Chapter 8
Events & Responses

Synopsis Having already understood the analytical implications of control loops in cyber-physical systems via their logical characterizations with unwinding and invariants, this chapter investigates their impact on the important design paradigm of *event-triggered control systems*, also known as event-driven control systems. In such a system the controllers respond to certain events whenever they happen. The resulting event detection for the various events of interest is then executed in a control loop. A safe controller makes the appropriate response for each of the events of relevance. This direct response principle for the respective events provides systematic ways of designing event-triggered CPS controllers and leads to relatively simple safety arguments. But event-triggered systems are hard if not impossible to implement, because they require perfect event detection. That makes this chapter an ideal setting for a number of crucial modeling lessons for CPS.

8.1 Introduction

Chapter 3 already saw the importance of control and loops in CPS models, Chap. 5 presented a way of unwinding loops iteratively to relate repetition to runs of the loop body, and Chap. 7 finally explained the central inductive proof principle for loops using invariants.

That has been a lot of attention on loops, but there are even more things to be learned about loops. This is no coincidence, because loops or other forms of repetitions are one of the most difficult challenges in CPS [3–6]. The other difficult challenge comes from the differential equations. If differential equations are simple and there are no loops, CPSs suddenly become easy (they are even decidable [4]).

This chapter will focus on how these two difficult parts of CPS interact: how loops interface with differential equations. That interface is ultimately the connection between the cyber and the physical part, which, as we have known since Chap. 2, is fundamentally represented by the evolution domain constraints that determine when physics pauses to let cyber look and act.

A. Platzer, *Logical Foundations of Cyber-Physical Systems*,
https://doi.org/10.1007/978-3-319-63588-0_8

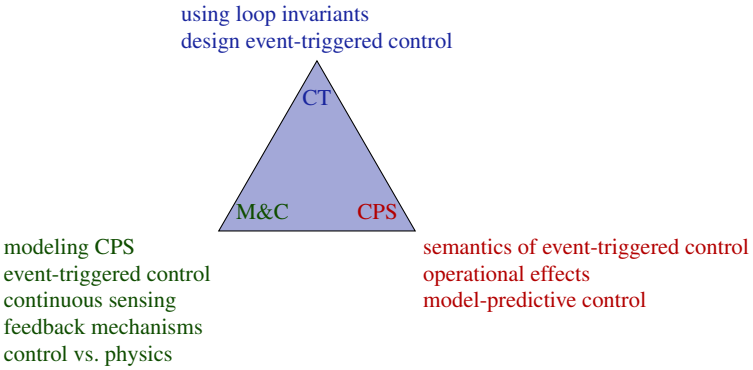
This and the next chapter focus on two important paradigms for making cyber interface with physics to form cyber-physical systems. The two paradigms play an equally important rôle in classical embedded systems. One paradigm is that of *event-triggered control*, where responses to events dominate the behavior of the system, and an action is taken whenever one of the events is observed. The other paradigm is *time-triggered control*, which uses periodic actions to influence the behavior of the system at certain frequencies. Both paradigms follow naturally from an understanding of hybrid program principles for CPS. Event-triggered control will be studied in this chapter, while time-triggered control will be pursued in the next chapter. Both chapters come with complementary lessons that are important in the design of virtually any CPS model and controller but are also important in simple embedded systems. Events and correct responses to events will be our challenge of choice for this chapter.

Based on the understanding of loops from Chap. 7, the most important learning goals of this chapter are:

Modeling and Control: This chapter provides a number of crucial lessons for modeling CPSs. We develop an understanding of one important design paradigm for control loops in CPS: event-triggered control. The chapter studies ways of developing models and controls corresponding to this feedback mechanism, which is based on issuing appropriate control responses for each of the relevant events of a system. This will turn out to be surprisingly subtle to model. The chapter focuses on CPS models with continuous sensing, i.e., we assume that sensor data is available and can be checked always.

Computational Thinking: This chapter uses the rigorous reasoning techniques for CPS loops with invariants from Chap. 7 based on the axiomatic reasoning approach from Chap. 5 to study CPS models with event-triggered control. As a running example, the chapter builds on the bouncing ball that has served us so well for conveying subtleties of hybrid system models in an intuitive example. This time, we add control decisions to the bouncing ball, turning it into a vertical ping-pong ball, which retains the intuitive simplicity of the bouncing ball, while enabling us to develop generalizable lessons about how to design event-triggered control systems correctly. While the chapter could hardly claim to show how to verify CPS models of any appropriate scale, the foundations laid in this chapter definitely carry significance for numerous practical applications, because the design of safe event-triggered controllers follows the same principles developed in a simple representative example here. It is easier to first see how cyber and physics interact in an event-triggered way for the significantly simpler and familiar phenomenon of bouncing balls, which provides the same generalizable lessons but in a simpler setting.

CPS Skills: This chapter develops an understanding of the precise semantics of event-triggered control, which can often be surprisingly subtle even if superficially simple. This understanding of the semantics will also guide our intuition of the operational effects caused by event-triggered control. Finally, the chapter shows a brief first glimpse of higher-level model-predictive control and design by invariant, even if a lot more can be said about that topic.



8.2 The Need for Control

Having gotten accustomed to the little acrophobic bouncing ball Quantum since Chap. 4, this chapter will simply stick to that. Yet, Quantum asks for more action now, for he had so far no choice but to wait until he was down on the ground at height $x = 0$. When his patience paid off so that he finally observed height $x = 0$, then his *only* action was to make his velocity bounce back up. Frustrated by this limited menu of actions to choose from, Quantum begs for a ping-pong paddle. Thrilled at the opportunities opened up by flailing around with a ping-pong paddle, Quantum first performs some experiments to use it in all kinds of directions. But he never knew where he was going to land if he tried the ping-pong paddle sideways so he quickly gave up the thought of sideways actuation. The ball probably got so accustomed to his path of going up and down on the spot that he embraced the thought of keeping it that way. With the help of the ping-pong paddle, Quantum has high hopes to do the same, just faster without risking the terrified moments inflicted on him by his acrophobic attitude to heights. Setting aside all Münchausian concerns about how effective ping-pong paddles can be for the ball if the ball is using the paddle on itself in light of Newton's third law about opposing forces, let us investigate this situation regardless.[1] After all, the ping-pong-crazy bouncing ball Quantum still has what it takes to make control interesting: the dynamics of a physical system and decisions on when to react and how to react to the observed status of the system.

