EECS4312 eHealth Project

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

December 8, 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

Revisions

Date	Revision	Description
December 8, 2015	1.0	Initial requirements document

Requirements Document: for Patient care eHealth System

Contents

1. System Overview **5**

2. Context Diagram **6**

3. Goals **7**

4. Monitored Events **8**

5. Controlled Variables **9**

6. Abstract variables needed for the Function Table **9**

7. E/R-descriptions **10**

7.1. Requirements Descriptions 10

7.2. Environmental Descriptions 12

8. Function Tables **14**

8.1. Function Table for eHealth 16

8.2. Function Table for init 16

8.3. Function Table for new_prescription(id, md, pt) 17

8.4. Function Table for add_medicine(id, m, d) 18

8.5. Function Table for remove_medicine(id, med) 19

8.6. Function Table for add_interaction(id1, id2) 20

9. Validation **21**

10. Acceptance Tests **22**

A. eHealth PVS **26**

List of Figures

1. Context Diagram 6

2. Abstract Variables used in Function Tables 9

3. Table of errors and warnings 15

4. Function Table for eHealth 16

5. Table 5. Function Table for init 16

6. Table 6. Function Table for new_prescription(id, doctor, patient) . . . 17

7. Table 7. Function Table for add_medicine(id, medicine, dose) 18
8. Table 8. Function Table for remove_medicine(id, medicine) 19
9. Table 9. Function Table for add_interaction(id1, id2) 20
10. Validated Isolette 21
11. First Acceptance Test 22

List of Tables

1. Monitored Events 8
2. Monitored Types 8
3. Controlled Variables 9

1. System Overview

The System Under Development (SUD) is a eHealth prescriptions management prototype that will eventually be part of a much larger system that manages health prescription records for Ontario. This requirements document is specifically for this prototype that will keep track of prescriptions for patients so that dangerous interactions between medications can be controlled and checked.

To do this, the system will keep will maintain a database of physicians, medications, patients, and patient prescriptions, each with a unique identifier (ID). The system will also maintain records of dangerous interactions between medications. Interactions are defined as undesirable outcomes when certain pairs of medications are taken together. For this system, we are to track these interactions as well as prescriptions that contain medications that interact dangerously.

Physicians are classified either as generalists, or specialists in the database. Generalists are allowed to prescribe medications so long as they do not cause dangerous interactions. Specialists are given discretion in this case - they are allowed to prescribe medications that would cause dangerous reactions, though this is flagged by the system and displayed in a report.

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 3 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 2.

