EECS4312 Isolette Assignment

Drew Noel (cse23217@cse.yorku.ca)
Yuval Alter (cse23163@cse.yorku.ca)

November 13, 2016

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: cse23217

Keep track of your revisions in the table below.

Revisions

Date	Revision	Description
October 29th	1.0	Initial requirements document

Requirements Document:
Temperature control for an Isolette

Contents

List of Figures

List of Tables

1 System Overview

The System Under Development (SUD) is a computer controller for the thermostat of an Isolette.[1] An Isolette is an incubator for for an infant that provides controlled temperature, humidity and oxygen (Fig. 1). Isolettes are used extensively in Neonatal Intensive Care Units for the care of premature infants.

This requirements document is specifically for the control of temperature. The purpose of the Isolette computer controller is to maintain the air temperature of an Isolette within a desired range. It senses the current temperature of the Isolette and turns the heat source on and off to warm the air as needed. If the temperature falls too far below or rises too far above the desired temperature range, it activates an alarm to alert the nurse. The system allows the nurse to set the desired temperature range and to set the alarm temperature range outside the desired temperature range of which the alarm should be activated. This requirements documents follows the specification in [1] (Appendix A) except where noted.

Figure 1: Isolette

[1]The image in Fig 1 is from: `www.nufer-medical.ch`.

2 Context Diagram

See Fig. A-1 in [1]. The System Under Description (SUD) is a computer *controller* to regulate the temperature of the Isolette. Everything else including the Operator Interface (described in [1]) is in the ecosystem (i.e. in the environment of the controller). The monitored variables and controlled variables for the controller are in Table 1 and Table 2, respectively. For clarity, simplicity and safety, there are some differences between the specifications in this document and the descriptions in [1].[2]

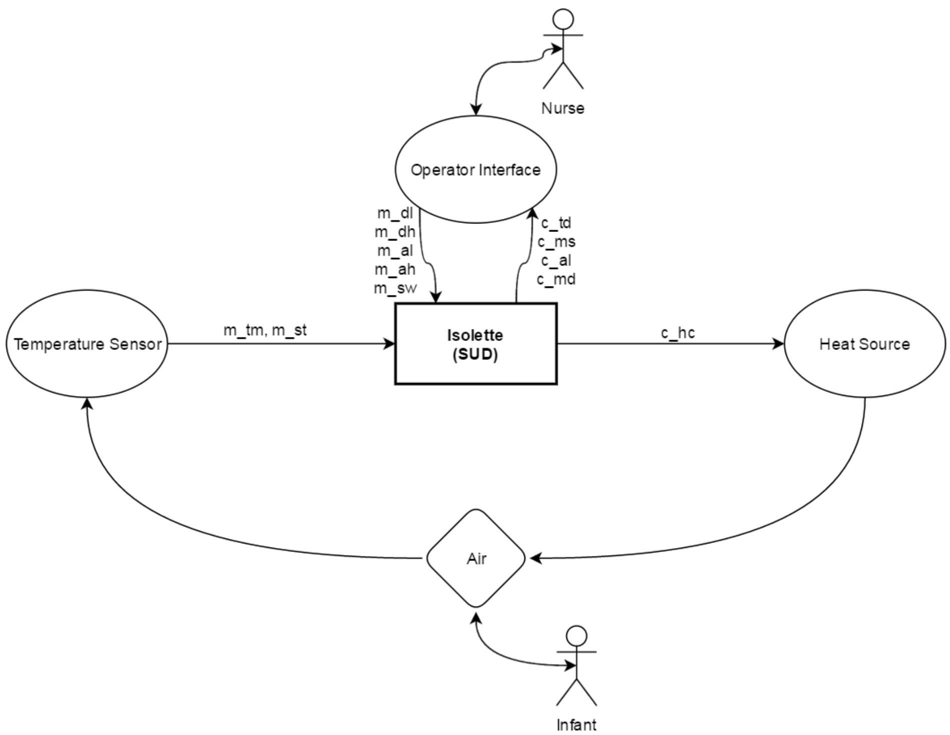


Figure 2: SUD Diagram For The Isolette

3 Goals

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

- G1—The Infant should be kept at a safe and comfortable temperature.
- G2—The Nurse should be warned if the Infant becomes too hot or too cold.

[2]Documented in the write-up to this assignment: `assign1-spec.pdf`.

- G3—The cost of manufacturing the computer controller for the thermostat should be as low as possible.

4 Monitored Variables

The monitored variables are a subset of those described in [1].[3] There is a single status variable m_st that is *invalid* whenever any one of the operator inputs or temperature sensor are in a failed state. Otherwise types and ranges are as in [1].

