Using Event-B Refinement to Verify a Control Strategy

Michael Butler, University of Southampton

May 2009

1 Introduction

In this paper we outline some on-going research on approximating continuous behaviour in Event-B [1] in order to model and reason about a control system. We make a distinction between a control *goal* and a control *strategy*. We outline how both can be modelled and how refinement can be used to prove that a strategy satisfies a goal.

We present the approach using a concrete example of a control system for a water tank system.

> The control goal is to maintain the water level between a high level, H, and a low level, L.

Water may flow out of the tank according to some known maximum rate. The controller can switch on a pump in order to increase the level in order to maintain a satisfactory water level.

> The control strategy is to sense the water level periodically and switch the pump on or off as appropriate.

We find it useful to follow the categorisation of modelling variables given in the Four-variable model of Parnas and Madey [5]. In this model, there are two main groupings of variables, *environment* variables and *controller* variables. Environment variables represent quantities in the environment of the controller. Controller variables represent quantities inside the controller machine. Each grouping has two kinds of variable. The two kinds of environment variable are as follows:

Monitored variables Environmental quantites whose value is not determined by the controller but that can be monitored. For example, the water level in the tank is a monitored variable.

Controlled variables Environmental quantites whose value is expected to be determined by the controller. For example the pump status (*on* or *off*) is a controlled variable.

The two kinds of controller variable are as follows:

Input variables Values stored in the controller and provided by some sensor. For example, the sensed water level is an input variable.

Output variables Values used to actuate the controlled environment variables as determined by the controller. For example the pump actuation register is an output variable.

The strategy we follow is to start with a model of the control goal expressed in terms of monitored variables. We then refine this by a model of the strategy which is expressed in terms of monitored and controlled variables. Note that the strategy is modelled purely in terms of environment variables, before introducing the input and output controller variables. The input and output variables may be introduced in later refinements though this is not addressed here.

2 Modelling the Control Goal

The control goal will be modelled in terms of monitored variables. We model a monitored variable as a time-varying value. In Event-B we approximate realtime using a discrete time domain. We use a simple form of timed automaton, where the state variables include a clock, clk, and a monitored variable, m, specified as a time-varying function from time zero up to the current time:

$$\begin{array}{rcl} clk & \in & \mathbb{N} \\ m & \in & 0..clk \rightarrow V \end{array}$$

By using a time-varying function we can represent the required control goal as an invariant on the monitored variable:

inv : $\forall t \cdot t \in 0 \mathinner{..} clk \Rightarrow P(m(t))$

For example, we use the following invariants to model the water level goal:

inv1 : $clk \in \mathbb{N}$

inv2 : $wl \in 0 \mathinner{..} clk \rightarrow \mathbb{N}$ history of the water level up to current time

inv3 : $\forall t \cdot t \in 0 \mathinner{..} clk \Rightarrow L \leq wl(t)$

inv4 : $\forall t \cdot t \in 0 \mathinner{..} clk \Rightarrow wl(t) \leq H$

In general, to model the dynamics of the system, we use an *environment* event that updates the clock and updates the monitored variables in a way that satisfies the goal P:

Event *UpdateMonitored* $\widehat{=}$

any

v

where

grd : $P(v)$

then

act1 : $m(clk+1) := v$

act2 : $clk := clk + 1$

end

For example, in the water tank example, we use the following environment event to represent the required dynamics in terms of the change in water level on one time unit:

Event *UpdateWaterLevel* $\widehat{=}$

any

l

where

grd1 : $L \leq l \wedge l \leq H$

then

act1 : $wl(clk+1) := l$

act2 : $clk := clk + 1$

end

It is important to note that altough the clock is discrete, it may be much more fine-grained than the duration of the periodic control cycle. It is an approximation of real time and, as we will see later, this approximation still allows us to gain a good understanding of why a control strategy achieves a goal.

3 Modelling the Control Strategy

To model the strategy we introduce controlled variables. For example, in the water tank we introduce a *pump* variable representing the status of the pump (*on* or *off*). Along with the controlled variables, we introduce *controller* events that model the strategy to be followed by the controller in order to achieve the control goal. For the water tank example, we adopt the following strategy rules at each control period:

C1 When the level goes below a low threshold LT, the pump is switched on.