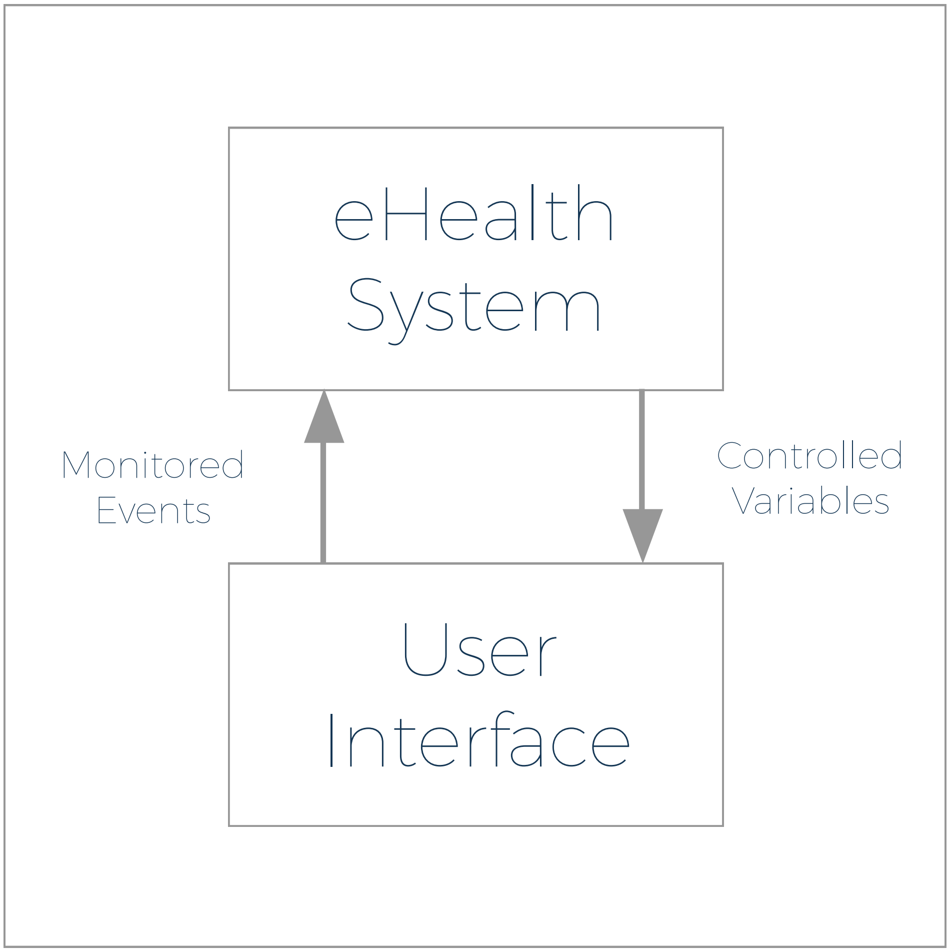


Figure 1: Context Diagram

3. Goals

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

- G1 — The user should be able to add physicians of type {generalist, specialist} with unique IDs
- G2 — The user should be able to add patients with unique IDs
- G3 — The user should be able to add medications with unique names and IDs and a safe dosage range
- G4 — The user should be able to add prescriptions with a unique ID between a doctor and patient
- G5 — The user should be able to add interactions between medications
- G6 — Physicians should be able to prescribe medications with specific dosages, so long as the dosage is within the safe range
- G7 — A Physician must be a specialist to prescribe a medication to a patient when it would create a dangerous interaction with a different already prescribed medication - regardless of the prescription or physician the other medication exists in
- G8 — The system should be able to display a report of all patients prescribed a particular medication
- G9 — The system should be able to display a dangerous prescriptions report that lists all patients with dangerous prescriptions along with the particular medications prescribed to them that interact dangeriously

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)	List all patients prescribed the medication
dpr_q	Print out a dangerous prescription report

Table 1: Monitored Events

Name	Type	Interpretation
PHYSICIAN_TYPE	{gn, sp}	Generalist or Specialist
MEDICATION	[name: NAME; kind: KIND; low: VALUE; hi: VALUE]	Name is unique. See following rows
KIND	{pill, liquid}	Pill or a liquid
DOSE	{mg, cc}	Milligrams or Cubic Centimetres

Table 2: Monitored Types

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 2
Patients	A list of all the Patients currently in the system	See table 2
Medications	A list of all the Medications currently within the system	See table 2
Interactions	A list of all the Interactions currently within the system	See table 2
Prescriptions	A list of all the Prescriptions within the system	See table 2
Error	A message displaying the highest priority error, or ”ok”	See table 2

Table 3: Controlled Variables

6. 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 2: Abstract Variables used in Function Tables

7. E/R-descriptions

7.1. Requirements Descriptions

REQ1	The system will keep track of Physicians, Patients, Medications, and Prescriptions with unique IDs.	See Table 3 for list of abstract states.

Rationale: Physicians and Patients may have non-unique names, and to avoid confusion when creating prescriptions, they must be identified by unique IDs.

REQ2	The system will make a distinction between Physicians that are Generalists and Specialists.	See Table 2 for Monitored Types.

Rationale: The system will not allow *Generalists* to add Medications to prescriptions where they may cause dangerous interactions. The assumption is made that *Specialists* have received more training and it is left up to their discretion to allow Medications with dangerous Interactions to be prescribed.

REQ3	The system will only accept unique Medication names to prevent confusion.	Non-unique Medication names increase the chance of human error.

Rationale: Though Medications are also identified by a unique ID, it is critical that these are not mixed up, and as such it is required that the names are not identical as they may be for Patients or Physicians.

REQ4	The system will keep track of dangerous Interactions between Medications, as well as the order in which they were added.	Tracking dangerous Interactions allow for the system to properly flag them.

Rationale: The system must keep track of dangerous Interactions to enforce rules such as Generalists not being allowed to add Medications that case dangerous interactions. The client specified that they require the interactions within a prescription to be output in the order that the interactions were added to the system.

REQ5	The system will allow Physicians to prescribe Medications to Patients.	Basic functionality of a system tracking Prescriptions.

Rationale: The system is designed to keep track of Prescriptions between a Physician and a Patient.

REQ6	The system will allow Patients to have multiple Prescriptions, each with a unique Physician.	See Table 3 for list of abstract states.

Rationale: The client has specified that Patients are not limited to one Physician, so they may have multiple Prescriptions. They will, however, have only one prescription per Physician.

REQ7	The system *will not* allow a Generalist to add a Medication to a Prescription if the Medication leads to a dangerous Interaction. The system *will* allow a Specialist to add a Medication to a Prescription if the Medication leads to a dangerous Interaction	See Table 1 for adding medicine, and Table 3 for dangerous interactions.

Rationale: The system will prevent Generalists from adding Medications to Prescriptions where they may cause harm to the Patient. Specialists are have more training, and therefore are determined to know what they are doing when prescribing such Medications.

REQ8	An interaction between two Medications cannot be added if both medications are prescribed to a patient by generalists.	See Table 1 for adding interaction, and Table 3.

Rationale: This critical safety requirement prevents a Generalist from being able to add dangerously interacting Medications to Prescriptions before the Interaction is added to the system. This requirement will prompt the user of the system to first remove one of the interacting Medications from the Prescription.

