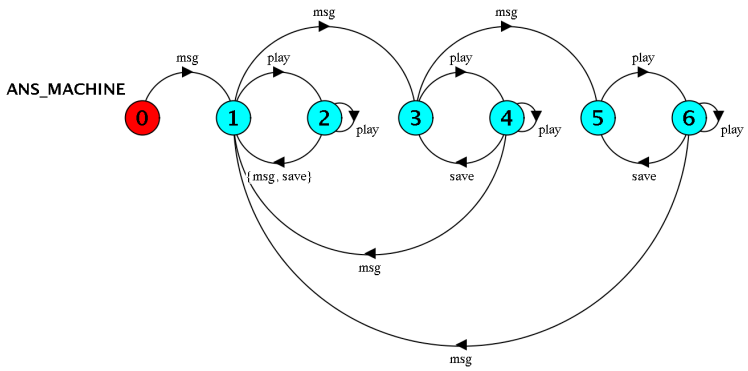
CMU 17-651 Models of Software Systems Fall 2018

Homework #6: State Machines II and FSP

Dario A Lencina-Talarico Due: 8 October 2018

1. Consider the answering machine described in HW 5. Write an FSP specification of **AnsMachine**. (For your answer, include the text of the specification and turn in a diagram drawn by LTSA as an attachment.)

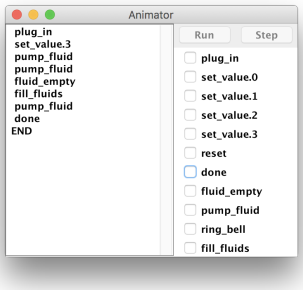



```
ANS_MACHINE = NONE,

NONE = (msg -> ONE),

ONE = (msg -> TWO
      |play -> TEMP1
      ),

TWO = (msg -> THREE
      |play -> TEMP2
      ),

THREE = (play -> TEMP3),

TEMP1 = (play -> TEMP1
        |save -> ONE
        |msg  -> ONE
        ),

TEMP2 = (play -> TEMP2
        |save -> TWO
        |msg  -> ONE
        ),
```

```
TEMP3 = (play -> TEMP3
        |save -> THREE
        |msg  -> ONE
        ).
```

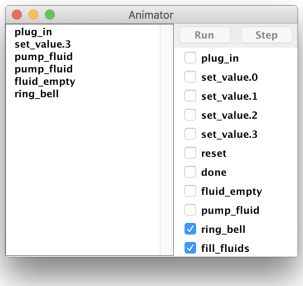
2. Consider the FSP specification of a simplified version of the Infusion Pump attached at the end of this assignment. Answer the following questions:

   a. Give an example of an action-based trace that causes the infusion pump to terminate without an alarm being raised.



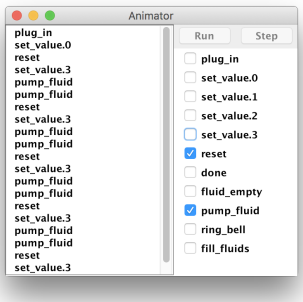
$trace\_alarm\_not\_raised == \langle plug\_in, set\_value\_3, pump\_fluid, pump\_fluid, fluid\_empty, fill\_fluid, pump\_fluid, done \rangle$

   b. Give an example of an action-based trace that causes the infusion pump to terminate with an alarm being raised.



$trace\_alarm\_raised == \langle plug\_in, set\_value\_3, pump\_fluid, pump\_fluid, fluid\_empty, ring\_bell \rangle$

   c. Is there a limit to how much medication can be administered to a patient? Explain why or why not.



As it can be seen in the picture, if we reset the machine between filling procedures, it is possible to administer an arbitrary ammount of medicine. This will require human intervention to reset and refill process.

On it's own the machine has a capacity of $FillAmt = 2$.

d. What happens if the nurse forgets to put any medicine in the bag (i.e., uses a refill amount of 0)?
$nurse\_forgot == \langle plug\_in, set\_value\_0 \rangle$
If the nurse makes the machine execute the previous trace, the only options available are to *end* or *reset* the machine. The nurse would be able to select another medicine if she/he selects *reset*.

e. Is it possible for an alarm to sound even if the patient has received the correct and full amount of medicine?
I was unable to find a trace that will end in *ring_bell* given the listed conditions.