Name	Type	Range	Units	Physical Interpretation
m_tm	$\mathbb{R}$	68.0 .. 105.0	°F	actual temperature of Isolette air temperature from sensor
m_dl	$\mathbb{Z}$	97 .. 99	°F	desired lower temperature set by operator
m_dh	$\mathbb{Z}$	98 .. 100	°F	desired higher temperature set by operator
m_al	$\mathbb{Z}$	93 .. 98	°F	lower alarm temperature set by operator
m_ah	$\mathbb{Z}$	99 .. 103	°F	higher alarm temperature set by operator
m_st	Enumerated	{valid, invalid}		status of sensor and operator settings
m_sw	Enumerated	{on, off}		switch set by operator

Table 1: Monitored Variables

5 Controlled Variables

The controlled variables are a subset of those described in [1].[4] In addition, there is a mode display c_md and a message display c_ms.[5]

[3] With some change of nomenclature. Monitored variables have an "m" prefix.
[4] With some change of nomenclature. Controlled variables have a "c" prefix.
[5] The mode "off" is added to that of Fig. A-4 in [1], and the mode transitions have been changed.

Name	Type	Range	Units	Physical Interpretation
c_hc	Enumerated	{on, off}		heat control: command to turn heat source on or off
c_td	$\mathbb{Z}$	$\{0\} \cup \{68..105\}$	°F	displayed temperature of Isolette (zero when Isolette is off)
c_al	Enumerated	{off, on}		sound alarm to call nurse
c_md	Enumerated	{off, init, normal, failed}		mode of Isolette operation (failed if $m\_st = invalid$)
c_ms	Enumerated	{ok, invalid_sensor, invalid_alarm_limits, alarm_triggered}		messages to display to nurse

Table 2: Controlled Variables

6 Mode Diagram

Provide a statechart for the mode-diagram and provide rationale for the statechart. The statechart can be found below:

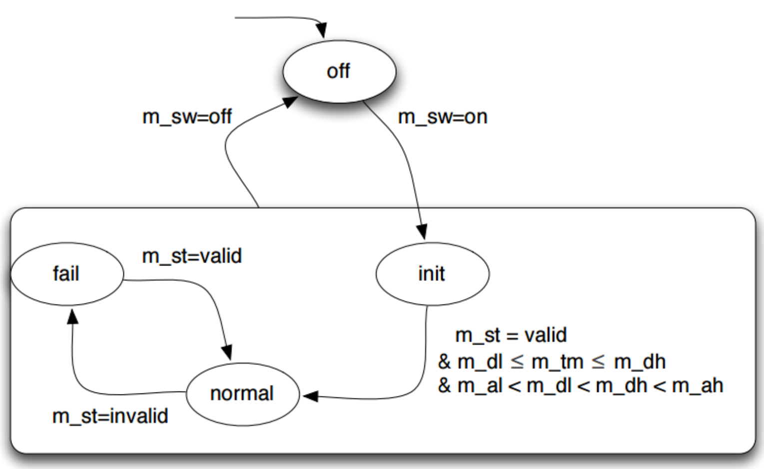


Figure 3: The modes of the isolette

Rationale: The isolette might not be on, or the operator has not yet configured the islotte, or there is a hardware failue. These conditions are continuosly checked and if the conditions pass, then the isolette begins operating in the normal mode with no errors.

7 E/R-descriptions

REQ1	The *controller* shall operate in one of four modes: *off*, *init*, *normal* and *fail*.	See statechart in Fig. 3.

Rationale: The controller is either: off, awaiting initial valid inputs, in a valid state, or in an invalid state. These are the only possible states, and the transitions are given in Fig. 3.

REQ2	In the *normal* mode, the temperature controller shall maintain current temperature inside the Isolette within a set temperature range (the *desired* range).	The *desired* temperature range is *m_dl..m_dh*. If the current temperature *m_th* is outside this range, the controller shall turn the heater on or off via the controlled variable *m_hc* to maintain the desired state.

Rationale: The *desired temperature range* will be set by the nurse to the desired range based on the infant's weight and health. The controller shall maintain the current temperature within this range under normal operation.

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: The following relevant hazard was identified through the safety assessment process:

1. H1: Prolonged exposure of Infant to unsafe heat or cold;
2. *Classification*: catastrophic
3. *Probability*: $< 10^{-9}$ per hour of operation

REQ4	Once the alarm is activated, it becomes deactivated in one of two ways: • The nurse turns off the Isolette • The alarm has lasted for 10 seconds, and after 10 seconds or more the alarm conditions are removed.	If the Isolette is powered on, the alarm will sound for 10 seconds, or until the alarm conditions no longer exist, whichever is later.