REQ9	A Medication may only be prescribed to a Patient once.	A Medication may only be prescribed within the safe dosage range

Rationale: This prevents different Physicians prescribing the same Medications and exceeding the safe dosage.

7.2. Environmental Descriptions

ENV10	All input to the system will be constrained to the input grammar.	See Table 1 for the list possible monitored events.

ENV11	Physicians and Patients may have identical names.	We cannot control the names of people. They may have the same name.

ENV12	Physicians are either *Generalists* or *Specialists*.	See Table 2 for PHYSICIAN_TYPE.

ENV13	The DOSE type will be constrained by the GUI grammar to be either mg, or cc.	See Table 2 for DOSE.

ENV14	The KIND type will be constrained by the GUI grammar to be either a pill or a liquid.	See Table 2 for KIND.

8. Function Tables

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

Figure 3: Table of errors and warnings

8.1. Function Table for eHealth

Monitored Inputs		
i = 0		See Table 5
i >0	new_prescription(id, doctor, patient)	See Table 6
	add_medicine(id, medicine, dose)	See Table 7
	remove_medicine(id, medicine)	See Table 8
	add_interaction(id1, id2)	See Table 9

Figure 4: Function Table for eHealth

8.2. 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 5: Table 5. Function Table for init

8.3. 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)	$rx_{-1} \upharpoonleft (id \mapsto (md, pt))$	NC
prs(i)	$prs_{-1} \upharpoonleft (id \mapsto empty\_prs(mnid_1))$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	$dpr_{-1} \upharpoonleft (id \mapsto \varnothing)$	NC
r(i)	ok	error

Figure 6: Table 6. Function Table for new_prescription(id, doctor, patient)

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

error: $\neg($
$id > 0$
$\wedge\, rxid_{-1}(id)$
$\wedge\, m > 0$
$\wedge\, mnid_1(m)$
$\wedge\, \neg(prs_{-1}(id)(m)‘1 > 0)$
$\wedge\, (\exists(b : ID\_MD, c : ID\_PT) :$
$\quad((prs_{-1}(id)(m)‘1 > 0$
$\quad\; \wedge\, (rx_{-1}(id) = (b, c)$
$\quad\; \wedge\, gs_{-1}(b) = sp))))$
$\wedge\, isValidDose(m, d))$

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)	$prs_{-1}(id) \upharpoonleft (m \mapsto d))$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	NC	NC
r(i)	ok	error

Figure 7: Table 7. Function Table for add_medicine(id, medicine, dose)

8.5. Function Table for remove_medicine(id, med)

error: $\neg($
$id > 0$
$\wedge\, rxid_{-1}(id)$
$\wedge\, med > 0$
$\wedge\, mnid_{-1}(med)$
$\wedge\, prs_{-1}(id)(med)`1 > 0)$

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)	$prs_{-1} \upharpoonleft (id \mapsto empty\_prs(mnid_1))$	NC
di(i)	NC	NC
gs(i)	NC	NC
dpr(i)	NC	NC
r(i)	ok	error

Figure 8: Table 8. Function Table for remove_medicine(id, medicine)