Chapter 7 proved the undamped bouncing ball with repetitions (shown in Fig. 8.1):

[1] If you find it hard to imagine a bouncing ball that uses a ping-pong paddle to pat itself on its top to propel itself back down to the ground again, just step back and consider the case where the ping-pong ball has a remote control to activate a device that moves the ping-pong paddle downwards. That will do just as well, but is less fun. Besides, Baron Münchhausen would surely be horribly disappointed if we settled for such a simple explanation for the need for control.

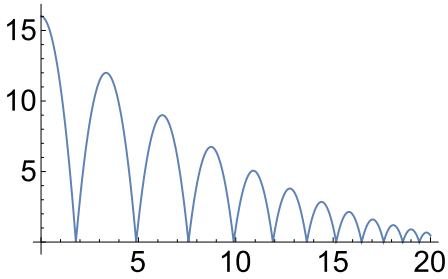


Fig. 8.1 Sample trajectory of a bouncing ball bouncing freely (plotted as position over time)

$$0 \leq x \land x = H \land v = 0 \land g > 0 \land 1 = c \rightarrow$$
$$[(\{x' = v, v' = -g \,\&\, x \geq 0\}; (?x = 0; v := -cv \cup ?x \neq 0))^*](0 \leq x \land x \leq H) \tag{7.12*}$$

With this pretty complete understanding of bouncing balls, let's examine how to turn the simple bouncing ball into a fancy ping-pong ball using clever actuation of a ping-pong paddle. Quantum tried to actuate the ping-pong paddle. By making the ping-pong paddle solely move up and down, Quantum ultimately figured out that the ball would go back down pretty fast as soon as he got a pat on the top from the paddle. He also learned that the upwards direction turned out to be not just difficult but also rather dangerous. Moving the ping-pong paddle upwards from underneath the ball was rather tricky and only made the ball fly up even higher than before. Yet, that is what the acrophobic bouncing ball Quantum did not enjoy at all, so he only ever used the ping-pong paddle to push the ball downwards. Moving the ping-pong paddle sidewards would make the bouncing ball leave its favorite path of going up and down on the same spot.

8.2.1 Events in Control

As a height that Quantum feels comfortable with, he chooses the magic number 5 and so he wants to establish $0 \leq x \leq 5$ to always hold as Quantum's favorite safety condition. The ball further installs the ping-pong paddle at a similar height so that he can actuate somewhere between height 4 and 5. He exercises great care to make sure he only moves the paddle downwards when the ball is underneath, never upwards when it is above, because that would take him frightfully high up. Thus, the effect of the ping-pong paddle will only be to reverse the ball's direction. For simplicity, Quantum figures that being hit by a ping-pong paddle might have a similar effect to being hit by the floor, except with a possibly different bounce factor $f \geq 0$ instead of the damping coefficient c.[2] So the paddle actuated this way is simply assumed to have the effect $v := -fv$. Since Quantum can decide to use the ping-pong paddle as

[2] The real story is a bit more complicated, but Quantum does not know any better yet.

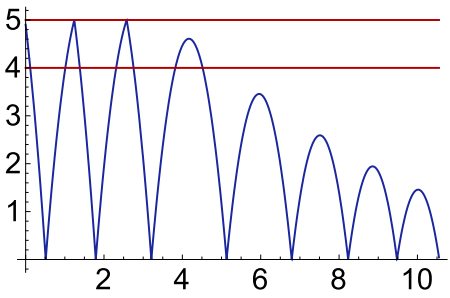


Fig. 8.2 Sample trajectory of a ping-pong ball (plotted as position over time) with the indicated ping-pong paddle actuation range

he sees fit (within the ping-pong paddle's reach between height 4 and 5), the ping-pong model is obtained from the bouncing-ball model by adding this additional (nondeterministic) choice to the HP. A sample trajectory for the ping-pong ball, where the ping-pong paddle is used twice is illustrated in Fig. 8.2. Observe how the use of the ping-pong paddle (here only at height $x = 5$) makes the ball bounce back faster.

Taking these thoughts into account, the ball devises a new and improved HP for ping-pong and conjectures its safety as expressed in the following dL formula:

$$\begin{aligned} &0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow \\ &\quad \big[\big(\{x' = v, v' = -g \,\&\, x \geq 0\}; \\ &\qquad (?x = 0; v := -cv \cup ?4 \leq x \leq 5; v := -fv \cup ?x \neq 0)\big)^*\big](0 \leq x \leq 5) \end{aligned} \tag{8.1}$$

Having taken the *Principle of Cartesian Doubt* from Chap. 4 to heart, the aspiring ping-pong ball Quantum first scrutinizes conjecture (8.1) before setting out to prove it. What could go wrong?

For one thing, (8.1) allows the right control options of using the paddle by $?4 \leq x \leq 5; v := -fv$ but it also always allows the wrong choice $?x \neq 0$ when above ground. Remember that nondeterministic choices are just that: nondeterministic! So if Quantum is unlucky, the HP in (8.1) could run so that the middle choice is never chosen and, if the ball has a large downwards velocity v initially, it will jump back up higher than 5 even if it was below 5 initially. That scenario falsifies (8.1). A concrete counterexample can be constructed, e.g., from initial state ω with

$$\omega(x) = 5, \omega(v) = -10^{10}, \omega(c) = \frac{1}{2}, \omega(f) = 1, \omega(g) = 10$$

A less extreme scenario is shown in Fig. 8.3, where the first control at around time 3 works flawlessly but the second event is missed.

Despite this setback in his first control attempt, Quantum is thrilled by the extra prospect of a proper control decision for him to make. So Quantum "only" needs to figure out how to restrict the control decisions such that nondeterminism will only ever take one of the (possibly many) correct control choices, quite a common problem in CPS control. How can Quantum fix this bug in his control and turn

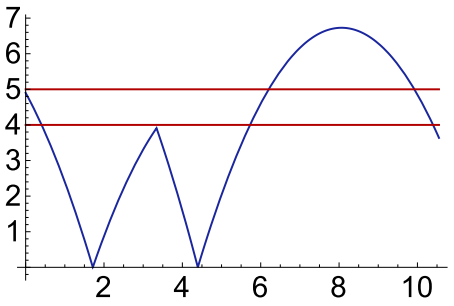


Fig. 8.3 Sample trajectory of a ping-pong ball (plotted as position over time) that misses one event to actuate the ping-pong paddle

himself into a daring ping-pong ball? The problem with the controller in (8.1) is that it permits too many choices, some of which are unsafe. Restricting these choices and making them more deterministic is what it takes to ensure the ping-pong paddle is actuated as intended:

$$0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, x \geq 0\};$$
$$(?x = 0; v := -cv \cup ?4 \leq x \leq 5; v := -fv \cup ?x \neq 0 \wedge x < 4 \vee x > 5)\big)^*\big](0 \leq x \leq 5) \tag{8.2}$$

Recalling the $\mathsf{if}(E)\,\alpha\,\mathsf{else}\,\beta$ statement from Chap. 3, the same system can be modeled equivalently:

$$0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, x \geq 0\};$$
$$(?x = 0; v := -cv \cup ?x \neq 0; \mathsf{if}(4 \leq x \leq 5)\, v := -fv)\big)^*\big](0 \leq x \leq 5)$$

Or, using if-then-else again, as the even shorter equivalent formula

$$0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, x \geq 0\};$$
$$\mathsf{if}(x = 0)\, v := -cv\, \mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5) \tag{8.3}$$

Is conjecture (8.3) valid?

