FORMAL SPECIFICATION

The control software for an automated insulin pump

CSc 365 Critical Systems Engineering 2002

Schema Definition for the Insulin Pump

The overall definition of an INSULIN_PUMP states that the pump is either starting up after being switched on (STARTUP), running normally (RUN), out of action as a result of some fault or because the user has switched to manual override (MANUAL), executing a self-test procedure (TEST) or resetting itself after a new phial of insulin has been fitted (RESET).

INSULIN_PUMP = STARTUP RESET TEST RUN MANUAL

The INSULIN_PUMP_STATE schema defines a set of state variables for the insulin pump and invariants for the device. Invariants are logical statements that are always true.

The system input hardware is modelled as follows:

- A blood sugar sensor which measures the current blood sugar reading in micrograms/millilitre. This is updated every 10 minutes. It is modelled in the Z specification as *Reading*?. The value of *Reading*? is normally between 1 and 35.
- A hardware test unit which runs a self-test on the hardware every 30 seconds. Problems detected by this unit are modelled in the Z specification as *HardwareTest?*.
- A three position switch which can be set to off, manual or auto mode. This is modelled in the Z specification as *switch?.*
- A button to specify the amount of insulin to be delivered when in manual mode. The number of button presses within a 5 second period specifies the number of units of insulin to be injected. The system must be in manual mode for this to be operational and you may assume that after the first press is detected, the number of presses is counted automatically by the hardware. This is modelled in the Z specification as *ManualDeliveryButton?*.
- Presence sensors for the insulin reservoir and the needle assembly. These switch status when the insulin reservoir/needle is attached or removed. They are modelled in the Z specification as *InsulinReservoir?* and *Needle?*.
- A hardware clock which is factory set and which maintains the current time. This is represented by the state variable *clock?*.

The Z specification does not mention any timing information. It is assumed that this is preset in the hardware. In the simulation, you must simulate the changing values of these inputs but do not have to simulate the timing of these changes.

The output hardware is as follows:

- Three displays that provide information to the user about the state of the system. These are modelled as *display1!, display2!* and *clock!*. The operation of these displays is described in requirement 3.8 above and in the following schema.
- A pump unit that delivers a specified dose of insulin. This is modelled in the Z specification as *dose!*.
- An alarm which gives an audible signal to the user if a problem has arisen. This is modelled as *alarm!*. The hardware examines this value every 10 seconds and either sets off or cancels the audible alarm.

Several other state variables are also defined in the INSULIN_PUMP_STATE schema:

- *status* represents the current status of the controller. In normal operation, the status is running; if a state exists which is potentially hazardous then the status is warning; if a hazardous state exists, then the status is error. In the error state, normal operation of the device is suspended ie blood sugar levels are not computed.
- *r0*, *r1*, and *r2* maintain information about the last three readings from the sugar sensor. *r2* holds the current reading, *r1* the previous reading and *r0* the reading before that. These are used to compute the rate of change of blood sugar readings
- *capacity* represents the capacity of the system's insulin reservoir and *insulin_available* represents the amount of insulin in the reservoir that is currently available for delivery.
- *max_daily_dose*, *max_single_dose* and *minimum_dose* are included as a result of the system safety requirements and reduce the probability that an insulin overdose will be delivered.

max_daily_dose is the maximum dose of insulin that can be delivered in a 24 hour period; *max_single_dose* is that maximum dose of insulin that can be delivered in a single injection. *minimum_dose* is the dose that may be delivered to maintain an existing trend in blood sugar levels.

- *safemin*, *safemax* are the boundaries of the region defining the safe levels of blood sugar.
- *CompDose* is the insulin dose required according to the insulin dosage computation. This may be overridden for safety reasons; *cumulative_dose* is the total dose that has already been delivered over the last 24 hours.

INSULIN_PUMP_STATE

```
//Input device definition

switch?: (off, manual, auto)
ManualDeliveryButton?: ℕ
Reading?: ℕ
HardwareTest?: (OK, batterylow, pumpfail, sensorfail, deliveryfail)
InsulinReservoir?: (present, notpresent)
Needle?: (present, notpresent)
clock?: TIME

//Output device definition

alarm! = (on, off)
display1!, ℙ string
display2!: string
clock!: TIME
dose!: ℕ

// State variables used for dose computation

status: (running, warning, error)
r0, r1, r2: ℕ
capacity, insulin_available : ℕ
max_daily_dose, max_single_dose, minimum_dose: ℕ
safemin, safemax: ℕ
CompDose, cumulative_dose: ℕ
```

```
r2 = Reading?
dose! ≤ insulin_available
insulin_available ≤ capacity
// The cumulative dose of insulin delivered is set to zero once every 24 hours
clock? = 000000 ⇒ cumulative_dose = 0
// If the cumulative dose exceeds the limit then operation is suspended
cumulative_dose >= max_daily_dose ⇒ status = error ⇒ display1! = “Daily dose exceeded”

// Pump configuration parameters.
capacity = 100
safemin = 6
safemax = 14
max_daily_dose = 25
max_single_dose = 4
minimum_dose = 1

display2! = nat_to_string (dose!)
clock! = clock?
```

The RUN schema defines the system state for normal operation. The software defined in the RUN schema should execute every 10 minutes.

