TPLP: Page 1–Y. © The Author(s), 2024. Published by Cambridge University Press 2024
doi:10.1017/xxxxx

SUPPLEMENTARY MATERIAL FOR: Early Validation of High-level System Requirements with Event Calculus and Answer Set Programming

VAŠÍČEK, ONDŘEJ[1] ARIAS, JOAQUIN[2] FIEDOR, JAN[1,7] GUPTA, GOPAL[3]
HALL, BRENDAN[4] KŘENA, BOHUSLAV[1] LARSON, BRIAN[5]
VARANASI, SARAT CHANDRA[6] VOJNAR, TOMÁŠ[1,8]

[1]*Brno University of Technology, CZ* [2]*Universidad Rey Juan Carlos, ES* [3]*UT Dallas, USA*
[4]*Ardent Innovation Labs, USA* [5]*Multitude Corp., USA* [6]*GE Aerospace Research, USA*
[7]*Honeywell International s.r.o., CZ* [8]*Masaryk University, CZ*

Appendix A Additional Details for Sections from the Paper

This appendix contains further details which did not fit into the paper. In particular, a description of how we fixed an overdose issue in the PCA pump specification in Section A.1, an example of decoupling the trigger and effects for an event in Section A.2, and a full version of a table of experiments (instead of a shortened version) in Section A.3.

A.1 Fixing the Overdose Error in the PCA Pump Requirements

[Section 4.2.2] discussed an overdose error that we have discovered in the requirements specification of the PCA pump. In this section, we discuss how we fixed the model in order to remove this issue.

One option to fix the overdose issue is to introduce the missing overdose protection measure for basal delivery, as was the original intention defined in the exception cases in the specification. This solution would prevent harm to the patient, however, it could allow an early spike of boluses only to cause a max dose warning during subsequent basal delivery, which leads to loss of therapy and requires the attention of a clinician. Further, adding a triggered event which can halt basal delivery based on the maximum safe dose, in a similar way as what is already included for a clinician-requested bolus, would cause a significant increase in solving complexity and would turn basal delivery into a self-ending trajectory.

Therefore, we propose a different, preemptive approach. When the pump reasons about denying a bolus, it should consider that (at least) basal delivery will follow under normal operation by *assuming a max dose window full of basal delivery*. The actual amount of extra drug delivered via boluses is then added on top of the assumption. The overdose protection measure defined for clinician-requested boluses needs to be modified in a similar fashion. In this way, there is no need for an overdose protection measure during basal delivery because the preemptive nature of handling boluses will not allow such situation, assuming that the max dose is defined such that basal delivery by itself (without any boluses) will not cause an overdose. Further, the preemptive denial of a bolus should raise the max dose warning only for cases where the overdose would occur immediately

after or during the bolus, not for cases where the overdose was preemptively prevented (i.e., only raise a warning in cases when even the original model would). This approach is what our "fixed" model of the PCA pump currently uses. The two versions of the model can be found in model-original.pl and model-fixed.pl.

A.2 Example for the Decoupling of Event Triggers and Effects

In [Section 5.5], we discussed a multi-run approach for mitigating increases in reasoning complexity caused by certain triggered events. The approach is based on decoupling the trigger of the event from the effects of the event. In this section, we provide a code demonstration of how the decoupling is done. Given some event `alarm`, which is a triggered event which can only be triggered up to once per narrative, it can have effects on some fluents via `initiates`/`terminates` and on other events via triggering their occurrence. The original code for the `alarm` event and its effects might look like this:

```
1 event(alarm).
2 happens(alarm, T) :- some_trigger_rule(T).  % event trigger
3 terminates(alarm, some_fluent, T).          % effect on fluents
4 happens(alarm_response_event, T) :-         % effect on other events
5   happens(alarm, T).
```

In order to decouple the effects of the `alarm` event from its trigger, we replace the above code with the following code:

```
1 event(alarm).                               % original event
2 happens(alarm, T) :- some_trigger_rule(T).  % original event trigger
3
4 event(alarm_EFFECT).                        % new effect event
5 terminates(alarm_EFFECT, some_fluent, T).   % decoupled effect on fluents
6 happens(alarm_response_event, T) :-         % decoupled effect on other events
7   happens(alarm_EFFECT, T).
8
9 happens(alarm_EFFECT,T):-                   % option to undo decoupling
10  full_alarm_reasoning_enabled,
11  happens(alarm,T).
```

We introduce a new `alarm_EFFECT` event and move all the effects of the original `alarm` event to this new event (lines 4-7). The new event has no trigger rule, and the original event now has no effects. We also include an option to undo the decoupling in order to re-enable full reasoning (lines 9-11) via a configuration fact `full_alarm_reasoning_enabled`, which is not defined by default.

