CIS 301 – Logical Foundations of Programming
Homework #2 – 2018

Working with System Requirements in Propositional Logic

Due Monday February 12 at 11:59pm

Purpose of Assignment:

- To help you understand how to translate natural language requirements into propositional logic
- To help you understand how to use a SAT/SMT solver to automatically reason about propositional logic, and in a very simple fashion illustrate a few of the benefits of SAT/SMT solvers

Total Points: 30 points + 4 bonus points (the bonus comes from the fact that the total number of points is 34, but the assignment will be graded out of a total of 30 points).

Background:

Figure 1: Typical PCA pump with bolus trigger

A Patient-Controlled Analgesia (PCA) pump is a medical device often used in clinical settings to intravenously infuse pain killers (e.g., opioids) at a programmed rate into a patient's blood stream through an IV line. KSU CS researchers in the SAnToS Laboratory and their partners at Adventium Labs are funded by the Department of Homeland Security to develop an open source safe/secure implementation of a PCA pump. For more background information on PCA Pumps (not required for this assignment), see our project website: http://openpcapump.santoslab.org/node/1

In this assignment, you will translate some example requirements of a PCA pump into propositional logic.

Information for This Homework

There are three basic ways that a PCA pump can be commanded to infuse opioids to a patient:

1. A clinician may program the pump to deliver a *basal infusion* – a relatively low-level amount of opioid that is continuously infused into the patient
2. If the clinician senses that the patient is in pain, the clinician may command the pump to give a burst of opioid (called a *clinician* bolus or *square* bolus dose)
3. If the patient feels like they are in pain, they can press a button attached to the pump to receive a *patient* bolus dose. This allows patients to have some control over their own pain relief.

The pump motor mechanism operates at different rates to infuse opioids as appropriate for the current mode. For example, the pump may infuse at a slower rate for the basal infusion and then change to a higher rate when a clinician bolus or patient bolus dose is requested.

Over-infusion of opioids can lead to respiratory depression and brain damage or death. Therefore, PCA pumps have a variety of safe guards built into them. Here are some examples that are relevant for this homework:

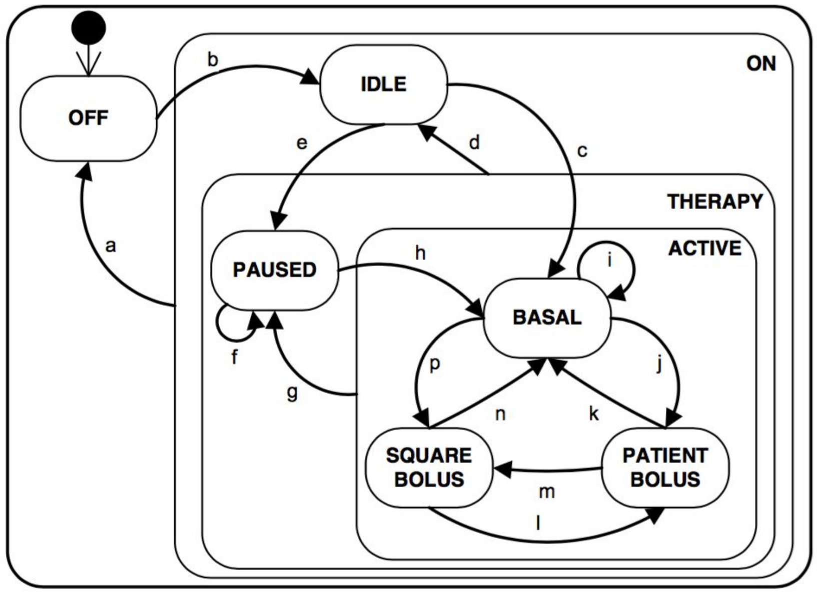
- When the patient pushes the patient bolus button and patient bolus dose is delivered, pushing the button again will have no effect until a *lockout interval* (e.g., 10 minutes) has passed.
- There is a *max volume of drug* per hour that can be infused. This max amount is configured by the clinician during the patient/pump set up. Once that limit is reached, no more drug will be infused until the hour has passed.

The pump also has different types of sensors to automatically detect various types of problems with the pump that may affect its safety or effective operation.

- The PCA pump device uses a pressure sensor in the fluid lines to detect when there is an "upstream occlusion", i.e., a blockage between the drug reservoir and pump mechanism that prevents opioid from flowing through the pump. Similar sensors are used to detect a "downstream occlusion", i.e., a blockage between the pump mechanism and the patient (e.g., due to a kinked IV line).
- The PCA pump device uses a sensor to detect air bubbles ("air embolisms") in the fluid line. Infusing air bubbles into a patient's circulatory system can be harmful.

