COM4506: Testing and verification in safety-critical systems

by

Shubham K Yadav

Assignment: Requirements and Specification for an Artificial Pancreas

Department of Computer Science

1 Graphical EFSM

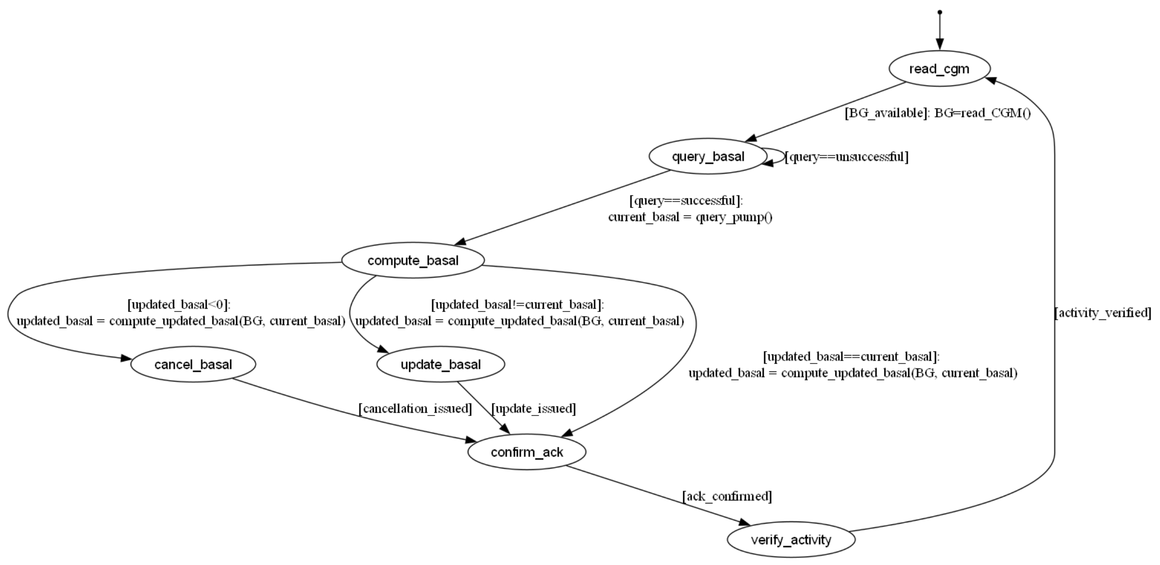
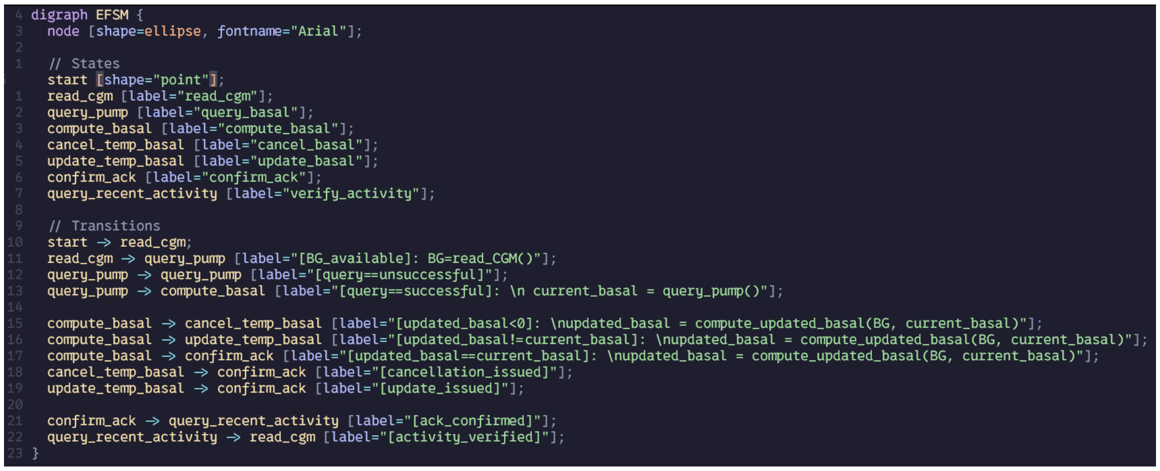