For each narrative, we can either choose to ignore reasoning about the `alarm` event entirely (when it is deemed not relevant), or we can use the multi-run approach, or we can re-enable the full (slow) reasoning. For this particular example, the multi-run approach would consist of two runs. The first run would use the query `?- happens(alarm,T).` If the query fails, then the `alarm` event does not occur in the particular narrative and, thus, is not relevant. If the query succeeds, then we introduce a new fact `happens(alarm_EFFECT,t).`, where `t` is the binding for `T` from the result of the query. The second run of reasoning, then uses any query of interest while taking into account the new fact which might have been introduced in the previous run.

In the case of decoupling multiple events, e.g., `E1` and `E2`, one needs to consider if one of the decoupled events has any impact on the occurrence of the other decoupled event. If the occurrence of `E1` influences the occurrence of `E2` in any way, then we need to use three runs in the multi-run approach. The first run queries the occurrence of `E1` and introduces the appropriate fact, the second run then queries the occurrence of `E2` while already considering the effects of `E1` via the new fact, and the third run then runs any query of interest while taking into account facts introduced by both of the prior runs.

A.3 Full Version of [Table 1]

[Section 4.1.1] presented [Table 1], which was a shortened version due to space constraints and due to limited value of presenting execution times of every UC/ExC. We include the full version in Table A1.

Table A1. *Full results of simulation of relevant use cases and exception cases*

	Use Case Name	**Variant**	**Result**	**Time(s.)**
UC1	Normal Operation	no variants	OK	1.69
UC2	Patient-Requested Bolus	no variants	OK	3.17
UC3a	Clinician-Requested Bolus	not suspend	OK	2.84
UC3b	Clinician-Requested Bolus	suspended and resumed	OK	37.13
UC7a	Resume Infusion	after basal delivery	OK	1.92
UC7b	Resume Infusion	after patient bolus	OK	3.83
UC7c	Resume Infusion	after clinician bolus	OK	3.56
ExC1	Bolus Request Too Soon	no variants	OK	2.72
ExC7a	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	1.87
ExC7b	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	1.87
ExC7c	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	2.17
ExC7d	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	4.62
ExC7e	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	1.53
ExC7f	Over-Flow Rate Alarm	defined in ExC step 1	FAIL	4.24
ExC8a	Under-Flow Rate Alarm	basal delivery	OK	1.56
ExC8b	Under-Flow Rate Alarm	defined in ExC step 1	FAIL	2.19
ExC8c	Under-Flow Rate Alarm	defined in ExC step 1	FAIL	2.15
ExC9a	Pump Overheating	during basal delivery	OK	1.68
ExC9b	Pump Overheating	during patient bolus	OK	2.88
ExC9c	Pump Overheating	during clinician bolus	OK	2.31
ExC10a	Downstream Occlusion	during basal delivery	OK	1.67
ExC10b	Downstream Occlusion	during patient bolus	OK	3.15
ExC10c	Downstream Occlusion	during clinician bolus	OK	2.32
ExC11a	Upstream Occlusion	during basal delivery	OK	1.66
ExC11b	Upstream Occlusion	during patient bolus	OK	3.10
ExC11c	Upstream Occlusion	during clinician bolus	OK	2.35
ExC12a	Air-in-line Embolism	during basal delivery	OK	1.66
ExC12b	Air-in-line Embolism	during patient bolus	OK	2.91
ExC12c	Air-in-line Embolism	during clinician bolus	OK	2.31
ExC13a	Maximum Safe Dose	during basal delivery	FAIL	25.62
ExC13b	Maximum Safe Dose	during patient bolus	OK	31.61
ExC13c	Maximum Safe Dose	during clinician bolus	OK	53.45
ExC20a	Reservoir Low	during basal delivery	FAIL	1.99
ExC20b	Reservoir Low	during patient bolus	FAIL	7.95
ExC20c	Reservoir Low	during clinician bolus	FAIL	5.07
ExC21	Reservoir Empty	no variants	OK	40.99