SFWR ENG 2F04 Assignment 5: Typechecking, Software Verification, Induction, and much, much more!

Due: 1230 Thursday December 2, 1999

All of your PVS work for this assignment should be done in a single file called `A5.pvs`. Download this file from the web at the URL:

`http://www.cas.mcmaster.ca/~lawford/2F04/Notes/A5.pvs`

It contains some of the PVS you will need to do this assignment. You will have to add to this file as detailed in the questions below. When you are done the PVS part, you will submit it electronically as a PVS dump file called `A5.dmp`. Written work will be handed in separately at the *start* of class on the due date.

NOTE: For up to date information on how all you voracious little PVS piranhas can submit your work to the sacrificial cow, please check out the URL:

`http://www.cas.mcmaster.ca/~lawford/2F04/e-submissions.html`

1. **Tabular Specification I: Weakening conditions on tabular definitions** (25 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.

   The PVS example in theory `A5Q1` of the file `A5.pvs` provides some insight as to why the Disjointness conditions generated by PVS are overly restrictive.

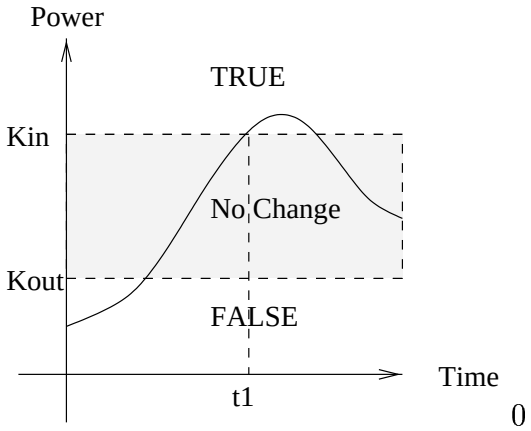
   b) (2 marks) Prove the theorem `same`.

   c) (6 marks) Typecheck the theory and have PVS try to prove the resulting TCCs with the PVS/Parsing and Typechecking/typecheck-prove command. Take a look at the TCCs for the file with the PVS/Viewing TCCs/show-tccs command. As you can see, the disjointness condition `h_TCC1` fails for the table defining `h`. Rerun the proof PVS tried by placing you cursor on this TCC and invoking the prover. Write down the characteristic equation for the resulting unprovable sequent and use it to obtain a counter example to the disjointness condition. This is a case where the table still defines a total function even though PVS' disjointness condition is violated. Why?

   d) (7 marks) 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.

   e) (3 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?

2. **Tabular Specifications II: Code Reuse Caveats** (25 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 `PwrCond` 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 for the rest of the time shown on the graph.

Since different sensors might have different conditioning in and conditioning out values, `Kin` and `Kout` respectively, a general power conditioning function is proposed. The PVS description of the proposed general power conditioning function appears in theory `A5Q2` of the `A5.pvs` file.

a) (1 mark) Typecheck the definition with the typecheck-prove command. View the TCCs and rerun the unprovable disjointness TCC `PwrCond_TCC1`. Write down the unprovable sequent that results from trying to prove PwrCond_TCC1.

b) (6 marks) Write down the characteristic formula for the unprovable sequent. 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 - $posreal = \{x : real | x > 0\}$.)

c) (6 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.

d) (6 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?

e) (6 marks) Use dependent typing to create a new version of the table called `PwrCond2` immediately below the definition of `PwrCond`. This new table must make the assumed relation between `Kin` and `Kout` explicit and thereby rule out any counter examples like those from (b). The columns of the resulting table should be disjoint. Verify that this is the case for your example by using the typecheck-prove command and then viewing the TCCs.

3. **Proof by Induction** (30 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 informally 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 by hand that $4^n - 1$ is divisible by 3 for all $n \geq 0$.

d) (5 marks) Prove the same thing by Induction in PVS by proving theorem `Q3d` of theory `A5Q3`. As the first proof step use the `(INDUCT "n")` command to let PVS know you are attempting a proof by induction on the variable `n`. Next, use the command `(EXPAND "^")` to rewrite the exponent shorthand in terms of the recursive function `expt(r,n)`. Expand the definition of `expt` in the bottom part of the sequent but not the top, then use the `(BOTH-SIDES ...)` command

to obtain something that you can use with the `(REPLACE ...)` command. Use the PVS/Getting Help/help-pvs-prover and help-pvs-prover-command menu options for more information on these commands.

e) (5 marks) Do Rubin p. 302 A 3,22 by hand.

f) (5 marks) Do Rubin p. 302 A 3 in PVS by adding to the theory `A5Q3` a recursive function definition called `sumA3` and an appropriate theorem called `Q3f`. Again, use the command `(INDUCT "n")` as the first command to help prove the theorem. Be sure to typecheck-prove your theory and view the TCCs. Note the termination TCC that is generated by the declaration of `sumA3` guaranteeing the there is a bounded monotonically decreasing quantity associated with the recursive definition!

4. **Partial Functions & Predicate Subtypes in Logic** (20 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 partial function $h : \mathbb{R} \to \mathbb{R}$ given by:

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

Create a new theory called `A5Q4` in the file A5.pvs. In the theory define appropriate domain and range predicate subtypes called `domh` and `rangeh` and use them to write down a definition of the function h that will allow its use in a traditional (PVS style) logic. Restrict the domain the absolute minimum required to guarantee h will always be defined. Restrict the return type of the function as much as possible to help with any definedness proofs (TCCs) for any theory that would make use of h.

Justify your choice of domain and range for h in your written part of the work.

d) (5 marks) Prove all TCCs for your theory (This one is a little tricky and may require stuff from the prelude file. Enjoy! 8-)