15-424: Foundations of Cyber-Physical Systems

Lecture Notes on Events & Delays

André Platzer

Carnegie Mellon University
Lecture 8

1 Introduction

Lecture 3 on Choice & Control demonstrated the importance of control and loops in CPS models, Lecture 5 on Dynamical Systems & Dynamic Axioms presented a way of unwinding loops iteratively to relate repetition to runs of the loop body, Lecture 6 on Truth & Proof showed a corresponding way of unwinding loops in sequent calculus, and Lecture 7 on Control Loops & Invariants finally explained the central proof principle for loops based on induction.

That has been a lot of attention on loops, but there are even more things to be learned about loops. Not by coincidence, because loops are one of the difficult challenges in CPS. The other difficult challenge comes from the differential equations. If the differential equations are simple and there are no loops, CPS suddenly become easy (they are even decidable).

This lecture will focus on how the 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 know since Lecture 2 on Differential Equations & Domains, is fundamentally represented by the evolution domain constraints that determine when physics pauses to let cyber look and act.

Today's lecture focuses on two important paradigms for making cyber interface with physics to form cyber-physical systems, which played an equally important role in classical embedded systems. One paradigm is that of *event-driven architecture*, where reactions to events dominate the behavior of the system. The other paradigm is *time-triggered control*, which use periodic actions to affect the behavior of the system. Both paradigms fall out naturally from an understanding of the hybrid program principle for CPS.

These lecture notes are loosely based on [Pla12b, Pla10].

2 The Need for Control

Having gotten accustomed to the little bouncing ball, this lecture will simply stick to it. Yet, the bouncing ball asks for more action, for it had so far no choice but to wait until it was at height $h = 0$. And when its patience paid off so that it finally observed height $h = 0$, then its only action was to make its velocity bounce back up. Frustrated by this limited menu of actions to choose from, the bouncing ball asks for a ping pong paddle. Thrilled at the opportunities opened up by a ping pong paddle, the bouncing ball first performs some experiments and then settles on using the ping pong paddle high up in the air to push itself back down again. It had high hopes that proper control exerted by the ping pong paddle would allow the ball to go faster without risking the terrified moments inflicted on it by its acrophobic attitude to heights. Setting aside all Münchausian concerns about how effective ping pong paddles can be for the ball if itself is in control of the paddle to be used on itself in light of Newton's third law about opposing forces, let us investigate this situation regardless. After all, it has what it takes to make control interesting: the dynamics of a physical system and decisions on when to react how to the observed status of the system.

Lecture 7 developed a sequent proof of the undamped bouncing ball with repetitions:

$$0 \leq h \wedge h = H \wedge v = 0 \wedge g > 0 \wedge 1 = c \rightarrow$$
$$[\big(h' = v, v' = -g \,\&\, h \geq 0; (?h = 0; v := -cv \cup ?h \geq 0)\big)^*](0 \leq h \wedge h \leq H) \quad (1)$$

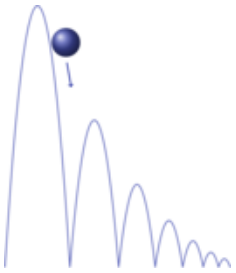

Figure 1: Sample trajectory of a bouncing ball (plotted as position over time)

With this basic understanding of (undamped) bouncing balls, let's examine how to turn the bouncing ball into a ping pong ball using clever actuation of a ping pong paddle. The bouncing ball tried to actuate the ping pong paddle in all kinds of directions. But it never knew where it was going to land if it tried the ping pong paddle sideways. So it quickly gave up the thought of using the ping pong paddle sideways. The ball got so accustomed to its path of going up and down on the spot. With the ping pong paddle, it wanted to do the same, just faster.

By making the ping pong paddle move up and down, the bouncing ball ultimately figured out that the ball would go back down fast as soon as it got a pat on the top by the paddle. It also learned that the other direction turned out to be not just difficult but

also rather dangerous. Moving the ping pong paddle up when the ball was above it to give it a pat on the bottom was tricky, but when it worked would even make the ball fly up higher than before. Yet that is what the acrophobic bouncing ball did not enjoy so much, so it tries to control the ping pong paddle so that the ball only bounces down, never up.