Before you read on, see if you can find the answer for yourself.

8.2.2 Event Detection

The problem with the controller in (8.3) is that, even though it exercises the appropriate control choice whenever the controller runs, the model does not ensure the

controller will ever run at all when needed. The paddle control only runs after the differential equation stops, which can be almost any time. The differential equation is only guaranteed to stop when the ball bounces on the ground ($x = 0$), because its evolution domain constraint $x \geq 0$ would not be satisfied any longer on its way further down. Above ground, the differential equation model does not provide any constraints on how long it might evolve. Recall from Chap. 2 that the semantics of differential equations is nondeterministic in that the system can follow a differential equation *any amount of time*, as long as it does not violate the evolution domain constraints. In particular, the HP in (8.3) might miss the interesting *event* $4 \leq x \leq 5$ that the ping-pong ball's paddle control wanted to respond to. The system might simply skip over that region by following the differential equation $x' = v, v' = -g \,\&\, x \geq 0$ obliviously until the event $4 \leq x \leq 5$ has passed.

How can the HP from (8.3) be modified to make sure that the event $4 \leq x \leq 5$ will always be noticed and never missed?

Before you read on, see if you can find the answer for yourself.

Essentially the only way to prevent the system from following a differential equation for too long is to restrict the evolution domain constraint, which is the predominant way to make cyber and physics interact. Indeed, that is what the evolution domain constraint $\ldots \,\&\, x \geq 0$ in (8.3) did in the first place. Even though this domain was introduced for different reasons (first principle arguments that light balls never fall through solid ground), its secondary effect was to make sure that the ground controller $?x = 0; v := -cv$ will never miss the right time to take action and reverse the direction of the ball from falling to climbing.

Note 40 (Evolution domains detect events) Evolution domain constraints of differential equations in hybrid programs can detect events. That is, they can make sure the system evolution stops whenever an event happens on which the control wants to take action. Without such evolution domain constraints, the controller is not necessarily guaranteed to execute but may miss the event.

Following these thoughts further indicates that the evolution domain somehow ought to be augmented with more constraints that ensure the interesting event $4 \leq x \leq 5$ will never be missed accidentally. How can this be done? Should the event be conjoined to the evolution domain as follows?

$$0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, x \geq 0 \wedge 4 \leq x \leq 5\};$$
$$\mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5)$$

Before you read on, see if you can find the answer for yourself.

Of course not! This evolution domain would be entirely counterfactual and *require* the ball to always be at a height between 4 and 5, which is hardly the right

physical model for any self-respecting bouncing ball. How could the ball ever fall to the ground and bounce back, this way? It couldn't.

Yet, on second thoughts, the way the event $x = 0$ got detected by the HP was not by including $\ldots \& x = 0$ in the evolution domain constraint, either, but by merely including the inclusive limiting constraint $\ldots \& x \geq 0$, which made sure the system could perfectly well evolve before the event domain $x = 0$, but that it just couldn't rush past the event $x = 0$. What would the inclusion of such an inclusive limiting constraint correspond to for the intended ping-pong paddle event $4 \leq x \leq 5$?

When the ball is hurled up into the sky, the last point at which action has to be taken to make sure not to miss the event $4 \leq x \leq 5$ is $x = 5$. The corresponding inclusive limiting constraint $x \leq 5$ thus should be somewhere in the evolution domain constraint. This is in direct analogy to the fact that the rôle of the evolution domain constraint $x \geq 0$ is to guarantee detection of the discrete event $x = 0$ in the discrete action that makes the ball bounce back up.

$$\begin{aligned} &0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow \\ &\quad \big[\big(\{x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5\}; \\ &\qquad \mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5) \end{aligned} \tag{8.4}$$

Is this the right model? Is dL formula (8.4) valid? Will its HP ensure that the critical event $4 \leq x \leq 5$ will not be missed?

Before you read on, see if you can find the answer for yourself.

Formula (8.4) is valid. And, yet, (8.4) *is not at all the appropriate formula to consider!* It is absolutely crucial to understand why.

First, however, note that the HP in (8.4) allows the use of the ping-pong paddle anywhere in the height range $4 \leq x \leq 5$. Its evolution domain constraint enforces that this event $4 \leq x \leq 5$ will be noticed at the latest at height $x = 5$. So when exactly the ping-pong paddle is exercised in that range is nondeterministic (even if the control is written deterministically), because the duration of the differential equation is still chosen nondeterministically. This allows the ping-pong paddle to be controlled at the last height $x = 5$ or before it reaches height $x = 5$ as in Fig. 8.4.

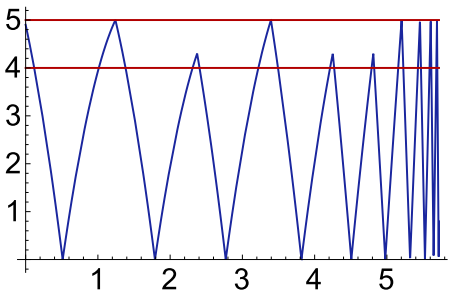


Fig. 8.4 Sample trajectory of a ping-pong ball (plotted as position over time) with the indicated ping-pong paddle actuation range, sometimes actuating early, sometimes late

Notice that (8.4) does not make sure that the critical event $4 \leq x \leq 5$ will not be missed in the case of a ball that is climbing up above the lower trigger 4 but starts falling down again already before it exceeds the upper trigger 5 of the event. Such a possible behavior of the ping-pong ball was already shown in Fig. 8.2. Yet, this is not actually problematic, because missing out on the chance to actuate the ping-pong paddle in a situation where the paddle is not needed to ensure height control is just missing an opportunity for fun, not missing a critical control choice.

