EECS4312 eHealth Project

Siraj Rauff (cse23188@cse.yorku.ca)
Skyler Layne (cse23170@cse.yorku.ca)

December 6, 2015

You may work on your own or in a team of no more than two students. **Submit only one document under one Prism account.**

Prism account used for submission: cse23188

Keep track of your revisions in the table below.

Revisions

Date	Revision	Description
date please	1.0	Initial requirements document

Requirements Document:
for Patient care eHealth System

Contents

List of Figures

List of Tables

1. System Overview

The System Under Development (SUD) is a computer system to create and manage health prescription records for Ontario.

This requirements document is specifically for prescription management. The purpose of the eHealth Patient care System is to maintain physicians, medications, patients, and patient prescriptions. The system will also control the undesirable interactions between medications, that is when two medications conflict in some way with one another. Only specialist physicians should be allowed to prescribe undesirable interactions while general physicians should be allowed to prescribe medications, as long as they do not create undesirable interactions.

2. Context Diagram

The System Under Description (SUD) is a computer *controller* to keep track of the physicians, patients, patient prescriptions, medications, and medication interactions. The monitored variables and controlled variables for this computer system can be found in Table 1 and Table 2 respectively.

The system must keep track of abstract state which isn't available to the user. For a list of the abstract states within the controller see Figure 3.

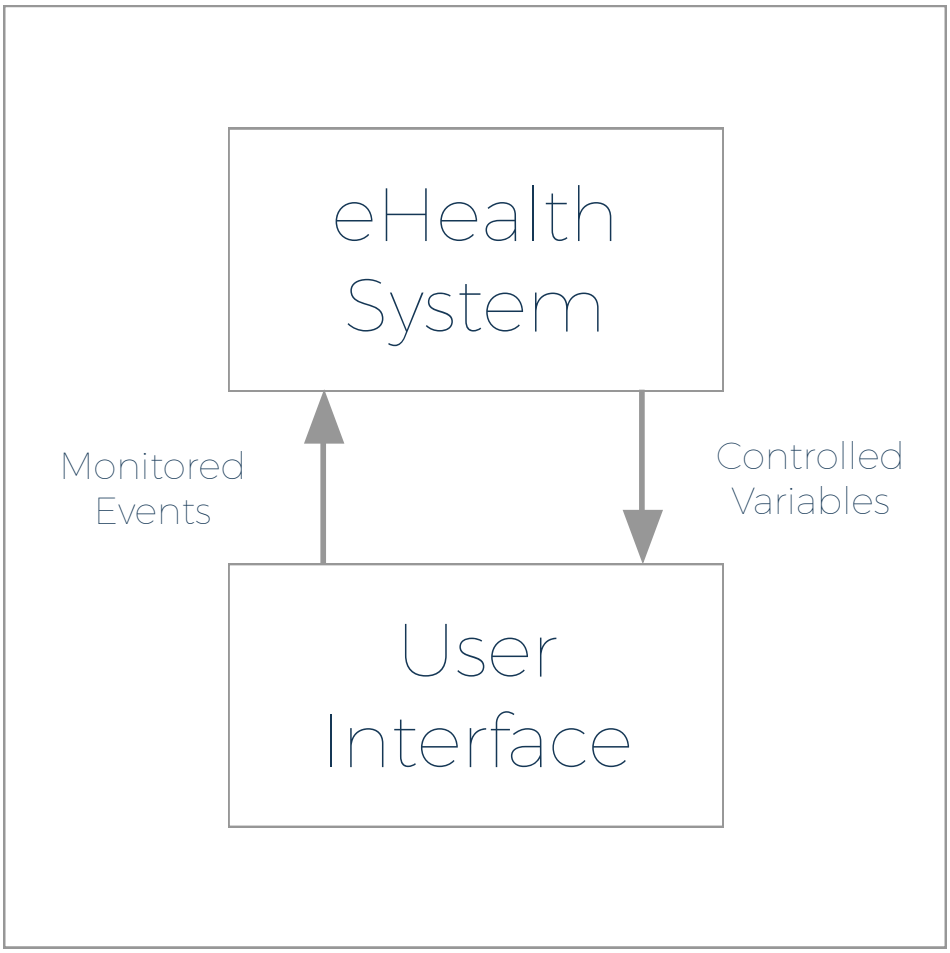

Figure 1: Context Diagram

3. Goals

The high-level goals (G) of the system are:

- G1— The user should be able to add doctors (generalists, and specialists)
- G2— The user should be able to add patients
- G3— The user should be able to add medications
- G4— The user should be able to add perscriptions
- G5— The user should be able to add interactions between medications
- G6— The user should only be able to add a prescription with a dangerous interaction if the doctor is a specialist

4. Monitored Events

The monitored events are those which come through the user interface. The following monitored events will be available to the user.

Name	Interpretation
add_physician(id: ID_MD; name: NAME; kind: PHYSICIAN_TYPE)	Add a Physician to the system
add_patient(id: ID_PT; name: NAME)	Add Patient to the system
add_medication(id: ID_MN; medicine: MEDICATION)	Add a medication to the system,
add_interaction(id1:ID_MN;id2:ID_MN)	Add an interaction between two medications
new_prescription(id: ID_RX; doctor: ID_MD; patient: ID_PT)	Add a new prescription to the system
add_medicine(id: ID_RX; medicine:ID_MN; dose: VALUE)	Add a medicine to a prescription
remove_medicine(id: ID_RX; medicine:ID_MN)	Remove a medication from a prescription
prescriptions_q(medication_id: ID_MN)	Get all the prescriptions with that medication
dpr_q	Get all the dangerous interactions prescribed

Table 1: Monitored Events

5. Controlled Variables

The controlled variables represent what will be shown to the user.

Name	Interpretation	Abstract State
Physicians	A list of all the Physicians currently within the system	See table 3
Patients	A list of all the Patients currently in the system	See table 3
Medications	A list of all the Medications currently within the system	See table 3
Interactions	A list of all the Interactions currently within the system	See table 3
Prescriptions	A list of all the Prescriptions within the system	See table 3
Error	A message displaying the highest priority error, or ok	See table 3

Table 2: Controlled Variables

6. Mode Diagram

REQ1 states The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*, shown in Fig. 2. As shown in the figure, the Isolette will begin in the *off* mode, and will enter *init* mode when the nurse flips *m_sw* into the *on* position. The Isolette will only be able to move from the *init* mode ot the *normal* mode if it is properly configured such that the desired range is valid and not overlapping with the alarm levels, and both the sensors and operator controls are working (see REQ**??**). Once inside the *normal* mode, the controller will move to the *fail* mode if either the controls or sensor fails, as specified in REQ**??**, and will only return to the *normal* mode once they are both working correctly (REQ7). In any of these states, if the nurse switches *m_sw* to *off* the controller will switch to the off mode (REQ8).