C2 When the level goes above a high threshold HT, the pump is switched off.

Since the monitored variables are outside the control of the controller, we need to make assumptions about how much the monitored variables can change in one unit of time. For simplicity, we assume that possible changes in a monitored variable in a single unit of time are bound by some constant value. For example, in the water tank example, we make the following assumptions about the possible changes in water level:

A1 When the pump is off, the water level may decrease by a maximum amount DD within 1 time unit.

A2 When the pump is on, the water level may increase by a maximum amount DU within 1 time unit.

These assumptions will be incorporated into our modelling as refined environment events. We will see that the assumptions allow us to reason about the maximum difference between the water level at two time points.

We assume that the control period never exceeds C time units. That is, the maximum delay between successive control events is C. To incorporate this into our modelling, we will bound the number of environment events that may happen in between each control event. To do this we introduce an *auxiliary* variable that represents a time stamp at which the most recent control event ocurred. The environment events are guarded to ensure that the difference between the current time and the time of the last control event does not exceed the cycle duration C.

To model the strategy for the water tank in the refined model we add controlled variable *pump* and auxiliary variable ct. These satisfy the following invariants:

inv5 : $pump \in BOOL$ FALSE means pump is off

inv6 : $ct \geq 0$ timestamp of the most recent control event

inv7 : $ct \leq clk$

inv8 : $clk \leq ct + C$ $clk - ct$ is bounded by C

Corresponding to the two assumptions on the the change in water level in a single unit of time (A1, A2), we have two refinements of the abstract *UpdateWaterLevel* event. One refinement of this event models the decrease in water level in one time unit (A1):

Event *DecreaseWaterLevel* $\widehat{=}$

refines *UpdateWaterLevel*

any

l

where

grd1 : $clk < ct + C$

grd2 : $pump = FALSE$

grd3 : $wl(clk) - DD \leq l$

grd4 : $l \leq wl(clk)$

then

act1 : $wl(clk + 1) := l$

act2 : $clk := clk + 1$

end

In this event, *grd1* enforces the time bound on environment events. The event deals with the case where the pump is off (*grd2*). The new value of the water level, represented by parameter l, is between the current level and the current level minus DD (*grd3* and *grd4*). The other refinement of *UpdateWaterLevel* deals with the case where the pump is on (A2). *IncreaseWaterLevel* is similar to *DecreaseWaterLevel* except that the water level is increased by a maximum amount DU.

We introduce new *control* events to represent the control decisions. A control event may modify the pump status (controlled variable) and must update the control timestamp auxiliary variable. We introduce one event for each of three cases: switch the pump on (C1), switch the pump off (C2) and leave the pump status unchanged. Consider the case of switching the pump on. We could model this as follows:

Event *ControlLow* $\widehat{=}$

when

grd1 : $wl(clk) \leq LT$

then

act1 : $pump := TRUE$

act2 : $ct := clk$

end

Here *grd1* means we are dealing with the case where the water level is below the low threshold (C1). The effect is to switch the pump on (*act1*) and set the timestamp variable (*act2*).

However this model of the control event is too strong. At this point, the control event represents the point at which the pump is actuated. In the real controller the decision whether or not to actuate will be based on a input variable that was sensed some time earlier than the actuation. We will assume that the difference between the actuation time and the sensing time is bounded by a constant A. A weaker, more feasible specification of this event will base the decision on the water level at an earlier time than the current time:

Event *ControlLow* $\widehat{=}$

any

a sensing time is a units earlier

where

grd1 : $0 \leq a \wedge a \leq A$

grd2 : $a \leq clk$

grd3 : $ct \leq clk - a$

grd4 : $wl(clk - a) \leq LT$

then

act1 : $pump := TRUE$

act2 : $ct := clk$

end

Here the sensing time is $clk - a$ where a is bounded by the constant A.