But there is a much deeper problem with (8.4). Formula (8.4) is perfectly valid. But why? Because all runs of the differential equation $x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5$ remain within the safety condition $0 \leq x \leq 5$ *by construction*. None of them are ever allowed to leave the region $x \geq 0 \wedge x \leq 5$, which, after all, is their evolution domain constraint. So formula (8.4) is trivially safe, because it says that a system that is constrained to not leave $x \leq 5$ cannot leave $x \leq 5$, which is a pretty trivial insight. A more careful proof involves that, every time around the loop, the postcondition holds trivially, because the differential equation's evolution constraint maintains it by definition, and the subsequent discrete control never changes the only variable x on which the postcondition depends. Hold on, the loop does not have to run but might be skipped over by zero iterations. Yet, in that case, the precondition ensures the postcondition, so, indeed, (8.4) is valid, but only quite trivially so.

Note 41 (Non-negotiability of physics) It is a good idea to make systems safe by construction; but not by changing the laws of physics, because physics is unpleasantly non-negotiable. If the only reason why a CPS model is safe is because we forgot to model all relevant behavior of the real physical system and modeled another universe instead, then correctness statements about those inadequate models do not apply to reality. We do not make this world any safer by writing CPS programs for another universe!
One common cause of counterfactual models is too restrictive evolution domain constraints that rule out physically realistic behavior.

That is what happened in (8.4). Quantum got so carried away with trying not to miss the event $4 \leq x \leq 5$ that he forgot to include a behavior in the model that takes place *after* the event has happened.

Contrast this with the rôle of the evolution domain constraint $\ldots \,\&\, x \geq 0$, which came into the system because of physics: to model the guaranteed bouncing back from the ground and to prevent the ball from falling through the ground. The constraint $x \geq 0$ models physical limitations of balls which cannot fall through solid soil. The evolution domain constraint $\ldots \,\&\, x \leq 5$ got added to the ping-pong HP for an entirely different reason. It came into play to model what our controller does, and inaptly so, because our feeble attempt ruled out physical behavior that could actually have happened in reality. There is no reason to believe that physics would be so kind to only evolve within $x \leq 5$ just because our controller model wants to respond to an event then. Remember never to do that. Ever!

Note 42 (Physical constraints versus control constraints) Some constraints of system models are included for physical reasons; other constraints are added later to describe the controller. Take care to ensure not to accidentally limit the behavior of physics when all you meant to do is impose a constraint on your system controller. Physics will not listen to your desires! This applies to evolution domain constraints but also other aspects of your system model such as tests. It is fine to limit the force that the ping-pong paddle exerts, because that is for the controller to decide. But it is not a good idea for a controller to limit or change the values of gravity or damping coefficients, because that is rather hard to implement without first leaving the planet.

To belabor the point more formally, we could have told directly from a proof of formula (8.4) that its model is broken. Let us use the following abbreviations:

$$A \overset{\text{def}}{\equiv} 0 \leq x \land x \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f \geq 0$$

$$B(x) \overset{\text{def}}{\equiv} 0 \leq x \land x \leq 5$$

$$\{x'' = ..\,\&\, x \geq 0 \land x \leq 5\} \overset{\text{def}}{\equiv} \{x' = v, v' = -g \,\&\, x \geq 0 \land x \leq 5\}$$

$$ctrl \overset{\text{def}}{\equiv} \mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv$$

The proof of formula (8.4) is completely straightforward:

$$\text{loop}\dfrac{\mathbb{R}\dfrac{*}{A \vdash B(x)} \quad [;]\dfrac{\mathbb{V}\dfrac{\mathbb{R}\dfrac{*}{x \geq 0 \land x \leq 5 \vdash B(x)}}{x \geq 0 \land x \leq 5 \vdash [ctrl]B(x)}}{\text{dW}\dfrac{}{B(x) \vdash [\{x'' = ..\,\&\, x \geq 0 \land x \leq 5\}][ctrl]B(x)}}{B(x) \vdash [\{x'' = ..\,\&\, x \geq 0 \land x \leq 5\}; ctrl]B(x)} \quad \mathbb{R}\dfrac{*}{B(x) \vdash B(x)}}{A \vdash [(\{x'' = ..\,\&\, x \geq 0 \land x \leq 5\}; ctrl)^*]B(x)}$$

In addition to the vacuous axiom V from Lemma 5.11 for unmodified postconditions, this sequent proof uses the differential weakening proof rule dW, which will later be explored in full in Lemma 11.2 of Chap. 11, but is already easy to understand right now.[3] The differential weakening proof rule dW proves any postcondition P of a differential equation that is implied by its evolution constraint Q:

$$\text{dW}\ \frac{Q \vdash P}{\Gamma \vdash [x' = f(x)\,\&\, Q]P, \Delta}$$

This is a beautiful and simple proof of (8.4) but there's a catch. Can you spot it?

Before you read on, see if you can find the answer for yourself.

[3] For solvable differential equations, rule dW can, of course, be derived from solution axiom $[']$ by appropriate generalization steps. But rule dW is sound for any other differential equation, which is why it will be explored in Chap. 11 of Part II, which focuses on advanced differential equations.

The above proof of (8.4) worked entirely by the differential weakening rule dW and the vacuous axiom V. The differential weakening rule dW discards the differential equation $\{x' = v, v' = -g\}$ and works entirely from the evolution domain constraint. The vacuous axiom V discards the controller *ctrl* since its postcondition $B(x)$ does not read any of the variables that the HP *ctrl* writes, which is just v.

Well, that is a remarkably efficient proof. But the fact that it *entirely discarded* the differential equation and controller everywhere shows that the property is independent of the differential equation and controller and, thus, holds for any other controller that does not assign to x and any other differential equation (bouncing ball under gravity or not) that shares the same evolution domain constraint $x \geq 0 \wedge x \leq 5$.

Note 43 (Irrelevance) After having constructed a proof, we can go back and check which assumptions, which evolution domain constraints, which differential equations, and which parts of the controller were needed to establish it. This is not just useful to identify crucial versus irrelevant assumptions, but is also insightful to identify which part of a controller or dynamics can be changed without affecting the truth of the property. If almost every aspect of a controller and differential equation turns out to be irrelevant, we should be wary about the model. More generally, the set of facts and expressions on which a proof depends informs us how general or how unique its conclusion is. If a proof was independent of most aspects of a hybrid program, then it states a very broadly applicable general property, but also does not tell us any particularly deep fact that is unique to this specific hybrid program.

Consequently, the fact that the differential equations and controllers were irrelevant for the above proof of (8.4) confirms again that its physics model is broken, because our practical experience clearly demonstrates that safety of the ping-pong ball really depends on a clever use of the ping-pong paddle.