3. Modify the FSP specification above to add two of the following capabilities, making sure to explain in your comments which capabilities you are adding.

a. self-check at start-up

b. confirmation of settings

c. a start and end of treatment time

d. other error condition detection

e. ability to set the amount of medicine to be dispensed at each dispensing step

f. power outage and automatic switch to backup power supply

Submit an electronic copy of the modified FSP specification.

```
//-------------------------------------------------
//  Infusion Pump modified by Dario
//-------------------------------------------------
//
// Set of actions that can be selected interactively to
// the animation of this model with the LTSA tool.
//
menu AnimationControlMenu = {
    plug_in, set_value[0..3], reset, fill_fluids
}
//-------------------------------------------------
//======================
// Constants and Ranges
//======================
const Max = 3 range Amt = 0 .. Max
const FillAmt = 2    // Amount in bag initially and after refilling
// Added by Dario:
// Ammout of medicine allowed to be dispensed at each step
//
range Step = 1 .. FillAmt

// Added by Dario:
const SimulationStartTime = 0  // Time when the program was executed.
const EndOfTime = 100 // This is the end of time for the simulation.
range Timestamp = SimulationStartTime .. EndOfTime

//======================
// Process Definitions
//======================
//
// Pump starts in power off state
//
PUMP = POWER_OFF,
//
// User must plug pump in before anything else can happen
//
POWER_OFF = (
    plug_in -> SELF_TEST
),

// Added by Dario:
// After plugging the machine, it performs a self-test or fire-up process.
// If successful, proceeds to the SETUP starge, else, it returns the error and ENDS.
// SELF_TEST assumes that failure occurs randomly for educational purposes.
//

SELF_TEST = (
    self_test -> SETUP
    | self_test -> SHOW_ERROR
),
```

```
SHOW_ERROR = (show_error -> ERROR),
//
// Before pump operation starts, user must enter amount of medicine to deliver
// to patient
//
SETUP = (
    set_value[deliver:Amt][step:Step] -> SETUP_CONFIRMATION[deliver][step]
),

//
// Added by Dario:
// The SETUP_CONFIRMATION step allows the nurse to confirm its selection and proceeed
// to the medicament PUMP process or to go back to SETUP and select another amount of
// medicine.
//

SETUP_CONFIRMATION[deliver:Amt][step:Step] = (
   user_confirmed_settings -> PUMP[deliver][FillAmt][step]
   |user_rejected_settings -> SETUP
),

//
// Modified by Dario to supply the ammount of medicine defined in step.
// Main operation of pump:
//  User may reset pump at any time
//  When the pump has delivered the amount of medicine requested it goes
//      to the DONE state
//  When fluid runs out, the pump goes into an alarm state
//  Otherwise, the pump delivers one unit of medicine
//

PUMP[deliver:Amt][remaining:Amt][step:Step] = (
    reset -> SETUP
    |
    when (deliver == 0)
        done -> DONE
    |
    when (remaining < step)
        fluid_empty -> EMPTY_ALARM[deliver][step]
    |
    when (deliver >= step && remaining >= step)
        pump_fluid -> PUMP[deliver-step][remaining-step][step]
    |
    when (deliver < step && remaining >= step)
        pump_fluid -> PUMP[0][remaining - step][step]
),
//
// Error state associated with empty pump:
//  Repeatedly rings bell until user refills the pump
//
EMPTY_ALARM[deliver:Amt][step:Step] = (
```

```
    ring_bell -> EMPTY_ALARM[deliver][step]
    | fill_fluids -> PUMP[deliver][FillAmt][step]
),

DONE = END.

//
// Added by Dario:
// POWER_MANAGEMENT is used to control whether the system operates using the energy grid or the backup b

POWER_MANAGEMENT = GRID,

GRID = ( tick -> GRID
        | tick -> BACKUP_BATTERY
),

//
// Once the system switches to the backup battery, it remains using it until the machine is stopped.
//
BACKUP_BATTERY = (tick -> BACKUP_BATTERY).

//
// Clock used to store the simulation time, on each tick it gets increased by one until it reaches the e
CLOCK = TICK[SimulationStartTime],

TICK[t:Timestamp] = (
    when(t < EndOfTime)
        tick -> TICK[t + 1]
    |
    when(t >= EndOfTime)
        tick -> ERROR
).
//
// Run Clock, Power Management and the Pump in parallel.
//
|| CLOCK_PUMP = (CLOCK || PUMP || POWER_MANAGEMENT).
```