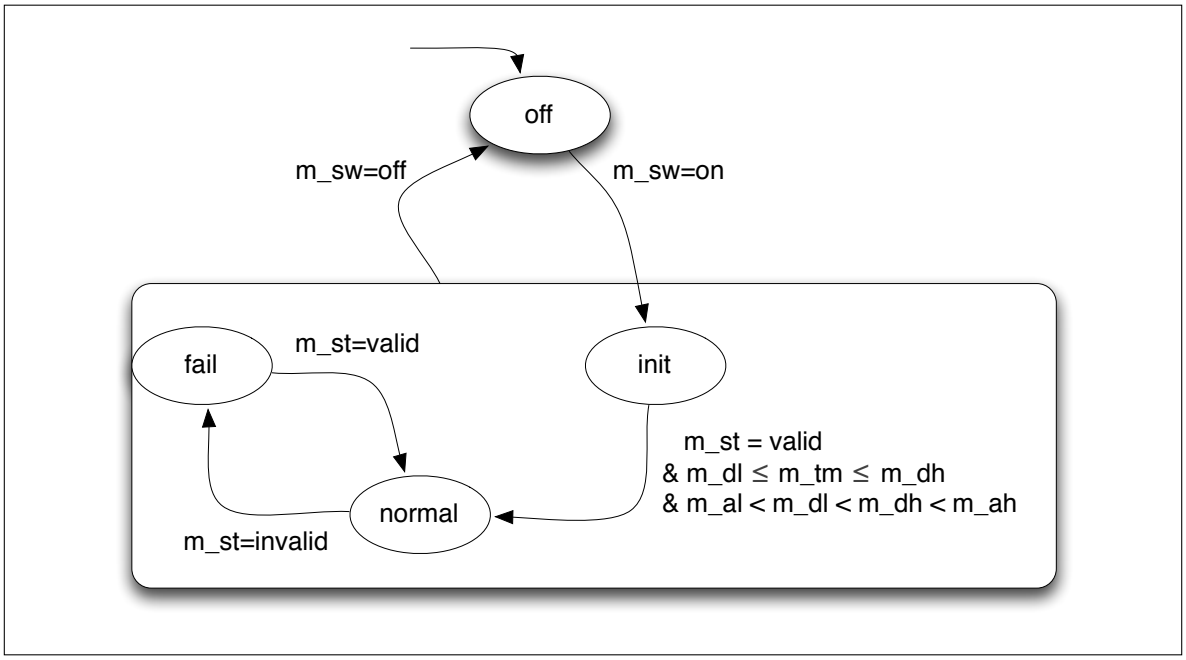

Figure 2: Statechart for the modes variable *c_md*

7. E/R-descriptions

7.1. Requirements Descriptions

REQ1	The system will keep track of Physicians, Patients, Medications, Interactions, and Prescriptions	...

REQ2	The system will output a different response depending on which monitored event occurs.	...

REQ3	In *normal* mode, the controller shall activate an alarm whenever • the current temperature falls outside the *alarm* temperature range (either through temperature fluctuation or a change in the alarm range by an operator), or • a failure is signalled in any of the input devices (temperature sensor and operator settings).	The alarm temperature range is m_al .. m_ah. Monitored variable m_st shows "invalid" when any of the input signals fail.

Rationale: During normal operation, if any conditions occur that could affect the wellbeing of the infant, the nurse should be notified by an alarm. This could include sensor or interface failure, or current temperature exceeding alarm values - even if this is caused by the nurse adjusting the values.

7.2. Environmental Descriptions

ENV4	All input to the system will be assumed as correct, i.e. without syntax errors.	see table for a list of monitored events.

ENV5	The desired and alarm temperatures received from the operator are all in increments of 1°F.	*m_ah*, *m_al*, *m_dh*, and *m_dl* in Table 1: Monitored Variables.

Rationale: The operator interface for the nurse is designed in such a way that the increments of the input temperatures from the control interface must change by whole numbers.

ENV6	Failure of the sensors or operator settings will cause the status to become invalid.	If the sensor or operator settings fail, *m_st* becomes invalid.

Rationale: The environment has ways of detecting failures of the operator interface or the temperature sensor and indicating their status to the controller. The Isolette must constantly be monitoring the status of the sensors and controls to ensure that safe assumptions can be made on their values so as to not endanger the infant.

8. Abstract variables needed for the Function Table

Name	Interpredation	Purpose
mnid	The set of all medication ids	Keep track of all the medication ids in the system
ptid	The set of all patient ids	Keep track of all the patient ids in the system
mdid	The set of all doctor ids	Keep track of all the doctor ids in the system
rxid	The set of all prescription ids	Keep track of all the prescription ids in the system
mdpt	Relationship between Doctor and Patient	Keep track of the doctor and the patient
rx	Relation between doctor patient and medication	Keep track of all the medical prescriptions between doctor and patient in the system
prs	List of all prescriptions including dosage	Keep track of all the prescriptions in the system
di	Set of dangerous interactions between medications	Keep track of all the dangerous interactions in the system
gs	The kind of doctor	Keep track of all the type of doctor
dpi	The dangerous prescription report	Notify dangerous interactions, if they exist