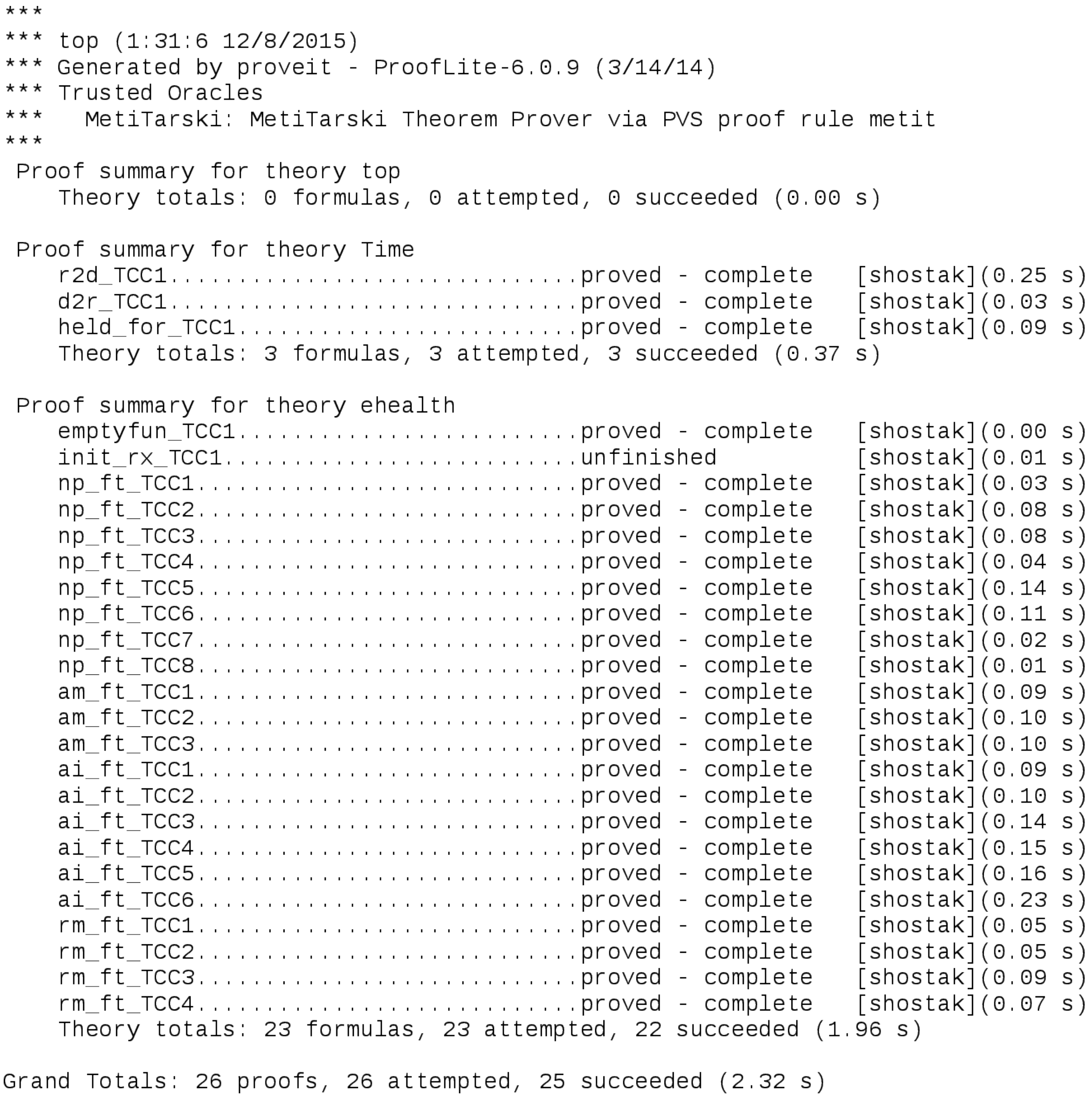
8.6. Function Table for add_interaction(id1, id2)

error: $\neg($
$id1 > 0$
$\wedge\, id2 > 0$
$\wedge\, \neg id1 = id2$
$\wedge\, member(id1, mnid_{-1})$
$\wedge\, member(id2, mnid_{-1})$
$\wedge\, \neg member((id1, id2), di_{-1})$
$\wedge\, (\exists (a : ID\_RX, b : ID\_MD, c : ID\_PT) :$
$\quad member(a, rxid_{-1})$
$\quad \wedge\, member(b, mdid_{-1})$
$\quad \wedge\, member(c, ptid_{-1})$
$\quad \wedge\, ((prs_{-1}(a)(id1)‘1 > 0$
$\qquad \wedge\, (rx_{-1}(a) = (b, c)$
$\qquad \wedge\, gs_{-1}(b) = sp))$
$\quad \vee\, (prs_{-1}(a)(id2)‘1 > 0$
$\qquad \wedge\, (rx_{-1}(a) = (b, c)$
$\qquad \wedge\, gs_{-1}(b) = sp)))))$
$\wedge\, \neg(member((id1, id2), di_{-1})$
$\vee\, (member((id1, id2), di_{-1})$

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 9: Table 9. Function Table for add_interaction(id1, id2)

9. Validation



```
***
*** top (1:31:6 12/8/2015)
*** 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](0.25 s)
    d2r_TCC1..............................proved - complete   [shostak](0.03 s)
    held_for_TCC1.........................proved - complete   [shostak](0.09 s)
    Theory totals: 3 formulas, 3 attempted, 3 succeeded (0.37 s)

 Proof summary for theory ehealth
    emptyfun_TCC1.........................proved - complete   [shostak](0.00 s)
    init_rx_TCC1..........................unfinished          [shostak](0.01 s)
    np_ft_TCC1............................proved - complete   [shostak](0.03 s)
    np_ft_TCC2............................proved - complete   [shostak](0.08 s)
    np_ft_TCC3............................proved - complete   [shostak](0.08 s)
    np_ft_TCC4............................proved - complete   [shostak](0.04 s)
    np_ft_TCC5............................proved - complete   [shostak](0.14 s)
    np_ft_TCC6............................proved - complete   [shostak](0.11 s)
    np_ft_TCC7............................proved - complete   [shostak](0.02 s)
    np_ft_TCC8............................proved - complete   [shostak](0.01 s)
    am_ft_TCC1............................proved - complete   [shostak](0.09 s)
    am_ft_TCC2............................proved - complete   [shostak](0.10 s)
    am_ft_TCC3............................proved - complete   [shostak](0.10 s)
    ai_ft_TCC1............................proved - complete   [shostak](0.09 s)
    ai_ft_TCC2............................proved - complete   [shostak](0.10 s)
    ai_ft_TCC3............................proved - complete   [shostak](0.14 s)
    ai_ft_TCC4............................proved - complete   [shostak](0.15 s)
    ai_ft_TCC5............................proved - complete   [shostak](0.16 s)
    ai_ft_TCC6............................proved - complete   [shostak](0.23 s)
    rm_ft_TCC1............................proved - complete   [shostak](0.05 s)
    rm_ft_TCC2............................proved - complete   [shostak](0.05 s)
    rm_ft_TCC3............................proved - complete   [shostak](0.09 s)
    rm_ft_TCC4............................proved - complete   [shostak](0.07 s)
    Theory totals: 23 formulas, 23 attempted, 22 succeeded (1.96 s)

Grand Totals: 26 proofs, 26 attempted, 25 succeeded (2.32 s)
```

