To: CSci 8801, All students
CC: Teaching Assistant
From: Dr. Mats Heimdahl
Date: 11/28/2006
Re: Homework Assignment 5

Introduction
We will again revisit the ASW (the one altimeter version) for exposure to a final tool—the model checker NuSMV. This time we will have to do some simplifying assumptions to make it easy to use to model checker. In particular, the symbolic model checkers are generally not easily used when we have large integers involved (and cannot be used at all if we have real numbers in there).

Allowable Simplifications
The altitude switch does not easily make it into SMV in its current form. Thus, here are some simplifications to consider.

Altitude
Consider replacing the altitude with an enumerated input (unknown, below, in-between, above, for example).

Alternatively, you could restrict the altitude to a small integer range (SMV can deal with that). For example, you could make the altitude range from 0-5, the threshold 2, and the hysteresis 2.

Timers
The timers are modeled as variables of the TIME type in your previous models. This will not work in SMV. Consider replacing the timers with Boolean variables (indicating whether a timeout happened or not).

Simultaneous inputs
If you leave the input variables to SMV unconstrained any number of them can change value arbitrarily—the notion of “one interface at the time” enforced in RSML-e does not hold). This is not really a problem, but you should be aware and you may need to consider this when you analyze your model.

The Problem
Build the model of the ASW using NuSMV and analyze it for the following requirements (you can do more if you can think of any interesting ones). Should a requirement not hold it is quite likely that the requirement is wrong and your model is right (see the discussion in the “Proving the Shalls” paper).
If that is the case, please refine the requirement to what it should really have been (based on your best judgment and forum discussions).

1. If we descend below the threshold the ASW we will send an on command to the DOI.
2. If any timeout has occurred, the alarm will be turned on.
3. If the alarm is turned on, it is possible to turn it off.

**Technical Support**
Questions regarding SMV shall be posted on the Forum.

**Your Task**
Develop an SMV model of the ASW and verify (and modify) the requirements using NuSMV.

**Deliverables**
I want you to turn in one solution either individually or in a team of two. I expect you to make sure the specification is (1) liberally commented, (2) syntactically correct and type checked (done when parsed into NuSMV), and (3) verified. Hand in the following:

1. A soft copy of the specification (email is preferred).

**Due Date**
Friday, December 15.
Interfaces to the Environment

Communication with the DOI

TYPE_DEF DOIStatusType {On, Off}

MESSAGE DOICommandMessage {
    command IS DOIStatusType
}

MESSAGE DOIStatusMessage {
    status IS DOIStatusType
}

IN_INTERFACE DOIStatusMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(DOIStatusMessage)

    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            DOIStatus := status
            END ASSIGNMENT
        END HANDLER

END IN_INTERFACE

OUT_INTERFACE DOICommandInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    OUTPUT_ACTION : SEND(DOICommandMessage)

    HANDLER :
        CONDITION : TBD
        ASSIGNMENT
            command := TBD
            END ASSIGNMENT
        ACTION : SEND
        END HANDLER

END OUT_INTERFACE
**Communication with the Alarm**

MESSAGE FaultMessage {
    fault IS BOOLEAN
}

OUT_INTERFACE FaultDetectionInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 200 MS
    OUTPUT_ACTION : SEND(FaultMessage)
    HANDLER :
        CONDITION : TBD
        ASSIGNMENT
            fault := TBD
        END ASSIGNMENT
        ACTION : SEND
    END HANDLER
END OUT_INTERFACE

**Communication with the altitude device**

TYPE_DEF AltitudeQualityType { Good, Bad }

MESSAGE AltitudeMessage {
    Alt IS INTEGER,
    aq IS AltitudeQualityType
}

IN_INTERFACE AltitudeMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(AltitudeMessage)
    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            TBD := Alt,
            TBD := aq
        END ASSIGNMENT
    END HANDLER
END IN_INTERFACE
**Communication with the inhibit button**

TYPE_DEF InhibitType { Inhibit, NoInhibit }

MESSAGE InhibitMessage {
  i IS InhibitType
}

IN_INTERFACE InhibitMessageInterface :
  MIN_SEP : 50 MS
  MAX_SEP : 100 MS
  INPUT_ACTION : RECEIVE(InhibitMessage)

  RECEIVE_HANDLER :
    CONDITION : TRUE
    ASSIGNMENT
      TBD := i
    END ASSIGNMENT
  END HANDLER

END IN_INTERFACE

**Communication with the reset button**

MESSAGE EmptyMessage {x IS INTEGER}
/* The current environment simulation requires a message field */
/* even if one is not needed. X will never be used. */

IN_INTERFACE ResetMessageInterface :
  MIN_SEP : 50 MS
  MAX_SEP : 100 MS
  INPUT_ACTION : RECEIVE(EmptyMessage)

  RECEIVE_HANDLER :
    CONDITION : TRUE
    ASSIGNMENT
      ivReset := TRUE
    END ASSIGNMENT
  END HANDLER

  HANDLER :
    CONDITION : TRUE
    ASSIGNMENT
      ivReset := FALSE
    END ASSIGNMENT
  END HANDLER

END IN_INTERFACE
/* The variable ivReset is used to capture if a reset message */
/* was received. The interface will set ivReset to TRUE when */
/* a message arrives and to false immediately in the next */
/* instance of time. */