Figure 3: Abstract Variables used in Function Tables

9. Function Tables

9.1. Table of Error codes and Warnings

Name	Meaning
err1	physician id must be a positive integer
err2	physician id already in use
err3	name must start with a letter
err4	patient id must be a positive integer
err5	patient id already in use
err6	name must start with a letter
err7	medication id must be a positive integer
err8	medication id already in use
err9	medication name must start with a letter
err10	medication name already in use
err11	require 0 <low-dose <= hi-dose
err12	medication ids must be positive integers
err13	medication ids must be different
err14	medications with these ids must be registered
err15	interaction already exists
err16	first remove conflicting medicine prescribed by generalist
err17	prescription id must be a positive integer
err18	prescription id already in use
err19	physician with this id not registered
err20	patient with this id not registered
err21	prescription already exists for this physician and patient
err22	prescription with this id does not exist
err23	medication id must be registered
err24	medication is already prescribed
err25	specialist is required to add a dangerous interaction
err26	dose is outside allowed range
err27	medication is not in the prescription
warn1	There are dangerous prescriptions
warn2	There are no dangerous prescriptions

Figure 4: Table of errors and warnings

9.2. Function Table for eHealth

Monitored Inputs		
i = 0		See Table 6
i >0	new_prescription(id, doctor, patient)	See Table 7
	add_medicine(id, medicine, dose)	See Table 8
	remove_medicine(id, medicine)	See Table 9
	add_interaction(id1, id2)	See Table 10

Figure 5: Function Table for eHealth

9.3. Function Table for init

error: $false$

Abstract State	¬ error	error
mind(i)	$\varnothing$	NC
ptid(i)	$\varnothing$	NC
mdid(i)	$\varnothing$	NC
rxid(i)	$\varnothing$	NC
mn(i)	$\varnothing$	NC
mns(i)	ε	NC
mdpt(i)	$\varnothing$	NC
rx(i)	ε	NC
prs(i)	ε	NC
di(i)	$\varnothing$	NC
gs(i)	$\varnothing \mapsto \varepsilon$	NC
dpr(i)	$\varnothing \mapsto \varepsilon$	NC
r(i)	ok	error

Figure 6: Function Table for init

9.4. Function Table for new_prescription(id, md, pt)

error: $\neg(id > 0 \wedge \neg rxid_1(id) \wedge md > 0 \wedge mdid_1(md) \wedge pt > 0 \wedge ptid_1(pt) \wedge mdpt_1(md, pt))$

Abstract State	¬ error	error
mind(i)	NC	NC
ptid(i)	NC	NC
mdid(i)	NC	NC
rxid(i)	$rxid_1 \cup \{id\}$	NC
mn(i)	NC	NC
mns(i)	NC	NC
mdpt(i)	NC	NC
rx(i)	$rx_1 \upharpoonright (id \mapsto (md, pt))$	NC
prs(i)	$prs_1 \upharpoonright (id \mapsto empty\_prs(mnid_1))$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	$dpr_1 \upharpoonright (id \mapsto \varnothing)$	NC
r(i)	ok	error

Figure 7: Function Table for new_prescription(id, doctor, patient)

9.5. Function Table for add_medicine(id, m, d)

error: $\neg(id > 0 \wedge \neg rxid_1(id) \wedge m > 0 \wedge mnid_1(m) \wedge pt > 0 has(m, prs_1(id)) \wedge sumthin \wedge isValidDose(m, d))$

Abstract State	¬ error	error
mind(i)	NC	NC
ptid(i)	NC	NC
mdid(i)	NC	NC
rxid(i)	NC	NC
mn(i)	NC	NC
mns(i)	NC	NC
mdpt(i)	NC	NC
rx(i)	NC	NC
prs(i)	$prs_1(id) \upharpoonright (m \mapsto d))$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	NC	NC
r(i)	ok	error