Figure 10: Validated Isolette

10. Acceptance Tests

```
  at1.txt
add_physician       (1, "Mayo", specialist)
add_patient         (3, "Dora")
add_patient         (1, "Drew")
add_medication      (1, ["Wafarin", pill, 1.0, 6.0])
add_medication      (3, ["caffeine", liquid, 1.0, 16.0])
add_medication      (2, ["acetaminophen", liquid, 1.0, 25.5])
add_interaction     (2,3)
add_interaction     (1,2)
new_prescription    (2, 1, 3)
new_prescription    (1, 1, 1)
dpr_q
add_medicine        (1, 1, 5.5)
add_medicine        (1, 2, 5.5)
add_medicine        (1, 3, 5.5)
add_medicine        (2, 2, 5.5)
add_medicine        (2, 3, 5.5)
add_medicine        (2, 1, 5.5)
prescriptions_q(1)
```

Figure 11: First Acceptance Test

```
-- at2.txt
add_physician       (1, "Mayo", specialist)
add_patient         (3, "Drew")
add_patient         (1, "Helen")
add_medication      (1, ["Wafarin", pill, 1.0, 6.0])
add_medication      (3, ["caffeine", liquid, 1.0, 16.0])
add_medication      (2, ["acetaminophen", liquid, 1.0, 25.5])
add_interaction     (1,2)
add_interaction     (1,3)
add_interaction     (2,3)
new_prescription    (2, 1, 3)
new_prescription    (1, 1, 1)
add_medicine        (1, 1, 5.5)
add_medicine        (1, 2, 5.5)
add_medicine        (1, 3, 5.5)
add_medicine        (2, 2, 5.5)
add_medicine       (2, 3, 5.5)
add_medicine       (2, 1, 5.5)
dpr_q
remove_medicine(2,1)
remove_medicine(2,2)
remove_medicine(2,3)
add_medicine       (1, 1, 5.5)
add_medicine       (1, 2, 5.5)
add_medicine       (1, 3, 5.5)
dpr_q
```

```
 -- at3.txt\
add_physician      (1, "Mayo", specialist)
add_physician      (2, "Mayo", specialist)
add_physician      (3, "Mayo", generalist)

add_patient        (3, "Drew")
add_patient        (1, "Drew")

add_medication     (1, ["Wafarin", pill, 1.0, 6.0])
add_medication     (2, ["acetaminophen", liquid, 1.0, 25.5])

new_prescription   (2, 1, 3)
new_prescription   (4, 3, 1)
new_prescription   (1, 1, 1)

new_prescription   (3, 2, 3)

add_medicine       (1, 1, 5.5)
add_medicine       (1, 2, 5.5)

add_medicine       (4, 2, 5.5)

add_medicine       (2, 2, 5.5)
add_medicine       (2, 1, 5.5)

add_medicine       (3, 1, 5.5)
add_medicine       (3, 2, 5.5)

-- error cases
-- add_physician
```

```
add_physician      (-1, "Mayo", specialist)
add_physician      (1, "Mayo", specialist)
add_physician      (6, "99Yo", specialist)

-- add_patient
add_patient        (-1, "Drew")
add_patient        (1, "Drew")
add_patient        (7, "76rew")

-- add_medication
add_medication     (-1, ["Wafarin", pill, 1.0, 6.0])
add_medication     (1, ["Wafarin", pill, 1.0, 6.0])
add_medication     (6, ["23afarin", pill, 1.0, 6.0])
add_medication     (7, ["Wafarin", pill, 1.0, 6.0])
add_medication     (8, ["Wafarin", pill, 6.0, 1.0])
```

```
dpr_q
-- add_interaction
add_interaction    (1,-2)
add_interaction    (1,1)
add_interaction    (7,9)
add_interaction    (1,2)
add_interaction    (1,2)
add_interaction    (2,1)
dpr_q

-- new_prescription
new_prescription   (-3, 2, 3)
new_prescription   (4, 2, 3)
new_prescription   (5, -2, 3)
new_prescription   (6, 2, 3)
new_prescription   (7, 2,-3)
new_prescription   (8, 2, 3)
new_prescription   (3, 20, 3)
new_prescription   (3, 2, 30)

-- add_medicine
add_medicine       (-2, 1, 5.5)
add_medicine       (2, -1, 5.5)
add_medicine       (2, 100, 5.5)
add_medicine       (2, 1, 5.5)
add_medicine       (2, 1, 5.5)
add_medicine       (2, 100, 55.00)

prescriptions_q(-1000)
prescriptions_q(1000)
```