Rationale: Avoid transient alarms, and only turn off the alarm once the conditions that triggered the alarm have been removed.

REQ5	The isolette shall check the user input and provide an error message if the low or high alarm limits are not properly set	The monitored variables m_dl, m_dh, m_al, and m_dh are cross checked, see table 6

Rationale: The operator might make a mistake configuring the isolette, an error should appear if they do so.

REQ6	The isolette shall check whether the heat sensor is valid and provide an error if it is not.	The monitored variable m_st is checked, see table 6

Rationale: The sensor might fail, the operator should know of any failures so the hardware can be replaced.

ENV7	The current temperature received from the sensor is a real number in the range 68.0°F to 105.0°F.	The environment assures that the sensor will be in the given range, and it is not required to account for temperatures outside this range.

Rationale: The sensor has some given range due to its nature, which allows for restriction of the expected input.

ENV8	The desired and alarm temperatures received from the operator are all in increments of 1°F.	The inputs for the temperatures will be limited to integers, due to the nature of the environment

Rationale: The given inputs have some given granularity, imposed requirements on the SUD.

ENV9	Failure of equipment will change the monitored variable m_st to invalid	The hardware environment ensures that monitored variable can be trusted to be true

Rationale: This allows the isolette to act predictably.

8 Abstract variables needed for the Function Table

1. $init\_cond = m\_st(i) = valid \wedge m\_dl(i) <= m\_tm(i) \wedge m\_tm(i) <= m\_dh(i) \wedge m\_al(i) < m\_dl(i) \wedge m\_dl(i) < m\_dh(i) \wedge m\_dh(i) < m\_ah(i)$
2. $cond\_alarm = EXISTS(should\_alarm : [DTIME- > BOOL], low\_alarm : [DTIME- > BOOL], high\_alarm : [DTIME- > BOOL]) :$
$limits\_alarm\_req(m\_ah, m\_tm, m\_al, eps, high\_alarm, should\_alarm, low\_alarm)$
3. $temp\_hits\_high = m\_tm(i) > m\_dh(i)$
4. $temp\_hits\_low = m\_tm(i) < m\_dl(i)$
5. $temp\_no\_change\_zone = m\_tm(i) <= m\_dh(i) \wedge m\_tm(i) => m\_dl(i)$
6. $temp\_out\_of\_bounds = m\_al(i) >= m\_ah(i) - 2 * eps(i)$
7. $temp\_in\_proper\_range = m\_al(i) < m\_ah(i) - 2 * eps(i)$
8. $overlap = overlap?(m\_dl(i), m\_dh(i))$

9 Function Table

9.1 Function Table for heat control: c_hc

[9]Defined in Abstract Variable 3
[9]Defined in Abstract Variable 4
[9]Defined in Abstract Variable 5
[9]Defined in Abstract Variable 8

				c_hc
$i = 0$				false
$i > 0$	$c\_md(i) = \text{off}$			false
	$c\_md(i) \neq \text{off}$	$\neg overlap$ [6]	$temp\_hits\_high$ [7]	false
			$temp\_hits\_low$ [8]	true
			$temp\_no\_change\_zone$ [9]	$c\_hc(i-1)$
		$overlap$		$c\_hc(i) = c\_hc(i-1)$

Table 3: Function Table for heat control: c_hc

9.2 Function Table for displayed temperature: c_td

			c_td
$i = 0$			0
$i > 0$	$m\_sw(i) = \text{off}$		0
	$m\_sw(i) = \text{on}$	$m\_st(i) = \text{invalid}$	0
		$m\_st(i) = \text{valid}$	(NAT) $(m\_tm(i))$

Table 4: Function Table for displayed temperature: c_td

9.3 Function Table for alarm: c_al

				c_al
$i = 0$				false
$i > 0$	$cond\_alarm$[10]			$c\_al(i-1)$
	$\neg cond\_alarm$	$sound\_alarm(i)$		true
		$\neg sound\_alarm(i)$	$held\_for(c\_al, 10)(i-10)$	false
			$\neg held\_for(c\_al, 10)(i-10)$	$c\_al(i-1)$