Figure 8: Function Table for add_medicine(id, medicine, dose)

9.6. Function Table for remove_medicine(id, med)

error: $\neg(id > 0 \wedge \neg rxid_1(id) \wedge med > 0 \wedge mnid_1(med) \wedge prs_1(id)(med)`1 > 0)$

Abstract State	¬ error	error
mind(i)	NC	NC
ptid(i)	NC	NC
mdid(i)	NC	NC
rxid(i)	NC	NC
mn(i)	NC	NC
mns(i)	NC	NC
mdpt(i)	NC	NC
rx(i)	NC	NC
prs(i)	$prs_1 \upharpoonright (id \mapsto \varepsilon)$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	NC	NC
r(i)	ok	error

Figure 9: Function Table for remove_medicine(id, medicine)

9.7. Function Table for add_interaction(id1, id2)

error: $\neg(id1 > 0 \wedge id2 > 0 \wedge \neg id1 = id2 \wedge mnid_1(id1) \wedge mnid_1(id2) \wedge (\exists a : (prs_1(a)(id1)`1 > 0 \wedge sumin) \vee (prs_1(id)(med)`1 > 0 \wedge sumin)))$

Abstract State	¬ error	error
mind(i)	NC	NC
ptid(i)	NC	NC
mdid(i)	NC	NC
rxid(i)	NC	NC
mn(i)	NC	NC
mns(i)	NC	NC
mdpt(i)	NC	NC
rx(i)	NC	NC
prs(i)	NC	NC
di(i)	$di_1 \cup (id1, id2) \wedge di_1 \cup (id2, id1)$	NC
gs(i)	NC	NC
dpr(i)	NC	NC
r(i)	ok	error

Figure 10: Function Table for add_interaction(id1, id2)

10. Validation

todo

```
***
*** top (23:34:28 11/15/2015)
*** Generated by proveit - ProofLite-6.0.9 (3/14/14)
*** Trusted Oracles
***   MetiTarski: MetiTarski Theorem Prover via PVS proof rule metit
***
 Proof summary for theory top
    Theory totals: 0 formulas, 0 attempted, 0 succeeded (0.00 s)

 Proof summary for theory Time
    r2d_TCC1.............................proved - complete   [shostak](0.23 s)
    d2r_TCC1.............................proved - complete   [shostak](0.03 s)
    held_for_TCC1........................proved - complete   [shostak](0.08 s)
    Theory totals: 3 formulas, 3 attempted, 3 succeeded (0.33 s)

 Proof summary for theory isolette
    c_md_ft_TCC1.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC2.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC3.........................proved - complete   [shostak](0.05 s)
    c_md_ft_TCC4.........................proved - complete   [shostak](0.10 s)
    c_md_ft_TCC5.........................proved - complete   [shostak](0.06 s)
    c_md_ft_TCC6.........................proved - complete   [shostak](0.03 s)
    c_md_ft_TCC7.........................proved - complete   [shostak](0.02 s)
    c_md_ft_TCC8.........................proved - complete   [shostak](0.02 s)
    c_md_ft_TCC9.........................proved - complete   [shostak](0.02 s)
    c_td_ft_TCC1.........................proved - complete   [shostak](0.01 s)
    c_hc_ft_TCC1.........................proved - complete   [shostak](0.07 s)
    c_hc_ft_TCC2.........................proved - complete   [shostak](0.11 s)
    c_hc_ft_TCC3.........................proved - complete   [shostak](0.07 s)
    c_hc_ft_TCC4.........................proved - complete   [shostak](0.05 s)
    c_hc_ft_TCC5.........................proved - complete   [shostak](0.03 s)
    c_al_ft_TCC1.........................proved - complete   [shostak](0.03 s)
    c_al_ft_TCC2.........................proved - complete   [shostak](0.04 s)
    c_al_ft_TCC3.........................proved - complete   [shostak](0.00 s)
    c_al_ft_TCC4.........................proved - complete   [shostak](0.07 s)
    c_al_ft_TCC5.........................proved - complete   [shostak](0.04 s)
    c_al_ft_TCC6.........................proved - complete   [shostak](0.03 s)
    inv_hc_holds.........................proved - complete   [shostak](0.34 s)
    inv_al_holds.........................proved - complete   [shostak](2.71 s)
    Theory totals: 23 formulas, 23 attempted, 23 succeeded (3.98 s)

Grand Totals: 26 proofs, 26 attempted, 26 succeeded (4.32 s)
```