A. eHealth PVS

```
% This is a partial theory to help you get started encoding your
% function tables in PVS for the eHealth project.
% This theory type checks but the function tables are
% not valid as the requirements have not been properly elicited.
% Furthermore the function tables do not respect our format
% as completeness and disjointness is circumvented by the
% ELSE keyword.You may not use the ELSE keyword in function tables
% for this project.
% You are not required to prove any invariants.
% Nevertheless, we show you below how to prove some simple
% invariants as part of the state as TCCs. See fields inv1 and
% inv2 in the STATE record using the unit ADT. You may omit these
% invariants in the state if you choose, but they do help to ensure
% the correct requirements if kept.
% Note that we show a change of state using the override WITH
% operator so that any part of the state not overriden is left
% unchanged.

ehealth: THEORY
BEGIN
  delta: posreal % sampling time
  IMPORTING Time[delta]
  IMPORTING structures@Unit_adt
  i: VAR DTIME

  % Definition of an empty function
  emptyfun [T, U : TYPE] (x : {x : T | FALSE}) : RECURSIVE U =
    emptyfun(x)
    MEASURE (LAMBDA (x : {x : T | FALSE}): 1)

  ID_MD: TYPE+ = int %physicians
  ID_PT: TYPE+ = int %patients
  ID_RX: TYPE+ = int %prescriptions
  ID_MN: TYPE+ = int %medications

  % Physician type
  GS: TYPE+ = {gn, sp}
  UNIT: TYPE+ = {cc, mg}
  DOSE: TYPE = [nnreal, UNIT]
```

```
NAME: TYPE+
KIND: TYPE+ = {pill, liquid}

MEDICINE: TYPE
= [name:NAME, kind:KIND, low:nnreal, hi:nnreal]

COMMAND : DATATYPE
  BEGIN
    m_np(id:ID_RX, md: ID_MD, pt: ID_PT): np?
    m_ai(id1:ID_MN, id2:ID_MN): ai?
    m_am(id:ID_RX, med:ID_MN, dose:DOSE): am?
    m_rm(id:ID_RX, med: ID_MN): rm?
  END COMMAND
cmd: VAR [POS_DTIME -> COMMAND]

invariant (p : bool) : TYPE = { x : Unit | p }
          % unit : { x : Unit | 2 is even }
          %  (type correct IFF: 2 is even [ x := unit ]
          %                 ...  2 is even
          %                      TRUE
          %
          % unit : { x : Unit | 3 is even }
          %  (type correct IFF: 3 is even [x := unit]
          %                 ...  3 is even
          %                      FALSE
          %
          % { x : Unit | p } = IF p THEN {unit} ELSE {} ENDIF

has [T : TYPE] (m : T, p : [T -> DOSE]) : bool = p(m)`1 > 0
    % does prescription p have a non-zero dose of m?

% Have to place the state in a record
STATE: TYPE =
  [#
      mnid: set[ID_MN]  % medication ids
    , ptid: set[ID_PT]  % patient ids
    , mdid: set[ID_MD]  % doctor ids
    , rxid: set[ID_RX]  % prescription ids
    , mdpt: set[[(mdid),(ptid)]]
    % (doctor, patient) care relation
    , rx:   [(rxid) -> (mdpt)]
    % care to rx ids,  needs to be a bijection
    , prs: [(rxid) -> [(mnid) -> DOSE]]
    % prescriptions
    , di: set[[(mnid), (mnid)]]
    , gs: [(mdid) -> GS] % kind of doctor
    , dpr : [(rxid) -> set[[(mnid), (mnid)]]]
 #]

PRES (s : STATE) : TYPE =  [(s`mnid) -> DOSE]
     % type of PRESCRIPTIONS for a given state

% would prescriptions p0 and p1 cause dangerous interactions
      % if they were prescribed to the same patient?
interact (s : STATE)(p0, p1 : PRES (s)) : bool =
         EXISTS (m0,m1 : (s`mnid)):
                s`di((m0,m1))
            AND has(m0,p0)
            AND has(m1,p1)

        % given state s, does medication m1 cause a problem
        % for the patient of prescription p0?
interactWith (s: STATE)(p0 : PRES (s), m1 : (s`mnid)) : bool =
         EXISTS (m0 : (s`mnid)): s`di((m0,m1)) AND has(m0,p0)

         medicine: MEDICINE
isValidDose(s : STATE)(m : ID_MN, d : DOSE) :
bool = d`1 > 0 AND d`1 > medicine`3 AND medicine`4 > d`1
         % is d a valid dose of medication m?