Table 5: Function Table for alarm: c_al

9.4 Function Table for Isolette mode: c_md

[10] Defined in Abstract Variable 2
[11] Defined in Abstract Variable 1

<table>
<tr><td colspan="4"></td><td>c_md</td></tr>
<tr><td colspan="4">$i = 0$</td><td>off</td></tr>
<tr><td rowspan="9">$i > 0$</td><td colspan="3">$m\_sw = \text{off}$</td><td>off</td></tr>
<tr><td rowspan="8">$m\_sw = \text{on}$</td><td colspan="2">$c\_md(i-1) = \text{off}$</td><td>init</td></tr>
<tr><td rowspan="2">$c\_md(i-1) = \text{init}$</td><td>$init\_cond$ [11]</td><td>normal</td></tr>
<tr><td>$\neg init\_cond$</td><td>init</td></tr>
<tr><td rowspan="2">$c\_md(i-1) = \text{normal}$</td><td>$m\_st(i) = \text{invalid}$</td><td>failed</td></tr>
<tr><td>$m\_st(i) = \text{valid}$</td><td>normal</td></tr>
<tr><td rowspan="2">$c\_md(i-1) = \text{failed}$</td><td>$m\_st(i) = \text{valid}$</td><td>normal</td></tr>
<tr><td>$m\_st(i) = \text{invalid}$</td><td>failed</td></tr>
</table>

Table 6: Function Table for Isolette mode: c_md

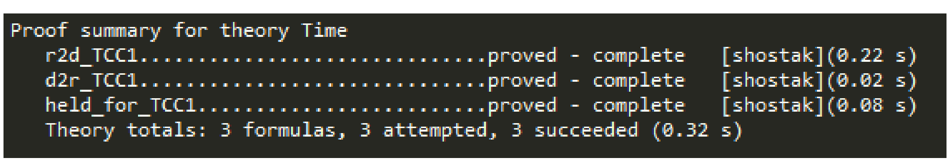
9.5 Function Table for displayed message: c_ms

<table>
<tr><td colspan="4"></td><td>c_ms</td></tr>
<tr><td colspan="4">$i = 0$</td><td>ok</td></tr>
<tr><td rowspan="4">$i > 0$</td><td colspan="3">$c\_al(i) = \text{true}$</td><td>$alarm\_triggered$</td></tr>
<tr><td rowspan="3">$c\_al(i) = \text{false}$</td><td colspan="2">$m\_st(i) = \text{invalid}$</td><td>$invalid\_sensor$</td></tr>
<tr><td rowspan="2">$m\_st(i) = \text{valid}$</td><td>$temp\_out\_of\_bounds$ [12]</td><td>$invalid\_alarm\_limits$</td></tr>
<tr><td>$temp\_in\_proper\_range$ [13]</td><td>ok</td></tr>
</table>

Table 7: Function Table for displayed message: c_ms

10 Validation

Proof of completeness and disjointness and validation of the requirements using PVS.



```
Proof summary for theory Time
   r2d_TCC1..............................proved - complete   [shostak](0.22 s)
   d2r_TCC1..............................proved - complete   [shostak](0.02 s)
   held_for_TCC1.........................proved - complete   [shostak](0.08 s)
   Theory totals: 3 formulas, 3 attempted, 3 succeeded (0.32 s)
```

Figure 4: Proof summary for Time

[13]Defined in Abstract Variable 6
[13]Defined in Abstract Variable 7

```
Proof summary for theory Hysteresis
   BOOL_TCC1.............................proved - complete   [shostak](0.00 s)
   hysteresis_st_TCC1....................proved - complete   [shostak](0.02 s)
   hysteresis_st_TCC2....................proved - complete   [shostak](0.01 s)
   hysteresis_st_TCC3....................proved - complete   [shostak](0.01 s)
   hysteresis_req_TCC1...................proved - complete   [shostak](0.10 s)
   hysteresis_req_TCC2...................proved - complete   [shostak](0.06 s)
   correct_hysteresis_st.................proved - complete   [shostak](0.16 s)
   hysteresis_req_pre_i_TCC1.............proved - complete   [shostak](0.09 s)
   hysteresis_req_pre_i_TCC2.............proved - complete   [shostak](0.07 s)
   correct_hysteresis_st_pre_TCC1........proved - complete   [shostak](0.05 s)
   correct_hysteresis_st_pre.............proved - complete   [shostak](0.14 s)
   hysteresis_req_pre_TCC1...............proved - complete   [shostak](0.10 s)
   hysteresis_req_pre_TCC2...............proved - complete   [shostak](0.06 s)
   correct_hysteresis_st_pre2............proved - complete   [shostak](0.13 s)
   Theory totals: 14 formulas, 14 attempted, 14 succeeded (1.02 s)
```