Figure 11: Validated Isolette

You must also provide and prove in PVS one important safety invariant for the heat control *c_hc* and one important safety invariant for the alarm control *c_al*.

Include the PVS sources in the appendix to this document but summarize the proofs here (top.summary).

11. Use Cases

See Section A2 of [**?**] for some use cases. The use cases need to be adapted to the revised descriptions of the previous sections of this document.

12. Acceptance Tests

In this section, the use cases have to be converted into precise acceptance tests (using the function table to describe pre/post conditions) to be run when the design and implementation are complete.

13. Traceability

Matrix to show which acceptance tests passed, and which R-descriptions they checked.

14. Glossary

The definition of important terms is placed in this section. You are not required to complete this.

A. Additional Requirements

REQ7	In *fail* mode, the controller shall only return to *normal* mode if • The sensor is working and • The operator controls are working	In *fail* mode the controller shall return to *normal* mode when *m_st* returns "valid"

REQ8	In any mode, the controller will transition to the *off* mode if the nurse turns the switch off.	The controller will transition to *off* mode from any mode if *m_sw* becomes *off*

B. Isolette PVS

```
isolette[delta:posreal]: THEORY
BEGIN

  %% Import timing resolution
  importing Time[delta]
  i: VAR DTIME

  %% TYPE declarations
  SWITCH: TYPE = {on, off}
  MSTATE: TYPE =  {valid, invalid}
  CONTROL: TYPE = {on, off}
  STATE: TYPE = {off, init, normal, failed}
  ERROR: TYPE = {ok, invalid, config, low, high}
  DISPLAY: TYPE = {i: nat | i = 0 OR (68 <= i AND i <= 105)}
        CONTAINING 0
  ALARM: TYPE = {on, off}
  TM: TYPE+ = {r: real | r >= 68.0 AND r <= 105.0} CONTAINING 68.0
  DL: TYPE+ = {i: nat | i >= 97 AND i <= 99} CONTAINING 97
  DH: TYPE+ =  {i: nat | i >= 98 AND i <= 100} CONTAINING 98
  AL: TYPE+ = {i: nat | i >= 93 AND i <= 98} CONTAINING 93
  AH: TYPE+ = {i: nat | i >= 99 AND i <= 103} CONTAINING 99

  %% Monitored Variables
  m_tm: [DTIME -> TM]
  m_dl: [DTIME -> DL]
  m_dh: [DTIME -> DH]
  m_al: [DTIME -> AL]
  m_ah: [DTIME -> AH]
  m_st: [DTIME -> MSTATE]
  m_sw: [DTIME -> SWITCH]

  %% Controlled Variables
  c_hc: [DTIME -> CONTROL]
  c_td: [DTIME -> DISPLAY]
  c_al: [DTIME -> ALARM]
  c_md: [DTIME -> STATE]
  c_ms: [DTIME -> ERROR]
```

```
al_on(i): bool = c_al(i) = on %% Alarm is on

% General Function table conditions
c1(i): bool = m_st(i) = valid
c2(i): bool = m_dl(i) <= m_tm(i) <= m_dh(i)
c3(i): bool = m_al(i) < m_dl(i) < m_dh(i) < m_ah(i)
c4(i): bool = c1(i) AND c2(i) AND c3(i)
c5(i): bool = m_tm(i) <= m_al(i) + 0.5
c6(i): bool = m_tm(i) <= m_al(i) - 0.5
c7(i): bool = c1(i) AND c3(i) AND m_al(i) < m_tm(i) < m_ah(i)
c8(i): bool = NOT c1(i) OR NOT c3(i) OR c5(i) OR c6(i)
held_for(i): bool = held_for(al_on, 10)(i)

% Mode Function Table
c_md_ft(i): bool =
COND
      i = 0 ->  c_md(i) = off,
      i > 0 ->
      COND
              m_sw(i) = off -> c_md(i) = off,
