**ECE/CS 584: Embedded System Verification** Fall 2012

*Homework 2— Due on October $22^{nd}$, 2012* *Your name*

For this homework, submit a zip file containing appropriate .pvs and .prf files along with a document briefly describing your model (using the hybrid automaton language we use in class) and your proof approach.

**Problem 1 (40 points).** Consider a idealized billiard table of length $a$ and width $b$. This table has no pockets, its surface has no friction, and it's boundary bounces the balls perfectly. Write a hybrid automaton model of the position of **two** balls on this table in PVS. The balls have some initial velocities. They collide whenever either $|x_1 - x_2| \leq \epsilon$ or $|y_1 - y_2| \leq \epsilon$, where $\epsilon$ is some constant. Whenever a collision occurs, the balls exchange their velocity vectors. Prove conservation of momentum as an invariant property in PVS.

Some suggestions. Start by modifying the `MinProblem` theory. Use the `simplemachine.pvs` theory and the induction theorem to prove the invariance. Since the mass of the balls remain constant, you may simplify the model and the proofs to keep track of the velocities (instead of momentum). You may use basic axioms of real arithmetic (without proving them).

**Problem 2. (50 points)** In this problem, you will model the $n$-process distributed token ring system (from last problem set) in PVS and prove its key invariant.

**Recall the system description.** Consider $n$ processes $0, \ldots, n-1$ connected in a directed ring. We say process $i+1 \mod n$ is the successor of process $i$. Each process $i$, has a value $v_i$ which can be an element of the set $\{0, \ldots, k\}$ for some $k > n$. Each process behaves as follows: Process $i$, $i \neq 0$, is said to have a token iff $v_i \neq v_{i-1}$. Process 0 has a token iff $v_0 = v_{n-1}$. Each process has a real-valued period parameter $\Delta_i > 0$. Exactly every $\Delta_i$ time, process $i$ performs the following action if it has the token: if $i = 0$ then $v_i := (v_i + 1) \mod n$, otherwise $v_i := v_{i-1}$.

**Part (a)** An almost complete PVS specification of the system is provided from the homeworks page. This specification, albeit not correct, should parse and typecheck correctly in your installation of PVS. Complete the specification with appropriate expressions in the lines marked Fill in.

**Part (b)** Check for type correctness. Are there any unproved TCCS ? Prove them. You may have to use basic lemmas on modular arithmetic from `prelude.pvs`.

**Part (c)** The predicate *two_val* implies that there is at most one token in the system. Prove invariance of *two_val* using the PVS prover. This is broken down into two lemmas in the supplied theory. Partial proof are provided to get you started.