Figure 5: Proof summary for Hysteresis

```
Proof summary for theory isolette
   modes_ft_TCC1.........................proved - complete   [shostak](0.03 s)
   modes_ft_TCC2.........................proved - complete   [shostak](0.06 s)
   modes_ft_TCC3.........................proved - complete   [shostak](0.05 s)
   modes_ft_TCC4.........................proved - complete   [shostak](0.05 s)
   modes_ft_TCC5.........................proved - complete   [shostak](0.04 s)
   modes_ft_TCC6.........................proved - complete   [shostak](0.06 s)
   modes_ft_TCC7.........................proved - complete   [shostak](0.11 s)
   modes_ft_TCC8.........................proved - complete   [shostak](0.06 s)
   modes_ft_TCC9.........................proved - complete   [shostak](0.03 s)
   modes_ft_TCC10........................proved - complete   [shostak](0.02 s)
   modes_ft_TCC11........................proved - complete   [shostak](0.02 s)
   modes_ft_TCC12........................proved - complete   [shostak](0.03 s)
   heater_ft_TCC1........................proved - complete   [shostak](0.06 s)
   heater_ft_TCC2........................proved - complete   [shostak](0.08 s)
   heater_ft_TCC3........................proved - complete   [shostak](0.05 s)
   heater_ft_TCC4........................proved - complete   [shostak](0.05 s)
   heater_ft_TCC5........................proved - complete   [shostak](0.03 s)
   heater_ft_TCC6........................proved - complete   [shostak](0.04 s)
   heater_ft_TCC7........................proved - complete   [shostak](0.01 s)
   heater_ft_TCC8........................proved - complete   [shostak](0.02 s)
   alarm_ft_TCC1.........................proved - complete   [shostak](0.22 s)
   alarm_ft_TCC2.........................proved - complete   [shostak](0.27 s)
   alarm_ft_TCC3.........................proved - complete   [shostak](0.21 s)
   alarm_ft_TCC4.........................proved - complete   [shostak](0.15 s)
   alarm_ft_TCC5.........................proved - complete   [shostak](0.15 s)
   dispTemp_ft_TCC1......................proved - complete   [shostak](0.04 s)
   dispTemp_ft_TCC2......................proved - complete   [shostak](0.03 s)
   dispTemp_ft_TCC3......................proved - complete   [shostak](0.03 s)
   dispTemp_ft_TCC4......................proved - complete   [shostak](0.02 s)
   dispTemp_ft_TCC5......................proved - complete   [shostak](0.01 s)
   dispTemp_ft_TCC6......................proved - complete   [shostak](0.01 s)
   msg_ft_TCC1...........................proved - complete   [shostak](0.13 s)
   msg_ft_TCC2...........................proved - complete   [shostak](0.09 s)
   msg_ft_TCC3...........................proved - complete   [shostak](0.02 s)
   usecase_a_2_1.........................proved - complete   [shostak](0.84 s)
   usecase_a_2_3.........................proved - complete   [shostak](0.62 s)
   t1....................................proved - complete   [shostak](0.23 s)
   t2....................................proved - complete   [shostak](0.26 s)
   t3....................................proved - complete   [shostak](0.21 s)
   t4_TCC1...............................proved - complete   [shostak](0.47 s)
   t4....................................proved - complete   [shostak](0.13 s)
   t5....................................proved - complete   [shostak](0.29 s)
   t6....................................proved - complete   [shostak](0.13 s)
   Theory totals: 43 formulas, 43 attempted, 43 succeeded (5.43 s)
```

Figure 6: Proof summary for isolette

```
Proof summary for theory Limits_Alarm
   high_alarm_req_TCC1...................proved - complete   [shostak](0.05 s)
   high_alarm_req_TCC2...................proved - complete   [shostak](0.07 s)
   high_alarm_req_TCC3...................proved - complete   [shostak](0.06 s)
   high_alarm_req_TCC4...................proved - complete   [shostak](0.02 s)
   high_alarm_req_TCC5...................proved - complete   [shostak](0.01 s)
   low_alarm_req_TCC1....................proved - complete   [shostak](0.05 s)
   low_alarm_req_TCC2....................proved - complete   [shostak](0.08 s)
   low_alarm_req_TCC3....................proved - complete   [shostak](0.05 s)
   correct_limits_alarm_fbd..............proved - complete   [shostak](0.37 s)
   req_entails_inv.......................proved - complete   [shostak](0.12 s)
   limits_alarm_req2_TCC1................proved - complete   [shostak](0.33 s)
   limits_alarm_req2_TCC2................proved - complete   [shostak](0.10 s)
   limits_alarm_fbd_pre_TCC1.............proved - complete   [shostak](0.22 s)
   limits_alarm_fbd_pre_TCC2.............proved - complete   [shostak](0.05 s)
   Theory totals: 14 formulas, 14 attempted, 14 succeeded (1.58 s)
```