RUN

ΞINSULIN_PUMP_STATE

switch? = auto
status = running status = warning
insulin_available ≥ max_single_dose
cumulative_dose < max_daily_dose

(SUGAR_LOW SUGAR_OK SUGAR_HIGH)

// If the computed insulin dose is zero, don't deliver any insulin

CompDose = 0 dose! = 0

// The maximum daily dose would be exceeded if the computed dose was delivered

CompDose + cumulative_dose > max_daily_dose alarm! = on status' = warning
 dose! = max_daily_dose – cumulative_dose

// The normal situation. If maximum single dose is not exceeded then deliver computed dose

CompDose + cumulative_dose < max_daily_dose
(CompDose ≤ max_single_dose dose! = CompDose

// The single dose computed is too high. Restrict the dose delivered to the maximum single dose

CompDose > max_single_dose dose! = max_single_dose
)
insulin_available' = insulin_available – dose!
cumulative_dose' = cumulative_dose + dose!

insulin_available ≤ max_single_dose * 4 status' = warning display1! = display1!
 "Insulin low"

r1' = r2
r0' = r1

The MANUAL schema models the system behaviour when it is in manual override mode. Notice that cumulative_dose is still updated but that no safety checks are applied until the system is reset to automatic mode.

MANUAL

Ξ INSULIN_PUMP_STATE
switch? = manual
display1! = "Manual override"
dose! = ManualDeliveryButton?
cumulative_dose' = cumulative_dose + dose!
insulin_available' = insulin_available – dose!

The SUGAR_LOW schema defines the state when the measured sugar level in the user's blood falls below the safe minimum value.

SUGAR_LOW

r2 < safemin
CompDose = 0
alarm! = on
status' = warning
display1! = display1! ⌢ "Sugar low"

The SUGAR_OK schema defines the state when the measured sugar level in the user's blood falls within the desired range. Insulin is only delivered in circumstances where it appears that the level is likely to go outside this range.

SUGAR_OK

r2 ≥ safemin ∧ r2 ≤ safemax
// sugar level stable or falling
r2 ≤ r1 ⇒ CompDose = 0

// sugar level increasing but rate of increase falling
r2 > r1 ∧ (r2-r1) < (r1-r0) ⇒ CompDose = 0

// sugar level increasing and rate of increase increasing compute dose
// a minimum dose must be delivered if rounded to zero
r2 > r1 ∧ (r2-r1) ≥ (r1-r0) ∧ (round ((r2-r1)/4) = 0) ⇒
CompDose = minimum_dose

r2 > r1 ∧ (r2-r1) ≥ (r1-r0) ∧ (round ((r2-r1)/4) > 0) ⇒
CompDose = round ((r2-r1)/4)

The SUGAR_HIGH schema defines the state when the level of sugar in the user's blood is above the desired level. A dose of insulin is computed that should bring down this sugar level to within the desirable range.

SUGAR_HIGH

r2 > safemax
// sugar level increasing. Round down if below 1 unit.
r2 > r1 ∧ (round ((r2-r1)/4) = 0) ⇒ CompDose = minimum_dose

r2 > r1 ∧ (round ((r2-r1)/4) > 0) ⇒ CompDose = round ((r2-r1)/4)

// sugar level stable
r2 = r1 ⇒ CompDose = minimum_dose

// sugar level falling and rate of decrease increasing
r2 < r1 ∧ (r2-r1) ≤ (r1-r0) ⇒ CompDose = 0

//sugar level falling and rate of decrease decreasing
r2 < r1 ∧ (r2-r1) > (r1-r0) ⇒ CompDose = minimum_dose

The STARTUP schema models the behaviour of the system when the user switches on the device. It is assumed that the user's blood sugar at that stage is OK. Note that cumulative_dose is NOT set in the startup sequence but can only be set to zero at midnight.

This means that the total cumulative dose delivered can always be tracked by the system and is not affected by the user switching the machine on and off.

STARTUP

Δ INSULIN_PUMP_STATE
switch? = off ∧ switch?' = auto
dose! = 0
r0' = safemin
r1' = safemax
TEST

The RESET schema models the system when the user changes the insulin reservoir. Notice that this does not require the device to be switched off. The design of the reservoir is such that it is not possible to insert reservoirs that are partially full.

RESET

Δ INSULIN_PUMP_STATE
InsulinReservoir? = notpresent and InsulinReservoir?' = present
insulin_available' = capacity

TEST

The TEST schema models the behaviour of the hardware self-test unit which runs a test on the system hardware every 30 seconds. The specification here does not specify what happens when there is the simultaneous failure of more than 1 hardware unit. You may chose to simulate this as you wish. Multiple display facility of display1! Is set up if simultaneous failure

TEST

Δ INSULIN_PUMP_STATE

(HardwareTest? = OK ∧ Needle? = present ∧ InsulinReservoir? = present ∧
status' = running ∧ alarm! = off ∧ display1!= "")
(
status' = error
alarm! = on
(
Needle? = notpresent ∧ display1! = display1! ∧ "No needle unit"
(InsulinReservoir? = notpresent insulin_available < max_single_dose)
∧ display1! = display1! ∧ "No insulin"
HardwareTest? = batterylow ∧ display1! = display1! ∧ "Battery low"
HardwareTest? = pumpfail ∧ display1! = display1! ∧ "Pump failure"
HardwareTest? = sensorfail ∧ display1! = display1! ∧ "Sensor failure"
HardwareTest? = deliveryfail ∧ display1! = display1! ∧ "Needle failure"
)
)