read_cgm
[BG_available]: BG=read_CGM()
query_basal
[query==unsuccessful]
[query==successful]:
current_basal = query_pump()
compute_basal
[updated_basal<0]:
updated_basal = compute_updated_basal(BG, current_basal)
[updated_basal!=current_basal]:
updated_basal = compute_updated_basal(BG, current_basal)
[updated_basal==current_basal]:
updated_basal = compute_updated_basal(BG, current_basal)
[activity_verified]
cancel_basal
update_basal
[cancellation_issued]
[update_issued]
confirm_ack
[ack_confirmed]
verify_activity

Figure 1: Graphical version of the EFSM

2 Dot-encoded version of the EFSM



```
digraph EFSM {
  node [shape=ellipse, fontname="Arial"];

  // States
  start [shape="point"];
  read_cgm [label="read_cgm"];
  query_pump [label="query_basal"];
  compute_basal [label="compute_basal"];
  cancel_temp_basal [label="cancel_basal"];
  update_temp_basal [label="update_basal"];
  confirm_ack [label="confirm_ack"];
  query_recent_activity [label="verify_activity"];

  // Transitions
  start -> read_cgm;
  read_cgm -> query_pump [label="[BG_available]: BG=read_CGM()"];
  query_pump -> query_pump [label="[query==unsuccessful]"];
  query_pump -> compute_basal [label="[query==successful]: \n current_basal = query_pump()"];

  compute_basal -> cancel_temp_basal [label="[updated_basal<0]: \nupdated_basal = compute_updated_basal(BG, current_basal)"];
  compute_basal -> update_temp_basal [label="[updated_basal!=current_basal]: \nupdated_basal = compute_updated_basal(BG, current_basal)"];
  compute_basal -> confirm_ack [label="[updated_basal==current_basal]: \nupdated_basal = compute_updated_basal(BG, current_basal)"];
  cancel_temp_basal -> confirm_ack [label="[cancellation_issued]"];
  update_temp_basal -> confirm_ack [label="[update_issued]"];

  confirm_ack -> query_recent_activity [label="[ack_confirmed]"];
  query_recent_activity -> read_cgm [label="[activity_verified]"];
}
```

Figure 2: Dot-encoded version of the EFSM

3 Modelling assumptions and test sequences of the EFSM

3.1 State descriptions

- **read_cgm**: Reads the latest blood glucose data from the CGM.
- **query_pump**: Queries the insulin pump for the current basal rate.
- **compute_basal**: Computes the updated basal rate using the blood glucose and the current basal rate.
- **cancel_basal**: Cancels the temporary basal rate if *updated_basal* is negative.
- **update_basal**: Updates the temporary basal rate if *updated_basal* is not equal to *latest_basal*.
- **confirm_ack**: Confirms that the pump received and acknowledged the update.
- **verify_activity**: Verifies recent activity to ensure any new temporary basal rate successfully took effect.

3.2 Modelling assumptions

- The EFSM models a periodic process where OpenAPS continuously interacts with the CGM and the insulin pump to monitor and adjust insulin delivery based on blood glucose levels.
- Failures such as missing CGM data, unsuccessful pump queries, failed acknowledgments, or failed verification trigger retry attempts before escalating to a higher-level failure state or alert.
- Retries are finite and limited to a configurable number of attempts, although this limit is not explicitly represented in the EFSM.
- The computation of the updated basal rate (*compute_basal*) is assumed to always succeed when inputs are valid. Inputs are presumed sanitized.
- For safety, if *confirm_ack* or *verify_activity* fails repeatedly, the system defaults to maintaining the last known safe basal rate.