Figure 7: Proof summary for Limits_Alarm

```
Grand Totals: 74 proofs, 74 attempted, 74 succeeded (8.35 s)
```

Figure 8: Summary of all proofs

11 Use Cases

Two usecases were derived from [1] Appendix A

- 1. The isolette is off and the heat control is off
  2. The nurse turns on the isolette
  3. The isolette enters initialization mode and awaits configuration
  4. The nurse sets the desired limits
  5. The isolette enters normal mode and turns on the heat control
  6. The isolette runs for some time and then the nurse turns off the isolette
  7. The isolette powers off
- 1. The isolette is running with the temperature within the range
  2. The temperature exceeds the upper limit of the desired range
  3. The isolette turns off the heat control
  4. The temperature lowers and eventually drops below the lower limit of the desired range
  5. The isolette turns the heat control on

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.

```
The first acceptance test checks whether the isolette off switch is
    ↪ acting as predicted per REQ1.
Precondition: switch is off
Postcondition: isolette is in off mode
```

```
The second acceptance test checks whether the isolette can properly be
    ↪ initialized in the right state as per REQ1.$Precondition: isolette
    ↪ is initializing
Postcondition: isolette is on
```

```
The third acceptance test checks whether the isolette failing means that
    ↪ the sensor has failed stated per REQ1.
Precondition: isolette is in fail mode
Postcondition: sensor has indeed failed
```

```
The fourth acceptance test checks whether the isolette ringing for 10
    ↪ seconds or not being turned off through the messages has indeed
    ↪ turned off the alarm as described per REQ4.
Precondition: 10 seconds have passed and there is no alarm_triggered
    ↪ message
Postcondition: isolette is not sounding alarm
```

```
The fifth acceptance test checks whether the isolette displays an alarm\
    ↪ _triggered message when the alarm is triggered as per REQ5.
Precondition: alarm triggered
Postcondition: isolette displays alarm_triggered message
```

```
The sixth acceptance test checks whether the off switch turns the heater
    ↪ off as per REQ6.
Precondition: switch is off
Postcondition: heater is turned off
```

13 Traceability

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

Test	Requirements Testing	Passed
t1	R1	Yes
t2	R1	Yes
t3	R1	Yes
t4	R4	Yes
t5	R5	Yes
t6	R6	Yes