8.2.3 Dividing Up the World

Let's make up for this modeling mishap by developing a model that has both behaviors, the behaviors before and after the event, just in different continuous programs so that the decisive event in the middle cannot accidentally be missed.

$$\begin{aligned} &0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow \\ &\quad \big[\big((\{x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5\} \cup \{x' = v, v' = -g \,\&\, x > 5\}); \\ &\qquad \mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5) \end{aligned} \tag{8.5}$$

Instead of the single differential equation with a single evolution domain constraint in (8.4), the HP in (8.5) has a (nondeterministic) choice between two differential equations, here actually both the same, with two different evolution do-

main constraints. The left continuous system is restricted to the lower physics space $x \geq 0 \wedge x \leq 5$, the right continuous system is restricted to the upper physics space $x > 5$. Every time the loop repeats, there is a choice of either the lower physics equation or the upper physics equation. But the system can never stay in these differential equations for too long, because, e.g., when the ball is below 5 and speeding upwards very fast, then it cannot stay in the left differential equation above height 5, so it will have to stop evolving continuously and give the subsequent controller a chance to inspect the state and respond in case the event $4 \leq x \leq 5$ happened.

Now dL formula (8.5) has a much better model of events than the ill-advised (8.4). Is formula (8.5) valid?

Before you read on, see if you can find the answer for yourself.

The model in (8.5) is, unfortunately, horribly broken. We meant to split the continuous evolution space into the regions before and after the event $4 \leq x \leq 5$. But we overdid it, because the space is now fractured into two disjoint regions, the lower physics space $x \geq 0 \wedge x \leq 5$ and the upper physics space $x > 5$. How can the ping-pong ball ever transition from one to the other? Certainly, as the ball moves upwards within the lower physics space $x \geq 0 \wedge x \leq 5$, it will have to stop evolving at $x = 5$ at the latest. But then even if the loop repeats, the ball still cannot continue in the upper physics space $x > 5$, because it is not quite there yet. Being at $x = 5$, it is an infinitesimal step away from $x > 5$. Of course, Quantum will only ever move continuously along a differential equation. There is no continuous motion that would take the ball from the region $x \geq 0 \wedge x \leq 5$ to the disjoint region $x > 5$ without leaving them. In other words, the HP in (8.5) has accidentally modeled that there will never ever be a transition from lower to upper physics space nor the other way around, because of an infinitesimal gap in between.

Note 44 (Connectedness and disjointness in evolution domains) Evolution domain constraints need to be thought out carefully, because they determine the respective regions within which the system can evolve. Disjoint or unconnected evolution domain constraint regions often indicate that the model will have to be thought over again, because there cannot be any continuous transitions from one domain to the other if they are not connected. Even infinitesimal gaps in domain constraints can cause mathematical curiosities in a model that make it physically unrealistic.

Let's close the infinitesimal gap between $x \geq 0 \wedge x \leq 5$ and $x > 5$ by including the boundary $x = 5$ in both domains:

$$\begin{aligned} &0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow \\ &\quad \big[\big(((x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5) \cup (x' = v, v' = -g \,\&\, x \geq 5)); \\ &\qquad \mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5) \end{aligned} \tag{8.6}$$

Now there is a proper separation into lower physics $x \geq 0 \wedge x \leq 5$ and upper physics $x \geq 5$ but the system can be in either physics space at the switching bound-

ary $x = 5$. This makes it possible for the ball to pass from lower physics into upper physics or back, but only at the boundary $x = 5$, which, in this case, is the only point that the two evolution domain constraints have in common.

In fact, it is generally a good idea to work with overlapping (and often closed) evolution domain constraints to minimize the likelihood of accidentally causing infinitesimal gaps in the domains of the model.

Now dL formula (8.6) has a much better model of events than the ill-advised (8.4). Is formula (8.6) valid?

Before you read on, see if you can find the answer for yourself.

When the ball is jumping up from the ground, the model in (8.6) makes it impossible for the controller to miss the event $4 \leq x \leq 5$, because the only evolution domain constraint in the HP that applies at the ground is $x \geq 0 \wedge x \leq 5$. And that evolution domain stops being true above 5. Yet, suppose the ping-pong ball was flying up from the ground following the continuous program in the left choice and then stopped its evolution at height $x = 4.5$, which always remains perfectly within the evolution domain $x \geq 0 \wedge x \leq 5$ and is, thus, allowed. Then, after the sequential composition between the middle and last line of (8.6), the controller in the last line of (8.6) runs, notices that the formula $4 \leq x \leq 5$ for the event checking is true, and changes the velocity according to $v := -fv$, corresponding to the assumed effect of a pat with the paddle. That is actually its only choice in such a state, because the controller is deterministic, much unlike the differential equation. Consequently, the velocity has just become negative since it was positive before as the ball was climbing up. So the loop can repeat and the differential equation runs again. However, then the differential equation might evolve until the ball is at height $x = 4.25$, which will eventually happen since its velocity stays negative till the ground. If the differential equation stops then, the controller will run again, determine that $4 \leq x \leq 5$ is true still and so take action to change the velocity to $v := -fv$ again. That will, however, make the velocity positive again, since it was previously negative as the ball was in the process of falling. Hence, the ball will keep on climbing now, which, again, threatens the postcondition $0 \leq x \leq 5$. Will this falsify (8.6) or is it valid?

Before you read on, see if you can find the answer for yourself.

On second thoughts, that alone still will not cause the postcondition to evaluate to *false*, because the only way the bouncing ball can evolve continuously from $x = 4.25$ is by the continuous program in the left choice of (8.6). And that differential equation is restricted to the evolution domain $x \geq 0 \wedge x \leq 5$, which causes the controller to run before leaving $x \leq 5$. That is, the event $4 \leq x \leq 5$ will again be noticed by the controller so that the ping-pong paddle pats the ball back down; see Fig. 8.5.

However, exactly the same reasoning applies also to the case where the ball successfully made it up to height $x = 5$, which is the height at which any climbing ball has to stop its continuous evolution, because it would otherwise violate the evolution domain $x \geq 0 \wedge x \leq 5$. As soon as that happens, the controller runs, notices that the event $4 \leq x \leq 5$ is true, and responds with the ping-pong paddle to cause $v := -fv$.