              m_sw(i) = on ->
              COND
                      c_md(i-1) = off -> c_md(i) = init,
                      c_md(i-1) = normal OR c_md(i-1) = failed ->
                      COND
                              NOT c1(i) -> c_md(i) = failed,
                              c1(i) -> c_md(i) = normal
                      ENDCOND,
                      c_md(i-1) = init ->
                      COND
                              NOT c4(i) ->  c_md(i) = normal,
                              c4(i) ->  c_md(i) = init
                       ENDCOND
              ENDCOND
      ENDCOND
ENDCOND
```

```
% Temperature Display Function Table
c_td_ft(i): bool =
COND
      c_md(i) = normal -> c_td(i) = m_tm(i),
      NOT c_md(i) = normal -> c_td(i)= 0
ENDCOND

% Heat Control Function Table
c_hc_ft(i): bool =
          COND
              i = 0 -> c_hc(i) = off,
              i > 0 ->
                COND
                  c_md(i) = off OR (NOT c1(i))
                    OR (NOT c3(i)) -> c_hc(i) = off,
                 (NOT c_md(i) = off) AND c1(i) AND c3(i) ->
                       COND
                             c2(i) ->  c_hc(i) =  c_hc(i-1),
                                   m_tm(i) < m_dl(i)
                                     -> c_hc(i) = on,
                                   m_tm(i) > m_dh(i)
                                     -> c_hc(i) = off
                       ENDCOND
                  ENDCOND
          ENDCOND
```

```
% Alarm Function Table
c_al_ft(i): bool =
COND
      i = 0 ->  c_al(i) = off,
      i > 0 ->
      COND
              c_al(i-1) = off ->
              COND
                      c7(i) ->  c_al(i) =  c_al(i-1),
                      NOT c7(i) -> c_al(i) = on
              ENDCOND,
              c_al(i-1) = on ->
              COND
                      c8(i) ->  c_al(i) =  c_al(i-1),
                      NOT c8(i) ->
                      COND
                              held_for(i) -> c_al(i) = off,
                              NOT held_for(i) -> c_al(i) = on
                      ENDCOND
               ENDCOND
      ENDCOND
ENDCOND

% Message Display Function Table
c_ms_ft(i): bool =
          IF m_st(i) = invalid THEN c_ms(i) = invalid
          ELSIF NOT c3(i) THEN c_ms(i) = config
          ELSIF m_tm(i) < m_al(i) THEN c_ms(i) = low
          ELSIF m_tm(i) > m_ah(i) THEN c_ms(i) = high
          ELSE c_ms(i) = ok
          ENDIF


% Isolette Specification
isolette(i): bool = c_hc_ft(i) AND c_td_ft(i) AND c_al_ft(i)
      AND c_md_ft(i) AND c_ms_ft(i)
```

```
  % Checks
  inv_hc(i): bool = NOT (c1(i) AND c3(i)) IMPLIES c_hc(i) = off
  inv_hc_holds: CONJECTURE (FORALL i: isolette(i)) =>
                                (FORALL i: inv_hc(i))

  inv_al(i): bool = i > 0 AND (
    (NOT al_on(i-1) AND NOT c7(i))
    OR (al_on(i-1) AND c8(i))
    OR (al_on(i-1) AND NOT c8(i) AND NOT held_for(i))
  ) IMPLIES c_al(i) = on
  inv_al_holds: CONJECTURE (FORALL i: i>0 IMPLIES isolette(i))
                               => (FORALL i: i>0 IMPLIES inv_al(i))

END isolette
```