         % kept abstract; will need a counterpart in the state
         % in order to be refined

prsOfPt (s: STATE)(p: (s`ptid)) : set [(s`rxid)] =
        { r : (s`rxid) | s`rx(r)`2 = p }

ptOf (s: STATE)(r : (s`rxid)) : (s`ptid) = s`rx(r)`2

mdOf (s: STATE)(r : (s`rxid)) : (s`mdid) = s`rx(r)`1

s: VAR [ DTIME -> STATE ]

empty_prs (mdns : set[ID_MN])(m : (mdns)) : DOSE = (0, mg)
```

```
init_mdid: set[ID_MD]
init_gs: [(init_mdid) -> GS]
init_mnid: set[ID_MN]
init_ptid: set[ID_PT]
init_rxid: set[ID_RX]
init_mdpt: set[[(init_mdid),(init_ptid)]]
init_dpr: [(init_rxid) -> set[[(init_mnid), (init_mnid)]]]
init_prs: [(init_rxid) -> [(init_mnid) -> DOSE]]
init_rx: [(init_rxid)->(init_mdpt)]

init_state : STATE =
     (# mnid := init_mnid
      , ptid := init_ptid
      , mdid := init_mdid
      , rxid := init_rxid
      , mdpt := init_mdpt
      , rx := init_rx
      , prs := init_prs
      , di := emptyset
      , gs := init_gs
      , dpr := init_dpr
      #)

      % new_prescription (id: ID_RX; doctor: ID_MD;
      %               patient: ID_PT)
      %     prescription id must be a positive integer
      %     prescription id already in use
      %     physician id must be a positive integer
      %     physician with this id not registered
      %     patient id must be a positive integer
      %     patient with this id not registered
      %     prescription already exists for this physican
      %               and patient
np_ft(id:ID_RX, md: ID_MD, pt: ID_PT)(s)(i): bool =
  COND
      i = 0 -> s(i) = init_state,
      i > 0 ->
      COND
      id > 0
      AND NOT rxid_ (id)
```

```
        AND md > 0
        AND mdid_ (md)
        AND pt > 0
        AND ptid_ (pt)
        AND mdpt_ (md, pt) ->
               s(i) = s(i-1)
               WITH  [ rxid := add(id,rxid_)
                     , rx  := rx_ WITH [id := (md,pt)]
                     , prs := prs_ WITH [id := empty_prs(mnid_) ]
                     , dpr := dpr_ WITH [id := emptyset ]
                    ] ,
        NOT (id > 0
        AND NOT rxid_ (id)
        AND md > 0
        AND mdid_ (md)
        AND pt > 0
        AND ptid_ (pt)
        AND mdpt_ (md, pt))
          -> s(i) = s(i-1)
    ENDCOND
    where
       rxid_ = s(i-1)`rxid
      ,mdid_ = s(i-1)`mdid
      ,ptid_ = s(i-1)`ptid
      ,mdpt_ = s(i-1)`mdpt
      ,rx_   = s(i-1)`rx
      ,dpr_  = s(i-1)`dpr
      ,mnid_ = s(i-1)`mnid
      ,prs_ = s(i-1)`prs
      ENDCOND

%dose: DOSE
 % Add_medicine (id: ID_RX; medicine:ID_MN; dose: VALUE)
 %     prescription id must be a positive integer
 %     prescription with this id does not exist
 %     medication id must be a positive integer
 %     medication id must be registered
 %     medication is already prescribed
 %     specialist is required to add a dangerous interaction
 %     dose is outside allowed range
  am_ft(id:ID_RX, m: ID_MN, d: DOSE)(s)(i): bool =
    COND
        i = 0 -> s(i) = init_state,
        i > 0 -> COND
        id > 0
        AND rxid_ (id)
        AND m > 0
        AND mnid_ (m)
        AND NOT (prs_ (id)(m)`1 > 0)
        AND NOT (EXISTS (b: ID_MD, c: ID_PT):
        ((prs_ (id)(m)`1 > 0 AND (rx_ (id) = (b,c)
        AND gs_ (b) = sp ))))
        AND isValidDose_ (m,d)
            -> s(i) = s(i-1) WITH [ prs := prs_
            WITH [id := prs_(id) WITH [m := d] ] ]
     , NOT(     id > 0
        AND rxid_ (id)
        AND m > 0
        AND mnid_ (m)
        AND NOT (prs_ (id)(m)`1 > 0)
        AND NOT (EXISTS (b: ID_MD, c: ID_PT):
        ((prs_ (id)(m)`1 > 0 AND (rx_ (id) = (b,c)
        AND gs_ (b) = sp ))))
        AND isValidDose_ (m,d))
        -> s(i) = s(i-1)
    ENDCOND
    where
       rxid_ = s(i-1)`rxid
      ,mnid_ = s(i-1)`mnid
      ,di_ = s(i-1)`di
      ,rx_ = s(i-1)`rx