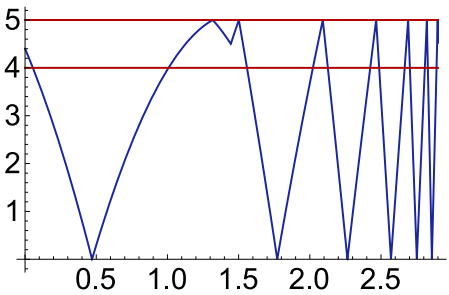


Fig. 8.5 Sample trajectory of a ping-pong ball (plotted as position over time) with the controller firing multiple times for the same event

If, now, the loop repeats, yet the continuous evolution evolves for duration zero only, which is perfectly allowed, then the condition $4 \leq x \leq 5$ will still be true so that the controller again notices this "event" and responds with ping-pong paddle $v := -fv$. That will make the velocity positive, the loop can repeat, the continuous program on the right of the choice can be chosen since $x \geq 5$ holds true, and then the bouncing ball can climb and disappear into nothingness high up in the sky if only its velocity has been large enough. Such a behavior is shown in Fig. 8.6. The second illustration in Fig. 8.6 uses the artistic liberty of delaying the second ping-pong paddle use just a tiny little bit to make it easier to tell the two ping-pong paddle uses apart, even if that is not actually quite allowed by the HP model, because such behavior would actually be reflected by a third ping-pong paddle use as in Fig. 8.5.

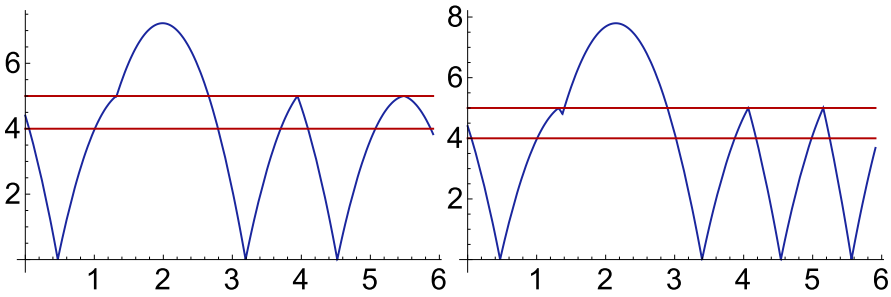


Fig. 8.6 Sample trajectory of a ping-pong ball (plotted as position over time) with the controller firing multiple times for the same event on the event boundary $x = 5$ between lower and upper physics

Ergo, (8.6) is not valid. What a pity! Poor Quantum would still have to be afraid of heights when following the control in (8.6). How can this problem be resolved?

Before you read on, see if you can find the answer for yourself.

8.2.4 Event Firing

The problem in (8.6) is that its left differential equation makes sure never to miss out on the event $4 \leq x \leq 5$ but its control may respond to it multiple times. Should each occasion of $4 \leq x \leq 5$ even be called a separate event? Quite certainly repeated responses to the same event according to control (8.6) cause trouble.

Note 45 (Multi-firing of events) In event-triggered control, exercise care to ensure whether you want events to fire only once when they occur for the first time, or whether the system stays safe even if the same event is detected and responded to multiple times in a row. The latter systems are more robust.

One way of solving this problem is to change the condition in the controller to make sure it only responds to the $4 \leq x \leq 5$ event when the ball is on its way up, i.e., when its velocity is not negative ($v \geq 0$). That is what Quantum had in mind originally, but, in the great tradition of sophisticated systems, neglected to control it appropriately. The ping-pong paddle should only be actuated downwards when the ball is flying up.

These thoughts lead to the following variation:

$$\begin{aligned} &0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow \\ &\big[\big((\{x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5\} \cup \{x' = v, v' = -g \,\&\, x \geq 5\}); \\ &\quad \mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5 \wedge v \geq 0)\, v := -fv\big)^*\big](0 \leq x \leq 5) \end{aligned} \tag{8.7}$$

Because the paddle action $v := -fv$ will disable the condition $v \geq 0$ for nonzero velocities, the controller in (8.7) can only respond once to the event $4 \leq x \leq 5$ to turn the upwards velocity into a downwards velocity, scaled by f (Exercise 8.1). Unlike in (8.6), this control decision cannot be immediately revertes inadvertently by the controller.

Is dL formula (8.7) valid?

Before you read on, see if you can find the answer for yourself.

Yes, formula (8.7) is valid. Finally! Note that it is still the case in (8.7) that, every time around the loop, there will be a nondeterministic choice to evolve within lower physics $x \geq 0 \wedge x \leq 5$ or within upper physics $x \geq 5$. This choice is nondeterministic, so any outcome will be possible. If the left differential equation is chosen, the subsequent continuous evolution must be confined to $x \geq 0 \wedge x \leq 5$ and stop before leaving that lower physics region to give the controller a chance to check for events and respond. If the right differential equation is chosen, the subsequent continuous evolution must be limited to $x \geq 5$ and must stop before leaving that upper physics region to give the controller a chance to inspect. In fact, the only way of leaving the upper physics space is downwards (with velocity $v < 0$), which, unlike in (8.6), will not trigger a response from the subsequent control in (8.7), because that controller checks for $v \geq 0$.

8.2.5 Event-Triggered Verification

How can dL formula (8.7) be proved, so that we have unquestionable evidence that it is, indeed, valid? The most critical element of a proof is finding a suitable invariant. What could be the invariant for proving (8.7)?

Before you read on, see if you can find the answer for yourself.

The postcondition

$$5 \geq x \geq 0 \tag{8.8}$$

is an obvious candidate for an invariant. If it is true, it trivially implies the postcondition $0 \leq x \leq 5$ and it holds in the initial state. It is not inductive, though, because a state that satisfies only (8.8) could follow the second differential equation if it satisfied $x \geq 5$. In that case, if the velocity were positive, the invariant (8.8) would be violated immediately. Hence, at the height $x = 5$, the control has to make sure that the velocity is negative, so that the right differential equation in (8.7) has to stop immediately. Can (8.8) be augmented with a conjunction $v \leq 0$ to form an invariant?

$$5 \geq x \geq 0 \land v \leq 0$$

No, that would not work either, because the bounce on the ground immediately violates that invariant, since the whole point of bouncing is that the velocity will become positive again. In fact, the controller literally only ensures $v \leq 0$ at the event, which is detected at $x = 5$ at the latest that is the safety-critical decision point. Gathering these thoughts, it turns out that the dL formula (8.7) can be proved in the dL calculus using the invariant

$$5 \geq x \geq 0 \land (x = 5 \rightarrow v \leq 0) \tag{8.9}$$

This invariant retains that the possible range of x is safe but is just strong enough to also remember the correct control choice at the event boundary $x = 5$. It expresses that the ball is either in lower physics space or at the boundary of both physics spaces. But if the ball is at the boundary of the physics spaces, then it is moving downwards. Invariant (8.9) follows the general principle of augmenting the expected postcondition with just enough information to guarantee safe control choices at all the critical handovers between the respective modes or decisions.