Below is a diagram of PCA pump operational modes developed by researchers at the University of Minnesota. Each rounded-corner rectangle represents a mode. Some modes contain sub-modes. For example, **Idle** and **Therapy** are submodes of the **On** mode; **Paused** and **Active** are submodes of the **Therapy** mode; **Basal**, **Square Bolus** and **Patient Bolus** are submodes of the **Active** mode.

In the full requirements document, every sub mode is subject to all the requirements defined for its parent mode in addition to the specific requirements defined for the new mode. An important class of requirements are those that define/constrain transitions between modes.



Questions:

The questions for this assignment are divided into two parts. In the first, you will be formalize natural language. In the second, you will use Z3, an SMT solver.

What to submit:
Answers should be entered into the provided files. These files should be committed to your student repository and pushed by the deadline. Responses with the wrong path, including directory, will not be graded.

Part 1:

Following the approach outlined in the 301 lecture *English to Propositional Logic*, formalize each of the requirements listed below in propositional logic. For each requirement, indicate your propositional variables, the intuitive meaning of the propositional variables, and then give the propositional formula that formalizes the requirement.

Example Requirement:

The PCA Pump shall detect downstream occlusions and upstream occlusions.

Solution:

- Propositional Variables
  - P = "The PCA Pump shall detect downstream occlusions"
  - Q = "The PCA Pump shall detect upstream occlusions"
- Logic formalization of Requirement: P ^ Q

General Hints:

Propositional logic doesn't enable us to talk about sequences in time, so capturing the exact meaning of something like "transitions to IDLE mode" is difficult. In such a case, we might make a convention to talk about the "current mode" and the "next mode". That still doesn't tell us about what we would allow as the maximum delay in getting to the next mode, but we will ignore that issue in this assignment.

- When talking about a current mode, e.g., "the device is IDLE mode", define a propositional variable (e.g., P) to be P = "current mode is IDLE"
- When talking about transition to a new mode, e.g., "transitioning to PATIENT bolus mode", define a propositional variable (e.g., P) to be P = "next mode is PATIENT".

(1) (3 points) Requirement (regarding a safety feature that guards against the infusion of air bubbles).

If an air bubble is detected, the device shall raise an alarm on the operator interface and transition to IDLE mode.

(2a) (2 points) Requirement:

The device is in PATIENT BOLUS sub mode only when it is in ACTIVE mode

(2b) (1 point) Based on the presentation of the mode diagram above, why would it be incorrect for the requirement to be phrased as "The device is in PATIENT BOLUS sub mode if and only if it is in ACTIVE mode."?

(3) (3 points) Requirement:

The device shall infuse at the basal rate F_{basal} when the device is in BASAL submode and only for that sub mode.

(4) (3 points) Requirement:

If the device is in Active Mode, it is either in the BASAL sub mode, SQUARE BOLUS sub mode or PATIENT BOLUS sub mode.

(5) (3 points) Requirement:

> *When the patient bolus button is pressed and the device is in Active Mode and the lockout interval is not in effect, the device shall transition to PATIENT BOLUS sub mode unless it is in SQUARE BOLUS sub mode.*

(6) (4 points total) Getting requirements into a canonical form – we saw in our lecture that there are many different English language works that can be mapped to each operator. For example, "but", "however", "moreover", etc. all mapped to "and". It almost always improves understanding of the requirements if we always write them consistently in terms of the same words (for example, here is a link to a consistent approach championed by an engineer from Intel called "EARS: The Easy Approach to Requirements Syntax".

https://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf

For this problem, we will NOT use the Intel approach, but will adopt our own simple strategy:

- Replace all words that correspond to the logical operator ^ with "and"
- Replace all words that correspond to the logical operator v with "or"
- Replace all words that correspond to the logical operator -> with "if … then…" with the "if" portion of the sentence occurring before the "then" portion of the sentence.
- Replace all words that correspond to the logical operator <-> with "if and only if"
- Replace all phrases "P1 or P2 or … or Pn" where the "or" corresponds to an exclusive or with "P1, P2, … or Pn, exclusively"

For 6a, 6b, 6c, and 6d, use this approach to rewrite requirements 1, 2, 3, and 4 respectively (making use of the logical representations you already created).