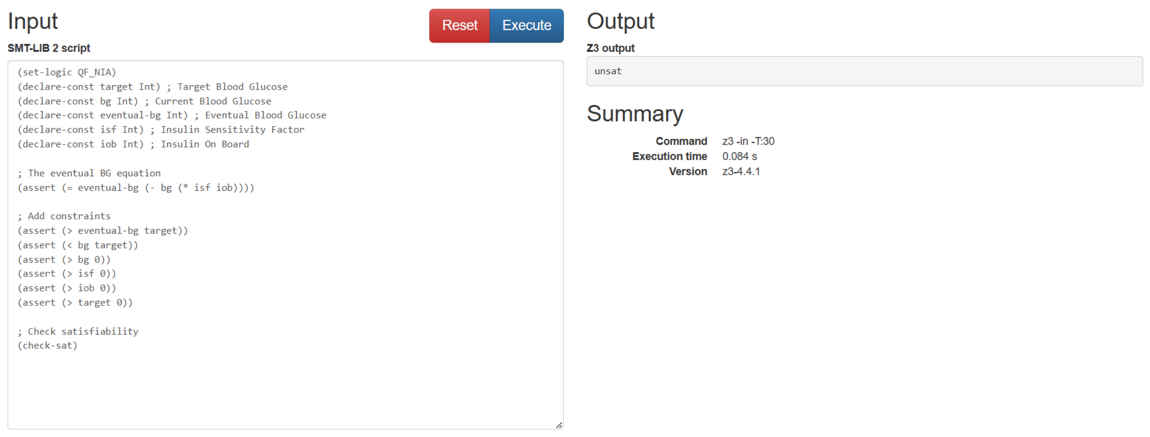
3.3 Test sequences

Below is a set of traces which achieve transition coverage:

- **BG reading with updated basal**: idle, *read_cgm*, *query_pump*, *compute_basal*, *update_basal*, *confirm_ack*, *verify_activity*, *read_cgm*.
- **BG data unavailable with compute basal**: idle, *read_cgm*, *read_cgm*, *query_pump*, *compute_basal*, *cancel_basal*, *confirm_ack*, *verify_activity*, *verify_activity*.

- **Query pump retry and acknowledgment retry with cancel basal**: idle, *read_cgm*, *query_pump*, *query_pump*, *compute_basal*, *cancel_basal*, *confirm_ack*, *confirm_ack*, *verify_activity*, *read_cgm*.
- **No change in basal**: idle, *read_cgm*, *query_pump*, *compute_basal*, *confirm_ack*, *verify_activity*, *read_cgm*.

4 Z3 code



```
(set-logic QF_NIA)
(declare-const target Int) ; Target Blood Glucose
(declare-const bg Int) ; Current Blood Glucose
(declare-const eventual-bg Int) ; Eventual Blood Glucose
(declare-const isf Int) ; Insulin Sensitivity Factor
(declare-const iob Int) ; Insulin On Board

; The eventual BG equation
(assert (= eventual-bg (- bg (* isf iob))))

; Add constraints
(assert (> eventual-bg target))
(assert (< bg target))
(assert (> bg 0))
(assert (> isf 0))
(assert (> iob 0))
(assert (> target 0))

; Check satisfiability
(check-sat)
```

```
unsat
```

```
       Command   z3 -in -T:30
Execution time   0.084 s
       Version   z3-4.4.1
```

Figure 3: Z3 Solver Code

4.1 Discussion

The Z3 solver returned *unsat*, indicating that the constraints are unsatisfiable. This result implies that no positive integer values for *bg*, *ISF*, *IOB*, and *target* simultaneously satisfy the equation and conditions. The unsatisfiability arises from a logical contradiction within the constraints.

The formula

$$\text{eventual\_bg} = \text{bg} - (\text{ISF} \times \text{IOB})$$

inherently decreases the value of *bg*, as $\text{ISF} \times \text{IOB}$ is positive. For $\text{eventual\_bg} > \text{target}$, the reduction caused by $\text{ISF} \times \text{IOB}$ must be small. However, the starting value of *bg* is already less than *target*. This creates an incompatibility, as it is not logically possible for the eventual blood glucose to exceed the target under these circumstances.

These results are significant for testing. They highlight a logical contradiction in the documentation rather than a bug in the implementation or an issue with inputs. Consequently, testers cannot use these constraints with the formula to derive meaningful test inputs. Instead, the findings provide insight into revising the constraints to make them logically consistent.

In summary, this analysis not only uncovers an issue in the documentation but also helps testers focus on feasible test cases by identifying necessary revisions to the specified constraints.