That is the reason why (8.9) is easily seen to be an invariant of (8.7). The invariant (8.7) is initially true, because the ball is initially in range and moving down. The invariant trivially implies the postcondition, because it consists of the postcondition plus an extra conjunction. The inductive step is most easily seen by considering cases. If the position before the loop body ran was $x < 5$, then the only physics possible to evolve is lower physics, which, by construction, implies the conjunct $5 \geq x \geq 0$ from its evolution domain constraint. The extra conjunct $x = 5 \rightarrow v \leq 0$ is true after the loop body has run, since, should the height actually be 5, which is the only case for which this extra conjunct is not already vacuously true, then the controller made sure to turn the velocity downwards by checking $4 \leq x \leq 5 \land$

$v \geq 0$ and negating the velocity. If the position before the loop body was $x \geq 5$ then the invariant (8.9) implies that the only position it could have had is $x = 5$ in which case either differential equation could be chosen. If the first differential equation is chosen, the reasoning for the induction step is as for the case $x < 5$. If the second differential equation is chosen, then the invariant (8.9) implies that the initial velocity is $v \leq 0$, which implies that the only possible duration that keeps the evolution domain constraint $x \geq 5$ of the upper physics true is duration 0, after which nothing has changed so the invariant still holds.

Observe how the scrutiny of a proof, which necessitated the transition from the broken invariant (8.8) to the provable invariant (8.9), has pointed us to subtleties with events and how ping-pong balls would become unsafe if they fired repeatedly. We discovered these issues by careful formal modeling with our "safety first" approach and a good dose of Cartesian Doubt. But had we not noticed it, the proof would not have let us get away with such oversights, because the (unreflected) invariant candidate (8.8) would not have worked, nor would the broken controller (8.6) have been provable. Of course, having a proof is not a replacement for exercising good judgment over a model to begin with.

Finally, recall that (global) invariants need to be augmented with the usual mundane assumptions about the unchanged variables, like $c \geq 0 \land g > 0 \land f \geq 0$, unless we use the more clever techniques from Sect. 7.5 that automatically preserve assumptions about constant parameters.

8.2.6 Event-Triggered Control Paradigm

The model that (8.7) and the other controllers in this section adhere to is called event-triggered control or sometimes also an event-triggered architecture.

Note 46 (Event-triggered control) One common paradigm for controller design is *event-triggered control*, in which the controller runs in response to certain events that happen in the system. The controller might possibly run under other circumstances as well—when in doubt, the controller simply skips over without any effect if it does not want to change anything about the behavior of the system. But event-triggered controllers assume they will run for sure whenever certain events in the system happen.
These events cannot be all too narrow, or else the system will not be implementable, though. For example, it is nearly impossible to build a controller that responds exactly at the point in time when the height of the bouncing ball is $x = 9.8696$. Chances are high that any particular execution of the system will have missed this particular height. Care must be taken in event-triggered design models also that the events do not inadvertently restrict the evolution of the system for the behavioral cases outside of events or after the events have happened. Those executions must still be verified.

Are we sure in model (8.7) that events are taken into account faithfully? That depends on what exactly we mean by an event like $4 \leq x \leq 5$. Do we mean that this event happens for the first time? Or do we mean every time this event happens? If multiple successive runs of the ping-pong ball's controller see this condition satisfied, do these count as the same or as separate instances of that event happening? Comparing the validity of (8.7) with the non-validity of (8.6) illustrates that these subtleties can have considerable impact on the system. Hence, a precise understanding of events and careful modeling is required.

The controller in (8.7) only takes an action for event $4 \leq x \leq 5$ when the ball is on the way up. Hence, the evolution domain constraint in the right continuous evolution is $x \geq 5$. If we wanted to model the occurrence of event $4 \leq x \leq 5$ also when the ball is on its way down, then we would have to have a differential equation with evolution domain $x \geq 4$ to make sure the system does not miss $4 \leq x \leq 5$ when the ball is on its way down either, without imposing that it would have to notice $x = 5$ already. This can be achieved by splitting the evolution domain regions appropriately, but was not necessary for (8.7) since the controller never responds to balls falling down, only those climbing up.

Note 47 (Subtleties with events) Events are a slippery slope and great care needs to be exercised to use them without introducing an inadequate executional bias into the model.

There is a highly disciplined way of defining, detecting, and responding to general events in differential dynamic logic based on the there and back again axiom of differential dynamic logic [4]. That is, however, much more complicated than the simpler account shown here.

Finally, notice that the proof of (8.7) was almost independent of the differential equation and just a consequence of the careful choice of the evolution domain constraint to reflect the events of interest as well as getting the controller responses to these events right. That is, ultimately, the reason why the invariant (8.9) could be so simple. This also often contributes to making event-triggered controllers easier to get right.

Note 48 (Correct event-triggered control) As long as a controller responds in the right ways to the right events, event-triggered controllers can be built rather systematically and are relatively easy to prove correct. But beware! You have to get the handling of events right, otherwise you only end up with a proof about counterfactual physics, which is not at all helpful since your actual CPS then follows an entirely different kind of physics.

8.2.7 Physics Versus Control Distinctions

Note 49 (Physics versus control) Observe that some parts of hybrid program models represent facts and constraints from physics, and other parts represent controller decisions and choices. It is a good idea to keep the facts straight and remember which part of a hybrid program model comes from which. Especially, whenever a constraint is added because of a controller decision, it is good practice to carefully think through what happens if this is not the case. That is how we ended up splitting physics into different evolution domain constraints, for example.

Partitioning the hybrid program in the verified dL formula (8.7) into the parts that come from physics (typographically marked like physics) and the parts that come from control (typographically marked like control) leads to the following.

Proposition 8.1 (Event-triggered ping-pong is safe). *This* dL *formula is valid:*

$$0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$$
$$\big[\big((\{x' = v, v' = -g \,\&\, x \geq 0 \wedge x \leq 5\} \cup \{x' = v, v' = -g \,\&\, x \geq 5\});\quad (8.7^*)$$
$$\mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5 \wedge v \geq 0)\, v := -fv\big)^*\big](0 \leq x \leq 5)$$

There could have been a second evolution domain constraint $x \geq 0$ for the physics in the second differential equation. But that evolution domain constraint was elided, because it is redundant in the presence of the evolution domain constraint $x \geq 5$ coming from the controller. Only controller constraints have been added compared to the initial physical model of the bouncing ball (7.12) that was entirely physics. This is a good indicator that the design was proper, because it did not alter the physics but merely added controller program parts, including control event detection by splitting differential equations into separate modes.

