Towards a Formal Model of Type 1 Diabetes for Artificial Intelligence

D. Brown, C. Martin, D. Duce, A. Aldea, R. Harrison

Department of Computing and Communication Technologies,
Oxford Brookes University, Wheatley, Oxford, OX33 1HX, England
dbrown@brookes.ac.uk

Abstract. Artificial Intelligence (AI) is potentially useful for cost effective diabetes self-management. One research priority for the development of robust and beneficial AI concerns the use of formal verification techniques to model such self-modifying systems. In the context of diabetes, formal methods may also have a role in fostering trust in the technology as well as facilitating dialogue between a multidisciplinary team to determine system requirements in a precise way. In this paper we show how the formal modelling language Event-B can be used to capture safety-critical constraints associated with AI systems for diabetes management.

1 Introduction

Most people with Type 1 Diabetes (T1D) have to perform complex insulin dose calculations several times a day. The computations must consider multiple factors and can occur in a variety of contexts that might affect cognitive load. Adaptive solutions that use artificial intelligence (AI) to replace this human decision-making are emerging [5, 6], but the application of such technology to safety-critical healthcare problems is often met with high levels of resistance [4]. Individuals are concerned about loss of control and question whether the perceived risk is outweighed by the potential benefit.

Such systems must therefore behave robustly and deliver the intended benefits consistently in order to foster trust. Some computer science research priorities that have been identified towards this goal include formal *verification* and *validation* [7]. Although features of AI systems, such as nondeterminism, can make them difficult to verify, it should be possible to build them from individual components that have been proved correct. Additionally, they might be used within the confines of a deterministic safety systems that have been formally verified.

Another dimension is introduced by the user interface of AI systems. Tools exist to support the formal design and analysis of human-machine interfaces, including the behaviour of objects such as buttons and keyboards [3]. Such technologies can be used to give a lightweight, formal analysis of the safety requirements of the interface. They may therefore have a role in facilitating dialogue between a multidisciplinary team, which could include clinicians and computer scientists, in evolving a common understanding of these requirements in a precise

way. In this paper we present a simplified formal specification of part of a T1D bolus recommender system in an attempt to show how some properties might be expressed. This might form part of a model that a trustworthy AI system must guarantee not to violate.

2 Formal Specification

Event-B is a formal modelling language developed by Jean-Raymond Abrial as an evolution of the B-Method [1]. The purpose of modelling with Event-B is to prove that a model will work prior to implementation and for early identification of problems with the system requirements. The Rodin Platform is an Eclipse-based integrated development environment (IDE) for Event-B modelling. It has an associated repository of resources to aid existing and new users of the platform, including a plug-in to integrate ProB into the IDE, which provides a method to systematically check the model for errors through animation. Other plug-ins include automated code generation from the Event-B specification and external provers to assist the automatic proofs. For example, the Atelier B prover plug-in was used in the work described below.

An Event-B model is validated through the successful discharge of *proof obligations*, which imply that the invariants and type declarations of the model are not violated by any aspect of the model. For example, if a variable x has type $\mathbb{N}1$, the assignment of 0 to x would fail to discharge the proof obligation created by $x \in \mathbb{N}1$ since x must be greater than or equal to 1. A key feature of the Event-B language is its incremental modelling process through *refinement*. Refinement allows gradual development of the specification starting at an abstract level, which can then be refined to include new functionality and gradually move the model towards the concrete implementation.

2.1 Event-B Specification of T1D System Constraints

The specification begins with an abstract specification of the system, focussing on the constraints of the system variables. Figure 1 provides an extract of the variables and invariants that includes the maximum bolus dose, and upper and lower target blood glucose ranges. A full specification of the variables and invariants is included in [2]. Event-B is limited to natural and integer numerical types, which presents a problem when decimal values are required. To overcome this, all natural numbers or integers in the specification are represented through multiplication by the power of 10, allowing precision to one decimal place (e.g. 5.5 is represented as 55). Comments in the specification will be used help to reinforce this representation.

The types of *maxBolus*, *targetRangeUpper* and *targetRangeLower* are defined in Figure 1 as natural number ($\mathbb{N}$). This invariant alone implies that the value of these variables must be greater than or equal to 0, and that negative values are not permitted. *inv2* states that *maxBolus* is restricted to a maximum bolus value of 50.0 and *inv7* ensures that the target range is non-empty.

MACHINE t1dm_m0
VARIABLES
`maxBolus` Maximum bolus dose limit
`targetRangeUpper` Target blood glucose range upper value
`targetRangeLower` Target blood glucose range lower value
INVARIANTS
`inv1` : $maxBolus \in \mathbb{N}$
`inv2` : $maxBolus \leq 500$
The maximum bolus dose must be in the range [0.0,50.0] IU
`inv3` : $targetRangeUpper \in \mathbb{N}$
`inv4` : $targetRangeUpper \geq 55 \land targetRangeUpper \leq 150$
Target blood glucose range upper value must be in the range [5.5,15.0]
`inv5` : $targetRangeLower \in \mathbb{N}$
`inv6` : $targetRangeLower \geq 30 \land targetRangeLower \leq 80$
Target blood glucose range upper value must be in the range [3.0,8.0]
`inv7` : $targetRangeLower \leq targetRangeUpper$
Target blood glucose range upper value must be in the range [3.0,8.0]

Fig. 1. Extract of machine variables and invariants

Both of the invariants for *maxBolus* state that $0 \leq maxBolus \leq 50$ insulin units. Additionally, the invariants for *targetRangeUpper* and *targetRangeLower* also define the constraints: $5.5 \leq targetRangeUpper \leq 15.0$ mmol/L, and $3.0 \leq targetRangeLower \leq 8.0$ mmol/L. These invariants prevent the variables from breaching the constraints imposed by the Accu-Chek® Aviva Expert blood glucose meter.