Table 8: Acceptance tests traceability matrix

14 Appendix A – PVS source

```
isolette: THEORY
BEGIN
        delta: posreal = 1.0
        importing Time[delta]
        importing Limits_Alarm[delta]
```

```
% Defined Types
% Monitored Variable Types: The name for the first set
   ↪ corresponds to
%                the variable name and it's type (ie. temperature)
TM_TEMP: TYPE = {x: real |  68.0 <= x AND x <= 105.0}
DL_TEMP: TYPE = {y: nat | 97 <= y AND y <= 99}
DH_TEMP: TYPE = {y: nat | 98 <= y AND y <= 100}
AL_TEMP: TYPE = {y: nat | 93 <= y AND y <= 98}
AH_TEMP: TYPE = {y: nat | 99 <= y AND y <= 103}
SENSOR: TYPE = {valid, invalid}

% Controlled Variables Types
DISP_TEMP: TYPE = {y: nat | y = 0 OR (68 <= y AND y <= 105)}
MODE: TYPE = {off, init, normal, failed}
MSG: TYPE = {ok, invalid_sensor, invalid_alarm_limits,
   ↪ alarm_triggered}
EPS_RANGE: TYPE = subrange(1,1)

% Monitored Variables
m_tm: VAR [DTIME->TM_TEMP] %temp monitored
m_dl: VAR [DTIME->DL_TEMP] %desired low temp
m_dh: VAR [DTIME->DH_TEMP] %desired high temp
m_al: VAR [DTIME->AL_TEMP] %desired low alarm temp
m_ah: VAR [DTIME->AH_TEMP] %desired high alarm temp
m_st: VAR [DTIME->SENSOR] %status of temp sensor
m_sw: VAR [DTIME->BOOL] %switch set by operator

% Controlled Variables:
c_hc: VAR [DTIME->BOOL] %heat control
c_td: VAR [DTIME->DISP_TEMP] %displayed isolette temp
c_al: VAR [DTIME->bool] %sound alarm
c_md: VAR [DTIME->MODE] %mode of isolette
c_ms: VAR [DTIME->MSG] %message to display

% Constant Variables:
eps: VAR [DTIME->EPS_RANGE]

% Environmental Assumptions

% Function Tables
% Helper function table overlap_desired?
overlap?(low: DL_TEMP, high: DH_TEMP): bool =
    high <= low

% Funtion table for modes: c_md
modes_ft(c_md, m_sw, m_st, m_dl, m_tm, m_dh, m_al, m_ah): bool =
  FORALL (i:DTIME):
```

```
    COND
      i = 0 ->
        c_md(i) = off,
      i > 0 ->
        COND
        m_sw(i) = false ->
          c_md(i) = off,
        m_sw(i) = true ->
          COND
          c_md(i-1) = off ->
            c_md(i) = init,
          c_md(i-1) = init ->
            COND
              m_st(i) = valid AND m_dl(i) <= m_tm(i) AND m_tm(i) <=
                 ↪  m_dh(i) AND m_al(i) < m_dl(i) AND m_dl(i) <
                 ↪ m_dh(i) AND m_dh(i) < m_ah(i) ->
                c_md(i) = normal,
              NOT (m_st(i) = valid AND m_dl(i) <= m_tm(i) AND m_tm(
                 ↪ i) <= m_dh(i) AND m_al(i) < m_dl(i) AND m_dl(i)
                 ↪  < m_dh(i) AND m_dh(i) < m_ah(i)) ->
                c_md(i) = init
            ENDCOND,
          c_md(i-1) = normal ->
            COND
              m_st(i) = invalid ->
                c_md(i) = failed,
              m_st(i) = valid ->
                c_md(i) = normal
            ENDCOND,
          c_md(i-1) = failed ->
            COND
              m_st(i) = valid ->
                c_md(i) = normal,
              m_st(i) = invalid ->
                c_md(i) = failed
            ENDCOND
          ENDCOND
        ENDCOND
    ENDCOND

  % Function table for heat control: c_hc
  heater_ft(c_hc, c_md, m_dl, m_dh, m_tm): bool =
    FORALL (i:DTIME):
    COND
      i = 0 ->
        c_hc(i) = false,
      i > 0 ->
```

```
      COND
        c_md(i) = off -> c_hc(i) = false,
        c_md(i) /= off -> COND
          NOT overlap?(m_dl(i), m_dh(i)) -> COND
            m_tm(i) > m_dh(i) ->
              c_hc(i) = false,
            m_tm(i) < m_dl(i) ->
              c_hc(i) = true,
            m_tm(i) >= m_dl(i) AND m_tm(i) <= m_dh(i) ->
              c_hc(i) = c_hc(i-1)
          ENDCOND,
          overlap?(m_dl(i), m_dh(i)) -> c_hc(i) = c_hc(i-1)
        ENDCOND
      ENDCOND
  ENDCOND

% Function table for alarm: c_al(i)
alarm_ft(c_al, m_al, eps, m_ah, m_tm): bool =
  EXISTS (should_alarm:[DTIME->BOOL], low_alarm:[DTIME->BOOL],
     ↪ high_alarm:[DTIME->BOOL]):
  limits_alarm_req(m_ah, m_tm, m_al, eps, high_alarm,
     ↪ should_alarm, low_alarm) AND
  FORALL (i:DTIME):
    COND
      i = 0 ->
        c_al(i) = 0,
      i > 0 ->
          COND
            should_alarm(i) = true ->
              c_al(i),
            should_alarm(i) = false AND held_for(c_al, 10)(i-1)
               ↪ ->
              NOT c_al(i),
            should_alarm(i) = false AND NOT held_for(c_al, 10)(i