8.3 Summary

This chapter studied event-triggered control, which is one important principle for designing feedback mechanisms in CPSs and embedded systems. The chapter illustrated the most important aspects for a running example of a ping-pong ball. Even if the impoverished ping-pong ball went vertically and may not be the most exciting application of control in the world, the effects and pitfalls of control events were sufficiently subtle already to merit focusing on a simple intuitive case.

Event-triggered control assumes that all events are detected perfectly and right away. The event-triggered controller in (8.7) took some precautions by defining the event of interest for using the ping-pong paddle to be $4 \leq x \leq 5$. This may look like a big event in space to be noticed in practice, except when the ball moves too quickly, in which case the event $4 \leq x \leq 5$ is over rather quickly. However, the model still

has $x \leq 5$ as a hard limit in the evolution domain constraint to ensure that the event will never be missed in its entirety as the ball is rushing upwards.

Event-triggered control assumes permanent continuous sensing of the event of interest, because the hard limit of the event is ultimately reflected in the evolution domain constraint of the differential equation. This evolution domain constraint is checked permanently according to its semantics (Chap. 3). That gives event-triggered controllers quite simple mathematical models but also often makes them impossible to implement faithfully for lack of continuous-sensing capabilities.

Event-triggered control models can still be useful abstractions of the real world for systems that evolve slowly but sense quickly, because that is close enough to permanent sensing to still detect events quickly enough. Event-triggered control gives bad models for systems that change their state much more quickly than the sensors can catch up. Even in cases where event-triggered controllers are no good match for reality, they can still be helpful stepping stones for the analysis and design of the more realistic time-triggered controllers [1, 2] that the next chapter investigates. If a controller is not even safe when events are detected instantly and perfectly, it will not be safe when events may be discovered only sporadically with certain delay.

Exercises

8.1. Can the ping-pong paddle in (8.7) ever respond to the event $4 \leq x \leq 5$ twice in a row? What would happen if it did?

8.2. Is the ping-pong ball's loop invariant (8.9) also an invariant for just its two differential equations?

8.3. Are any of the following formulas invariants for proving (8.7)?

$$0 \leq x \leq 5 \land (x = 5 \rightarrow v \leq 0) \land (x = 0 \rightarrow v \geq 0)$$
$$0 \leq x < 5 \lor x = 5 \land v \leq 0$$

8.4. Would the invariant (8.9) succeed in proving a variation of (8.7) in which the controller conjunction $\land v \geq 0$ is removed? If so explain why. If not, explain which part of the proof will fail and why.

8.5. Would a generalization of formula (8.7) be valid in which the assumption $v \leq 0$ on the initial state is dropped? If yes, give a proof. If not, show a counterexample and explain how to fix this problem in a way that leads to a generalization of (8.7) that is still a valid formula.

8.6. Could we replace the two differential equations in (8.7) with a single differential equation and a disjunction of their evolution domain constraints to retain a valid formula?

$$0 \leq x \land x \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, (x \geq 0 \land x \leq 5) \lor x \geq 5\};$$
$$\mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq x \leq 5)$$

8.7. Conduct a sequent proof proving the validity of dL formula (8.7). In the sprit of proof irrelevance, carefully track which assumptions are used for which case?

8.8. The hybrid program in (8.4) was an inadequate model of physics because it terminated the world beyond height 5. Model (8.6) fixed this by introducing the same differential equation with the upper physics world and a nondeterministic choice. Would the following model have worked just as well? Would it be valid? Would it be an adequate model?

$$0 \leq x \land x \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f \geq 0 \rightarrow$$
$$\big[\big(\{x' = v, v' = -g \,\&\, x \geq 0 \land (x = 5 \rightarrow v \leq 0)\};$$
$$\mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5)\, v := -fv\big)^*\big](0 \leq x \leq 5)$$

What about the evolution domain constraint $\ldots \,\&\, x \geq 0 \land x \neq 5$ instead?

8.9. What happens if we add an inner loop to (8.7)? Will the formula be valid? Will it be an adequate model of physics?

$$0 \leq x \land x \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f \geq 0 \rightarrow$$
$$\big[\big((\{x' = v, v' = -g \,\&\, x \geq 0 \land x \leq 5\} \cup \{x' = v, v' = -g \,\&\, x \geq 5\})^*;$$
$$\mathsf{if}(x = 0)\, v := -cv \,\mathsf{else}\, \mathsf{if}(4 \leq x \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq x \leq 5)$$

8.10. Modify the event-triggered controller such that its event detection also spots the event $4 \leq x \leq 5$ when descending at the latest at height 4 instead of always at height 5. Make sure this modified controller is safe and find a loop invariant for its proof.

8.11 (*). Design a variation of the event-triggered controller for the ping-pong ball that is allowed to use the ping-pong paddle within height $4 \leq x \leq 5$ but has a relaxed safety condition that accepts $0 \leq x \leq 2 \cdot 5$. Make sure to only force the use of the ping-pong paddle when necessary. Find an invariant and conduct a proof.

8.12 (2D ping-pong events). Design and verify the safety of a ping-pong controller that goes sideways with horizontal motion like in ordinary ping-pong matches.

8.13 (Robot chase). You are in control of a robot tailing another one in hot pursuit on a straight road. You can accelerate ($a := A$) or brake ($a := -b$). But so can the robot you're following! Your job is to design an event-triggered model whose controller makes sure the robots do not crash.

References

[1] Sarah M. Loos. Differential Refinement Logic. PhD thesis. Computer Science Department, School of Computer Science, Carnegie Mellon University, 2016.

[2] Sarah M. Loos and André Platzer. Differential refinement logic. In: *LICS*. Ed. by Martin Grohe, Eric Koskinen, and Natarajan Shankar. New York: ACM, 2016, 505–514. DOI: 10.1145/2933575.2934555.

[3] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.* **41**(2) (2008), 143–189. DOI: 10.1007/s10817-008-9103-8.

[4] André Platzer. The complete proof theory of hybrid systems. In: *LICS*. Los Alamitos: IEEE, 2012, 541–550. DOI: 10.1109/LICS.2012.64.

[5] André Platzer. Differential game logic. *ACM Trans. Comput. Log.* **17**(1) (2015), 1:1–1:51. DOI: 10.1145/2817824.

[6] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.* **59**(2) (2017), 219–265. DOI: 10.1007/s10817-016-9385-1.