Part 2:

Problems in this part require solutions in Z3 and Logika. You have two ways to run Z3 scripts. You can either use https://rise4fun.com/z3 or the command line utility built into your Sireum installation. The program is located at `<SIREUM_HOME>/apps/z3/bin/z3`. Note that on Windows, the slashes are reversed. On departments machines, you can simply use `z3`. To use the program, type `<program> -smt2 <input>`. Where `<program>` is the path to the program and `<input>` is the name of the Z3 file you want to run.

(7) (5 points total) Automatic reasoning tools like Z3 solvers allow us to debug our specifications by generating test cases. We will see that in a much more impressive

much later in the course, but for now we can use Z3 in a direct/manual way following the four steps below explained in the Automated Reasoning for Propositional Logic lecture:

1. declare all symbols involved in the formula
2. translate the formula and then assert it
3. issue the check-sat command
4. get model(s) (if applicable)

Consider the following requirement:

> *When the device is in OFF mode, it is neither in PAUSED mode nor is the pump infusing a drug.*

Consider a scenario where a company making the PCA Pump recently hired two employees Jill and Josie who were asked to review the requirements. To make sure they understood the requirements, Jill and Josie tried to study how the requirements used English words like “unless”, etc. that didn’t have an extremely clear logical meaning.

Both Jill and Josie agreed that the requirement above had three main atomic propositions:

P = Current mode is OFF mode
Q = Current mode is PAUSED
R = Pump is infusing a drug

Jill argued that a direct rendering of the formula in propositional logic would be:

P -> ~(Q V R)

Josie argued that formula was equivalent to

P -> (~Q ^ ~R)

…and that it would be better to reword the English requirements to always use “If” instead of “when” and to stick with using “and”, “or”, and “not” that always have a clear interpretation to Boolean logic instead of using words like “neither” and “nor” which don’t have a distinct symbol for them in Boolean logic.
Josie wanted to write…

> *If the device is in OFF mode, then the device is not in PAUSED mode and it is not infusing a drug.*

Since Jill and Josie are new on the job, they want to make sure that the different ways of writing the requirement are equivalent.

Show how you would help Jill and Josie prove the two formula above are equivalent using Z3, i.e., show that (P -> ~(Q V R)) <-> (P -> (~Q ^ ~R)) is a tautology (remember that "=" can be used for "<->" in Z3). See slides 60-61 and the associated FYTD exercises on slide 62 in the Automated Reasoning for Propositional Logic lecture. The solutions for the FYTD Exercises on slide 62 can be found here: https://drive.google.com/open?id=0B2v9RmdjXXbUb0tkTXJaSzJaNE0

(a) (3 points) Write a Z3 file that proves the equivalence of the two formulas above. Your file must run to receive full points.

(b) (2 points) In a comment in your Z3 file, explain why the output of your file really does prove the equivalence of the two formulas above. In Z3, end-of-line comments start with a `;`

(8) (10 points total) SAT and SMT solvers are sometimes used to help generate test cases.

Consider the following requirement:

R1: When the total volume of drug infused in the current hour reaches the *max volume of drug* per hour limit, the device transitions to PAUSED mode and a notice that the max volume of drug per hour limit has been reached is displayed on the operator interface.

The requirement R1 can be expressed as a propositional logic formula that has the following three atomic propositions:

> *P* = " the total volume of drug infused in the current hour reaches the *max volume of drug* per hour limit"
> *Q = "next mode = PAUSED"*
> *R = "max volume per hour limit has been reached is displayed on operator interface"*

Since the formula has three propositional variables, there will be 8 rows in its truth table (e.g., 8 test cases that will cover all possibilities of the system state relevant to the requirement above).

Following the "recipe" given in class, use Z3 to generate the rows in the truth table (i.e., "tests") for the requirement R1.

See slides See slides 44-58 and the associated FYTD exercises on slide 59 in the Automated Reasoning for Propositional Logic lecture. The solutions for the FYTD

Exercises on slide 59 can be found here:
https://drive.google.com/file/d/0B2v9RmdjXXbUblZBTDZIQm1YcGM/view
Complete these steps:
(a) Express R1 as a propositional logic formula.
(b) Start a logika truth table using the expression.
(c) Write a Z3 file to find and output a satisfying case for the expression.
(d) Add this case to your truth table and complete the row.
(e) Follow the "recipe" illustrated in class to assert the negation of the last case and find and output another satisfying case.
(f) Repeat (d) and (e) until there are no more satisfying cases.
(g) complete your truth table
(h) Give a one sentence English summary of the intuition of each of the rows in the truth table.

To receive full points:

(1) The truth table must have the correct expression
(2) The truth table must verify
(3) The Z3 file must have the correct initial assertion
(4) Each additional assertion must match an already outputted case
(5) The Z3 file must run without errors