As a height that the bouncing ball feels comfortable with, it chose 5 and so it wants to establish $0 \leq h \leq 5$ to always hold as its safety condition. The ball further puts the ping pong paddle at a similar height so that it can actuate somewhere between 4 and 5. It exercises great care to make sure it would every only move the paddle downwards when the ball is underneath, never above. Thus, the effect of the ping pong paddle will be to reverse the ball's direction. For simplicity, the ball figures that being hit by a ping pong paddle might have a similar effect as being hit by the floor, except with a possibly different factor $f > 0$ instead of the damping coefficient c.[1] So the paddle actuated this way is simply assumed to have effect $v := -fv$.

Taking these thoughts into account, the ball devises the following HP model and conjectures safety expressed in the following $\mathsf{d}\mathcal{L}$ formula:

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(h' = v, v' = -g \,\&\, h \geq 0; \\ &\qquad (?h = 0; v := -cv \cup ?4 \leq h \leq 5; v := -fv \cup ?h \geq 0)\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{2}$$

Having taken the *Principle of Cartesian Doubt* from Lecture 4 on Safety & Contracts to heart, the aspiring ping-pong ball first scrutinizes conjecture (2) before setting out to prove it. What could go wrong?

For one thing, (2) allows the right control options of using the paddle by $?4 \leq h \leq 5; v := -fv$ but also always allows the wrong choice $?h \neq 0$ when above ground. So if the bouncing ball is unlucky, the HP in (2) 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 (2) and a concrete counterexample can be constructed correspondingly, e.g., from initial state ν with

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

How can the bouncing ball bugfix its control and turn itself into a proper ping pong ball? The problem with the controller in (2) is that it permits too much choice, 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:

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(h' = v, v' = -g \,\&\, h \geq 0; \\ &\qquad (?h = 0; v := -cv \cup ?4 \leq h \leq 5; v := -fv \cup ?h \geq 0 \land h < 4 \lor h > 5)\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{3}$$

[1] The real story is quite a bit more complicated, but the bouncing ball does not know any better.

Recalling the $\texttt{if}(E)\,\alpha\,\texttt{else}\,\beta$ statement, the same system can be modeled equivalently:

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

Or, even shorter as the equivalent

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

Is conjecture (4) valid?

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

3 Events in Control

The problem with (4) is that, even though it exercises the right control choice whenever the controller runs, the model does not ensure the controller would run at all when needed. The paddle control only runs after the differential equation stops. That is guaranteed to happen when the ball bounces down to the ground ($h = 0$) but could otherwise be any time. Recall from Lecture 2 that the semantics of differential equations is nondeterministic. 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 (4) could miss the *event* $4 \leq h \leq 5$ that the ping pong ball's paddle control wanted to react to. The system might simply skip over that region by following the differential equation $h' = v, v' = -g \,\&\, h \geq 0$ obliviously.

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

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

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 physical interact. Indeed, that is what the evolution domain constraint $\ldots \& h \geq 0$ in (4) 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 $?h = 0; v := -cv$ will never miss the right time to take action and reverse the direction of the ball from falling to climbing.

Note 1 (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 $4 \leq h \leq 5$ will never be missed accidentally. How can this be done? Should the event be conjoined to the evolution domain as follows

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

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

Of course not! This evolution domain would require the ball to always be at height between 4 and 5, which is hardly the right model. How could the ball ever fall on the ground and bounce back, this way? It couldn't.

Yet, on second thought, the way the event $\ldots \& h = 0$ got detected by the HP in the first place was not by including $h = 0$ in the evolution domain constraint, but by including the inclusive limiting constraint $\ldots \& h \geq 0$, which made sure the system could perfectly well evolve outside this event domain $h = 0$, but that it couldn't just miss the event rushing past $h = 0$. What would the inclusion of such an inclusivelimiting constraint correspond to for the event $4 \leq h \leq 5$?

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 $4 \leq h \leq 5$ is $h = 5$. The corresponding inclusive limiting constraint $h \leq 5$ thus should be somewhere in the evolution domain constraint.

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

Is this the right model? Is $\mathsf{d}\mathcal{L}$ formula (5) valid? Will its HP ensure that the critical event $4 \leq h \leq 5$ will not be missed out on?

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

Formula (5) is valid. And, yet, (5) is not at all the appropriate formula to consider. It is crucial to understand why.

So, formula (5) is valid. But why? Because all runs of the differential equation $h' = v, v' = -g \,\&\, h \geq 0 \wedge h \leq 5$ remain within the safety condition $0 \leq h \leq 5$ *by construction*. None of them are ever allowed to leave the region $h \geq 0 \wedge h \leq 5$, which, after all, is their evolution domain constraint. So formula (5) is trivially safe. A more careful argument involves that, every time around the loop, the postcondition holds trivially, because the differential equation's evolution constraint maintains it by definition, the subsequent discrete control never changes the only variable h on which the postcondition depends. Hold on, the loop does not have to run but could be skipped over by zero iterations as well. Yet, in that case, the precondition ensures the postcondition, so, indeed, (5) is valid, but trivially so.

Note 2 (Non-negotiability of Physics)**.** *Usually, it is a good idea to make systems safe by construction. For computer programs, that is a great idea. But we need to remember that physics is unpleasantly non-negotiable. So if the only reason why a CPS model is safe is because we forgot to model all relevant behavior of the real system, then correctness statements about those inadequate models are not particularly applicable to reality.*

One common cause for counterfactual models are too generous evolution domain constraints that rule out physically realistic behavior.

And that is what happened in (5). The bouncing ball got so carried away with trying not to miss the event $4 \leq h \leq 5$ that it forgot to include a behavior in the model that happens after the event has happened. The evolution domain constraint $\ldots \,\&\, h \geq 0$ came was in the system for physical reasons: to model the guaranteed bouncing back on the ground and to prevent the ball from falling through the ground. We added the evolution domain constraint $h \leq 5$ 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.

Let's make up for this by developing a model that has both behaviors, just in different continuous programs so that the decisive event in the middle could not accidentally have been missed.

$$\begin{aligned} 0 \leq h \wedge h \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f > 0 \rightarrow \\ \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \wedge h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{6}$$

Now (6) has a much better model of events than the ill-advised (5). Is (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 (6) makes it impossible for the controller to miss the event $4 \leq h \leq 5$, because the only evolution domain constraint in the HP that applies at the ground is $h \geq 0 \wedge h \leq 5$. And that evolution domain stops being true above 5. Yet, suppose the ping pong ball was jumping up from the ground following the continuous program in the left choice and then stopped its evolution at height $h = 4.5$, which always remains perfectly within the evolution domain $h \geq 0 \wedge h \leq 5$ and is, thus, allowed. Then, after the sequential composition between the middle and last line of (6), the controller in the last line of (6) runs, notices that the formula $4 \leq h \leq 5$ 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. Yet, then the differential equation might evolve until the ball is at height $h = 4.25$, which will happen since its velocity is negative. If the differential equation stops then, the controller will run again, determine that $4 \leq h \leq 5$ is true still and so take action to change the velocity to $v := -fv$. 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 $0 \leq h \leq 5$. Will this falsify (6) or is it valid?

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

On second thought, that alone still will not cause the postcondition to evaluate to $false$, because the only way the bouncing ball can evolve continuously from $h = 4.25$ is still by the continuous program in the left choice of (6). And that differential equation is restricted to the evolution domain $h \geq 0 \wedge h \leq 5$, which causes the controller to run before leaving $h \leq 5$. That is, the event $4 \leq h \leq 5$ will again be noticed by the controller so that the ball is ping pong paddle pats the ball back down.

However, the exact same reasoning applies also to the case where the ball successfully made it up to height $h = 5$, which is the height at which any climbing ball has to stop its continuous evolution, because it would otherwise violate the evolution domain $h \geq 0 \wedge h \leq 5$. As soon as that happens, the controller runs, notices that the event $4 \leq h \leq 5$ came true and reacts with a ping pong paddle to cause $v := -fv$. If, now, the loop repeats, yet the continuous evolution evolves for duration zero only, which is perfectly allowed, then the condition $4 \leq h \leq 5$ will still be true so that the controller again notices this "event" and reacts 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 $h \geq 5$ 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.

Ergo, (6) is not valid. What a pity. And the bouncing ball would have to be afraid of heights when following the control in (6). How can this problem be resolved?

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

The problem in (6) is that its left differential equation makes sure never to miss out on the event $4 \leq h \leq 5$ but its control may react to it multiple times. It is not even sure whether each occasion of $4 \leq h \leq 5$ should be called an event. But certainly repeated reaction to the same event according to control (6) causes trouble.

One way of solving this problem is to change the condition in the controller to make sure it only reacts to the $4 \leq h \leq 5$ event when the ball is on its way up, i.e. when its velocity is not negative. That is what the bouncing ball wanted to ensure in any case. The ping pong paddle should only be actuated downwards when the ball is flying up.

These thoughts lead to the following variation:

$$\begin{aligned} &0 \leq h \wedge h \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f > 0 \rightarrow \\ &\quad \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \wedge h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ &\qquad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5 \wedge v \geq 0)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{7}$$

Because the paddle action $v := -fv$ will disable the condition $v \geq 0$ for nonzero velocities, the controller in (7) can only react once to the event $4 \leq h \leq 5$ to turn the upwards velocity into a downwards velocity, scaled by f. Unlike in (6), this control decision cannot be reverted inadvertently by the controller.

Is dℒ formula (7) valid?

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

In order to convince ourselves that the ping pong paddle control works as expected, we simplify the assumptions in formula (7) so that they match the ones in our prior proofs about bouncing balls in Lecture 7 on Control Loops & Invariants. Those additional assumptions are not all strictly necessary, but simplify the argument somewhat.

$$\begin{aligned}0 \leq h \leq 5 \land v \leq 0 \land 1 \geq c \geq 0 \land g > 0 \land f \geq 0 \rightarrow \\ \big[\big(((h' = v, v' = -g \,\&\, h \geq 0 \land h \leq 5) \cup (h' = v, v' = -g \,\&\, h \geq 5)); \\ \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5 \land v \geq 0)\, v := -fv\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{8}$$

How could dℒ formula (8) be proved? The most critical element of a proof is finding a suitable invariant. What could be the invariant for proving (8)?

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

The formula

$$5 \geq h \geq 0 \tag{9}$$

is an obvious candidate for an invariant. If it is true, it trivially implies the postcondition $0 \leq h \leq 5$ and it holds in the initial state. It is not inductive, though, because a state that satisfies (9) could follow the right differential equation if it satisfies $h \geq 5$. In that case, if the velocity is positive, the invariant (9) would be violated immediately. Hence, at the height $h = 5$, the control has to make sure that the velocity is negative, so that the right differential equation in (8) has to stop immediately. Could (9) be augmented with a conjunction $v \leq 0$? No that would not work either, because the bounce on the ground violates that invariant. In fact, the controller literally only ensures $v \leq 0$ at the event, which is detected at $h = 5$ at the latest. Indeed, the dL formula (7) can be proved in the dL calculus using the invariant

$$5 \geq h \geq 0 \land (h = 5 \rightarrow v \leq 0)$$

This invariant is just strong enough to remember the control choice at the event $h = 5$ and that the possible range of h is safe. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $c \geq 0 \land g > 0 \land f \geq 0$.

The model that (8) and the other controllers in this section adhere to is called event-driven control or also event-driven architecture.

Note 3 (Event-driven control)**.** *One common paradigm for designing controllers is the event-driven architecture, in which the controller runs in response to certain events that happen in the system. The controller could 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-driven 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 reacts exactly at the point in time when the height of the bouncing ball is $h = 4.12345$. Chances are high that any particular execution of the system will have missed this particular height. Care must be taken in event-driven design models also that the events do not inadvertently restrict the evolution of the system to the behavioral cases outside or after the events have happened. Those executions must still be verified.

Are we sure in model (8) that events are taken into account faithfully? That depends on what exactly we mean by an event like $4 \leq h \leq 5$. 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 separate instances of that event happening? Comparing the validity of (6) with the non-validity of (7) 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) only takes an action for event $4 \leq h \leq 5$ when the ball is on the way up. Hence, the evolution domain constraint in the right continuous evolution is $h \geq 5$. Had we wanted to model the occurrence of event $4 \leq h \leq 5$ also when the ball is on its way down, then we would have to have a differential equation with evolution domain $h \geq 4$ to make sure the system does not miss $4 \leq h \leq 5$ when the ball is on its way down either, without imposing that it would have to notice $h = 5$ already. This could be achieved by splitting the evolution domain regions appropriately, but was not necessary for (8) since it never reacts to balls falling down, only those climbing up.

Note 4. *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 reacting to general events in differential dynamic logic based on the there and back again axiom [Pla12a]. That is, however, much more complicated than the simpler account shown here.

4 Delays in Control

Event-driven control is a useful and intuitive model matching our expectation of having controllers react in response to certain critical conditions or events that necessitate intervention by the controller. Yet, one of its difficulties is that event-driven control can be hard or impossible to implement in reality. On a higher level of abstraction, it is very intuitive to design controllers that react to certain events and change the control actuation in response to what events have happened. