prove $\phi[k|n]$ for a given natural number $k \in \mathbb{N}$.

c) (7 marks) Use induction to prove that $4^n - 1$ is divisible by 3 for all $n \geq 0$.

Continued on page 5

5. Partial Functions & Types in Logic + a bit of PVS Predicate Logic (25 Marks Total)

a) (4 marks) Explain why $\neg(x < 1/y)$ is not logically equivalent to $x \geq 1/y$ in the traditional analysis style logic used by IMPS and Parnas.

b) (6 marks) Write down the most concise formulas in both the IMPS/Parnas (analysis style) logic and bounded quantification (PVS style) logic that could be used to specify that A, an N element array of integers, has the property:

The array does not contain a strictly increasing sequence of elements.

c) (5 marks) Consider the function $h : \mathbb{R} \to \mathbb{R}$ given by:

$$h(x) = 1 + \frac{1}{|x^3 + 5x^2 + 6x|}$$

Define appropriate subtypes and write down a new definition of the function h that will allow its use in a traditional (PVS style) logic. Restrict the return type of the function as much as possible to help with definedness proofs (TCCs)

Justify your choice of domain and range for f.

d) (5 marks) Consider the application of the PVS (SKOLEM!) command show below:

$$\begin{array}{|l} \phi_1 \\ \phi_2 \\ \vdots \\ \hline (\forall x)\psi \end{array} \quad \overset{\text{(SKOLEM!)}}{\Longrightarrow} \quad \begin{array}{|l} \phi_1 \\ \phi_2 \\ \vdots \\ \hline \psi[x_1|x] \end{array}$$

Here x_1 is a new variable not occurring elsewhere in the sequent. PVS is saying, in effect, that to prove $\Gamma \vdash (\forall x)\psi$, it is *sufficient* to prove $\Gamma \vdash \psi[x_1|x]$ for an appropriately chosen x_1. Why?

e) (5 marks) Using what you know about Logical Equivalence (Rule LE) and how PVS handles negations, show how the above explanation also covers the use of the PVS (SKOLEM!) command in the case where:

$$\begin{array}{|l} \phi_1 \\ \phi_2 \\ \vdots \\ (\exists x)\phi \\ \hline \psi_1 \\ \psi_2 \\ \vdots \end{array} \quad \overset{\text{(SKOLEM!)}}{\Longrightarrow} \quad \begin{array}{|l} \phi_1 \\ \phi_2 \\ \vdots \\ \phi[x_1|x] \\ \hline \psi_1 \\ \psi_2 \\ \vdots \end{array}$$

Use intermediate sequents to illustrate the process and provide justification for the transformation of one sequent to the next.

"Logic is good. That is all." - 2F03 student

The End