      ,gs_ = s(i-1)`gs
      ,prs_  = s(i-1)`prs
      ,mdOf_ = mdOf(s(i-1))
      ,ptOf_ = ptOf(s(i-1))
      ,interactWith_ = interactWith(s(i-1))
      ,prsOfPt_ = prsOfPt(s(i-1))
      ,isValidDose_ = isValidDose(s(i-1))
      ENDCOND

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FUN FUN FUNCTION TABLES %%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %add_interaction (id1:ID_MN;id2:ID_MN)
      %    medication ids must be positive integers
      %    medication ids must be different
      %    medications with these ids must be registered
      %    interaction already exists
      %    first remove conflicting medicine prescribed by
      % generalist
      ai_ft(id1:ID_MN, id2:ID_MN)(s)(i): bool = COND
          i = 0 -> s(i) = init_state,
          i > 0 -> COND
          id1 > 0 AND id2 > 0
          AND NOT (id1 = id2)
          AND member(id1, mnid_ )
          AND member(id2, mnid_ )
          AND NOT member((id1,id2), di_ )
          AND NOT (EXISTS (a: ID_RX, b: ID_MD, c: ID_PT):
          member(a, rxid_ ) AND member(b, mdid_ )
          AND member(c, ptid_ ) AND ((prs_ (a)(id1)`1 > 0
          AND (rx_ (a) = (b,c) AND gs_ (b) = sp ))
              OR (prs_ (a)(id2)`1 > 0 AND (rx_ (a) = (b,c)
              AND gs_ (b) = sp))))
         AND NOT (member((id1, id2), di_ )
            OR member((id2, id1), di_ ))
          -> s(i) = s(i-1) WITH [
              di := add((id1, id2), di_ )
              ,di := add((id2, id1), di_ )
             % ,dpr := dpr_ WITH []
          ],
          NOT (
            id1 > 0 AND id2 > 0
          AND NOT (id1 = id2)
          AND member(id1, mnid_ )
          AND member(id2, mnid_ )
          AND NOT member((id1,id2), di_ )
          AND NOT (EXISTS (a: ID_RX, b: ID_MD, c: ID_PT):
           member(a, rxid_ ) AND member(b, mdid_ )
           AND member(c, ptid_ ) AND ((prs_ (a)(id1)`1 > 0
           AND (rx_ (a) = (b,c) AND gs_ (b) = sp ))
              OR (prs_ (a)(id2)`1 > 0 AND (rx_ (a) = (b,c)
              AND gs_ (b) = sp))))
         AND NOT (member((id1, id2), di_ )
            OR member((id2, id1), di_ )) ) -> s(i) = s(i-1)
      ENDCOND
      where
        di_   = s(i-1)`di
        ,dpr_  = s(i-1)`dpr
        ,rxid_ = s(i-1)`rxid
        ,mnid_ = s(i-1)`mnid
        ,mdid_ = s(i-1)`mdid
        ,ptid_ = s(i-1)`ptid
        ,prs_  = s(i-1)`prs
        ,rx_ = s(i-1)`rx
        ,mdpt_ = s(i-1)`mdpt
        ,gs_ = s(i-1)`gs
        ENDCOND
      %remove_medicine  (id: ID_RX; medicine:ID_MN)
      %    prescription id must be a positive integer
      %    prescription with this id does not exist
      %    medication id must be a positive integer
      %    medication id must be registered
      %    medication is not in the prescription
      rm_ft(id:ID_RX, med: ID_MN)(s)(i): bool = COND
          i = 0 -> s(i) = init_state,
        i > 0 -> COND
           id > 0
          AND rxid_ (id)
          AND med > 0
          AND mnid_ (med)
          AND prs_ (id)(med)`1 > 0
          -> s(i) = s(i-1) WITH [
            prs := prs_ WITH [id := empty_prs(mnid_ ) ]
          ],
          NOT (
            id > 0
            AND rxid_ (id)
            AND med > 0
            AND mnid_ (med)
            AND prs_ (id)(med)`1 > 0
          ) -> s(i) = s(i-1)
      ENDCOND
```

```
      where
         rxid_ = s(i-1)`rxid
        ,dpr_  = s(i-1)`dpr
        ,mnid_ = s(i-1)`mnid
        ,prs_  = s(i-1)`prs
        ENDCOND

ehealth_ft(cmd)(s)(i): bool = COND
  i = 0 -> s(0) = init_state,
  i > 0 ->
    CASES cmd(i) OF
      m_np(id, md, pt):  np_ft(id, md, pt)(s)(i),
      m_ai(id1, id2): ai_ft(id1, id2)(s)(i),
      m_am(id, med, dose): am_ft(id, med, dose)(s)(i),
      m_rm(id, med): rm_ft(id, med)(s)(i)
    ENDCASES
ENDCOND

END ehealth
```