               ↪ -1) ->
              c_al(i) = c_al(i-1)
          ENDCOND
    ENDCOND


% Function table for display temperature: c_td
dispTemp_ft(c_td, m_sw, m_st, m_tm): bool =
  FORALL (i:DTIME):
  COND
    i = 0 ->
      c_td(i) = 0,
    i > 0 ->
```

```
        COND
          m_sw(i) = false ->
            c_td(i) = 0,
          m_sw(i) = true ->
            COND
              m_st(i) = invalid ->
                c_td(i) = 0,
              m_st(i) = valid ->
                c_td(i) = floor(m_tm(i) + 0.5)
            ENDCOND
        ENDCOND
    ENDCOND


  % Function table for error message: c_ms
  msg_ft(c_ms, c_al, eps, m_tm, m_dl, m_dh, m_al, m_ah, m_st): bool
     ↪  =
    FORALL (i:DTIME):
    COND
      i = 0 ->
        c_ms(i) = ok,
      i > 0 ->
        COND
          c_al(i) ->
            c_ms(i) = alarm_triggered,
          NOT c_al(i) ->
          COND
            m_st(i) = invalid ->
          c_ms(i) = invalid_sensor,
            m_st(i) = valid AND m_al(i) >= m_ah(i) - 2 * eps(i) ->
          c_ms(i) = invalid_alarm_limits,
            m_st(i) = valid AND m_al(i) < m_ah(i) - 2 * eps(i) ->
          c_ms(i) = ok
          ENDCOND
        ENDCOND
    ENDCOND

  % Master FT
  isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah, m_st, m_sw, eps, c_hc,
     ↪ c_td, c_al, c_md, c_ms): bool =
    modes_ft(c_md, m_sw, m_st, m_dl, m_tm, m_dh, m_al, m_ah) AND
    heater_ft(c_hc, c_md, m_dl, m_dh, m_tm) AND
    alarm_ft(c_al, m_al, eps, m_ah, m_tm) AND
    dispTemp_ft(c_td, m_sw, m_st, m_tm) AND
    msg_ft(c_ms, c_al, eps, m_tm, m_dl, m_dh, m_al, m_ah, m_st)

  % Use A.2.1 and A.2.2
```

```
usecase_a_2_1: CONJECTURE
  isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah, m_st, m_sw, eps, c_hc
     ↪ , c_td, c_al, c_md, c_ms) AND
  m_tm(0) = 68 AND
  m_sw(1) = true AND    % Nurse turns on the isolette
  m_dl(1) = 97 AND
  m_dh(1) = 100 AND
  m_tm(1) = 68 AND
  m_st(2) = valid AND
  m_dl(2) = 97 AND      % Nurse configures the isolette
  m_al(2) = 96 AND
  m_dh(2) = 100 AND     % Nurse waits for current temperature to
     ↪ reach range
  m_ah(2) = 101 AND
  m_tm(2) = 99 AND
  m_sw(2) = m_sw(1) AND
  m_tm(3) = 98 AND
  m_sw(3) = m_sw(2) AND
  m_sw(4) = false       % Nurse turns off the isolette
  IMPLIES
  c_md(0) = off AND
  c_hc(0) = false AND
  c_hc(1) = true AND    % Isolette powers on the heat
  c_md(1) = init AND    % Isolette awaits configuration
  c_md(2) = normal AND % Isolette enters normal mode (also
     ↪ fulfills A.2.2)
  c_md(4) = off

% Use A.2.3
usecase_a_2_3: CONJECTURE
  isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah, m_st, m_sw, eps, c_hc
     ↪ , c_td, c_al, c_md, c_ms) AND
  c_md(2) = normal AND
  m_tm(3) > m_dl(3) AND m_tm(3) < m_dh(3) AND
  m_tm(4) > m_dh(4) + eps(4) AND
  m_tm(5) < m_dl(5) - eps(5) AND
  m_dh(2) > m_dl(2) AND
  m_dh(3) = m_dh(2) AND m_dh(3) = m_dh(4) AND m_dh(5) = m_dh(4)
     ↪ AND
  m_sw(2) = 1 AND m_sw(3) = 1 AND m_sw(4) = 1 AND m_sw(5) = 1 AND
  m_st(2) = valid AND m_st(3) = m_st(2) AND m_st(4) = m_st(3) AND
     ↪  m_st(5) = m_st(4) AND
  NOT overlap?(m_dl(4), m_dh(4)) AND NOT overlap?(m_dl(5), m_dh
     ↪ (5))
  IMPLIES
  c_hc(4) = 0 AND c_hc(5) = 1
```

```
    % Acceptance tests
    t1: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 0
      IMPLIES
      (m_sw(i) = false IFF c_md(i) = off)

    t2: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 0
      IMPLIES
      (c_md(i) = init IMPLIES m_sw(i) = true)

    t3: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 0
      IMPLIES
      (c_md(i) = failed IMPLIES m_st(i) = invalid)

    t4: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 10
      IMPLIES
      (held_for(c_al, 10)(i-10) AND NOT c_ms(i) = alarm_triggered
         ↪ IMPLIES NOT c_al(i))

    t5: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 0
      IMPLIES
      (c_al(i) IMPLIES c_ms(i) = alarm_triggered)

    t6: CONJECTURE
      FORALL (i:DTIME): isolette_ft(m_tm, m_dl, m_dh, m_al, m_ah,
         ↪ m_st, m_sw, eps, c_hc, c_td, c_al, c_md, c_ms) AND i > 0
      IMPLIES
      (m_sw(i) = false IMPLIES c_hc(i) = false)

END isolette
```

References

[1] David L. Lempia and Steven P. Miller. *Requirements Engineering Management Handbook*. DOT/FAA, Springfield, Virginia, June 2009.