MACHINE t1dm_m0
EVENTS
Event *bolusCalc* $\widehat{=}$
Bolus calculator event for instances where s is > 0 and $<$ maxBolus
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{N}$
`grd2` : $s \leq maxBolus$
The bolus solution suggestion is $\leq$ maxBolus
then
`act1` : $bolusSuggestion := s$
Sets the bolus suggestion value to parameter s
end

Fig. 2. Abstract bolus calculation event

An abstract event for the bolus suggestion *bolusCalc* is described by Figure 2. One parameter is specified for the event, the bolus suggestion s. The event has two guards which check that parameter s is a natural number and that s is less

than or equal to the maximum bolus dose $maxBolus$:

$$bolusSuggestion \in \mathbb{N} \wedge bolusSuggestion \leq maxBolus$$

The *bolusCalc* event does not satisfy the requirements of the application alone as it is possible that the bolus suggestion may be negative or greater than the maximum bolus dose. In these circumstances, the bolus dose should be set to 0 and the *maxBolus* respectively. To model this, two new events are added to the model to allow for these circumstances *bolusCalcNeg* and *bolucCalcMax*. These events are shown in Fig. 3.

MACHINE t1dm_m0
EVENTS
Event *bolusCalcNeg* $\widehat{=}$
Bolus calculator event for instances where s is < 0
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{Z}$
`grd2` : $s < 0$
The bolus solution suggestion is < 0
then
`act1` : $bolusSuggestion := 0$
Sets the bolus suggestion to 0 as negative values are not permitted
end
Event *bolusCalcMax* $\widehat{=}$
Bolus calculator event for instances where s is $>$ maxBolus
any
s
s is the parameter of bolus suggestion
where
`grd1` : $s \in \mathbb{N}$
`grd2` : $s > maxBolus$
The bolus solution suggestion is $>$ the maximum bolus dose
then
`act1` : $bolusSuggestion := maxBolus$
Sets the bolus suggestion to maxBolus as values $>$ maxBolus are not permitted
end

Fig. 3. Additional abstract bolus calculation events

At present, the model does not include the computation to be performed to determine the bolus dose, but instead defines the abstract events required. This abstract model can now be refined into a more concrete one. In [2] it is refined to include a case base and deterministic bolus calculator. All proofs are automatically discharged by the prover and animation with ProB indicates that

the events are only activated by parameters which satisfy the guards. The same technique could be applied to the refinement for a nondeterministic calculator including AI.

3 Conclusion

This short example illustrates how critical safety constraints of a T1D bolus calculator involving AI can be captured formally. We believe that the construction of a validated, definitive core task model for the T1D individual and associated recommender system, verified using formal techniques, should be a research priority for artificial intelligence in diabetes. However, there are more difficult challenges related to the ability to verify AI systems that modify themselves, possibly repeatedly, over time. It is not yet known whether straightforward verification tools can be applied to this broader setting [7].

Acknowledgments

This work has received funding from the EU Horizon 2020 research and innovation programme under grant agreement No 689810.

References

1. Abrial, J.R., Abrial, J.R.: The B-book: assigning programs to meanings. Cambridge University Press (2005)
2. Brown, D.: Temporal Case-based Reasoning for Insulin Decision Support. Ph.D. thesis, Department of Computing and Communication Technologies, Oxford Brookes University (2015)
3. Fayollas, C., Martinie, C., Palanque, P., Masci, P., Harrison, M.D., Campos, J.C., e Silva, S.R.: Evaluation of Formal IDEs for Human-machine Interface Design and Analysis: the case of CIRCUS and PVSio-web. In: Dubois, C., Masci, P., Méry, D. (eds.) Proceedings of the Third Workshop on Formal Integrated Development Environment, F-IDE. EPCTS 240 (2016)
4. Hengstler, M., Enkel, E., Duelli, S.: Applied artificial intelligence and trustthe case of autonomous vehicles and medical assistance devices. Technological Forecasting and Social Change 105, 105–120 (2016)
5. Herrero, P., Lopez, B., Martin, C.: PEPPER: Patient Empowerment Through Predictive Personalised Decision Support. . In: Proceedings of the First ECAI Workshop on Artificial Intelligence (2016)
6. Herrero, P., Pesl, P., Reddy, M., Oliver, N., Georgiou, P., Toumazou, C.: Advanced insulin bolus advisor based on run-to-run control and case-based reasoning. Biomedical and Health Informatics, IEEE Journal of 19(3), 1087–1096 (2015)
7. Russell, S., Dewey, D., Tegmark, M.: Research priorities for robust and beneficial artificial intelligence. AI Magazine 36(4), 105–114 (2015)