There is a similar *ControlHigh* event that switches the pump off when $wl(clk - a) \geq HT$ (C2). The third control event *ControlMedium* deals with the case where the pump does not need to be switched. The guards of *ControlMedium* corresponding to *grd4* of *ControlLow* above are as follows:

grd4 : $pump = FALSE \Rightarrow wl(clk - a) > LT$

grd5 : $pump = TRUE \Rightarrow wl(clk - a) < HT$

That is, when the pump is off and the sensed water level is above LT, then it is ok to leave the pump unchanged (*grd4*). Similary for when the pump is on (*grd5*).

4 Reasoning about the Control Strategy

So now we turn our attention to what needs to be proved and how it is proved. In the refinement we changed the guards of the environment events. To see this, compare again the guards of the abstract *UpdateWaterLevel* event with the guards of the refined *DecreaseWaterLevel* event. The abstract guards simply specify that a new water level is chosen to satisfy the control goal. The concrete guards specify that the new water level is less than the current level by a bound DD. The proof obligation associated with refinement is the prove that the refined guards imply the abstract guards. However, the concrete guards on their own do not imply the abstract guards, i.e., they do not guarantee that the control goal continues to be satisfied. To be able to discharge the proof obligations we need to discover gluing invariants such that the invariants and the concrete guards together imply the abstract guards.

The refined environment events are guarded by the condition that the difference between the control timestamp and the current time is bounded by the cycle time C, e.g., *grd1* of *DecreaseWaterLevel*. This bound on the number of environment events in between control events allows us to derive overall bounds on the increase and decrease in the water level in between control events. The

key insight is that we can express these bounding properties as invariants. We separate the invariants into those dealing with lower bounds on water level and upper bounds. The invariants for the lower bounds are as follows:

$$\textbf{\textit{inv9}} \ : \ pump = FALSE \Rightarrow \\ (\forall i, j \cdot ct \leq i \wedge i \leq j \wedge j \leq clk \Rightarrow wl(j) \geq wl(i) - (DD \times (j - i)))$$

$$\textbf{\textit{inv10}} \ : \ pump = TRUE \Rightarrow \\ (\forall i, j \cdot ct \leq i \wedge i \leq j \wedge j \leq clk \Rightarrow wl(j) \geq wl(i))$$

Here *inv9* specifies that in the case that the pump is off, then the difference between the water level at two time points i and j is given by $DD \times (j - i)$ (i is earlier than j). That is the water level at point j is below the level at point i by at most $DD \times (j - i)$. This invariant is maintained by the environment events because the the decrease in water level in one time unit is bounded by DD (*DecreaseWaterLevel* event). Also, *inv10* specifies that when the pump is on then the water level at point j will not be below the water level at point i.

Now the proof obligation for the environment events is that the new water level does not go below L. While the pump is on this will be easily maintained. While the pump is off, we need to know that the water level at the point of the most recent control event exceeds L by at least the maximum possible decrease in between control events. This is characterised by the following invariant:

$$\textbf{\textit{inv11}} \ : \ pump = FALSE \Rightarrow wl(ct) \geq L + (DD \times C) + DD$$

Invariants *inv4*, *inv5*, *inv6* are sufficient to discharge the proof obligations associated with the *DecreaseWaterLevel* events.

We require similar invariants for the *IncreaseWaterLevel* environment event that characterise upper bounds on the water level:

$$\textbf{\textit{inv12}} \ : \ pump = TRUE \Rightarrow \\ (\forall i, j \cdot ct \leq i \wedge i \leq j \wedge j \leq clk \Rightarrow wl(j) \leq wl(i) + (DU \times (j - i)))$$

$$\textbf{\textit{inv13}} \ : \ pump = FALSE \Rightarrow \\ (\forall i, j \cdot ct \leq i \wedge i \leq j \wedge j \leq clk \Rightarrow wl(j) \leq wl(i))$$

$$\textbf{\textit{inv14}} \ : \ pump = TRUE \Rightarrow wl(ct) \leq H - (DU \times C) - DU$$

As part of the proof of the refinement, we needed to make assumptions about the relationships between the important constants such as H, HT, C, etc. The role of these constants and their relationships are as follows:

CONSTANTS

$\boldsymbol{L}$	minimum level
$\boldsymbol{H}$	maximum level
$\boldsymbol{LT}$	low threshold
$\boldsymbol{HT}$	high threshold

$\boldsymbol{DU}$	increase in water level in one time unit
$\boldsymbol{DD}$	decrease in water level in one time unit
$\boldsymbol{C}$	maximum time between successive control events
$\boldsymbol{A}$	maximum time between sample time and control time

AXIOMS

$\boldsymbol{axm1}$: $L < LT \wedge LT < HT \wedge HT < H$

$\boldsymbol{axm2}$: $A \leq C$

$\boldsymbol{axm3}$: $LT \geq L + (DD * (A + C + 1))$

$\boldsymbol{axm4}$: $HT \leq H - (DU * (A + C + 1))$

END

Here *axm3* specifies a minimum bound on the difference between the low level and the low threshold. When the pump is off and the sensed level is above the low threshold, the control events will leave the pump off. In this case we need to be sure that the level will not go below L before the next control event (in C time units). The maximum time between the sensing time and the next control time is $A + C$ so the maximum drop in level is $DD * (A + C)$[1].

Note that *axm3* and *axm4* where not specified a priori. Instead they were derived as part of the proof effort by inspecting failed proof steps. One could say that these constraints were synthesised as part of the proof effort.

5 Concluding

Although we have approximated continuous behaviour with a discrete clock, we believe the form of modelling and refinement proof oultined in this paper has an engineering value. We clearly separated the control goal form the control strategy and were able to distinguish control events from environment events. The environment events represent assumptions about the dynamic behaviour of the environment. The constraint axioms (*axm1* to *axm4*) represent assumptions about timing constraints and changes to monitored values within a fixed time unit. Note that we have not worked with a given time unit nor cycle time. The value of the cycle time is not important for our reasoning. What is important is the relationship between the cycle time and the degree of change possible in monitored variables in a time unit. The choice of actual time unit and cycle time should be such that the synthesised contraints (*axm1* to *axm4*) are satisfied.

The other engineering contribution that we belive the approach provides is the insight given by the invariants (*inv9* to *inv14*). As with loop invariants used in program correctness, these invariants provide insight into why the control strategy satisfied the control goal.

[1]The +1 in *axm3* is a result of over-caution by the author.

We have focused on expression of control goals that refer to individual time points, i.e., the monitored variables are required to satisfy some property at each time point. Treatment of properties over time intervals, as expressible in the Duration Calculus [3] or the approach of Hayes, Jackson and Jones [4], merits further investigation.

The approach outlined here builds on the approach of [2]. In that paper, actions systems (the basis of Event-B) are used to construct a model of the system that includes the monitored variables and the controlled variables. The difference between the present work and that of [2] is that [2] uses a much simpler modelling of time whereby the environment actions model the update to monitored variables in one complete control cycle. Our environment events operate at a much more fine-grained unit of time than the control cycle. Other differences are that, in the current work, we model the goal independently of the strategy and we use a time-varying function to model the environment variable. Use of the time-varying function and the fine-grained environment timing allows us to reason about the correctness of the strategy wrt the goal – at least to a strong degree of approximation.

What is clearly missing from this approach is any treatment of faults (e.g., pump or sensor failure) and fault tolerance. This will require further research.

References

[1] J.-R. Abrial. *Modelling in Event-B: System and Software Engineering.* To be published by Cambridge University Press, 2009.

[2] M. J. Butler, E. Sekerinski, and K. Sere. An action system approach to the steam boiler problem. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications – Specifying and Programming the Steam Boiler Control, LNCS 1165*, pages 129–148. Springer-Verlag, Berlin, 1996.

[3] Zhou Chaochen and Michael R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems.* Springer-Verlag, 2003.

[4] Ian Hayes, Michael Jackson, and Cliff Jones. Determining the specification of a control system from that of its environment. In Keijiro Araki, Stefani Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, volume 2805 of *LNCS*, pages 154–169. Springer Verlag, 2003.

[5] David Lorge Parnas and Jan Madey. Functional documents for computer systems. *Sci. Comput. Program.*, 25(1):41–61, 1995.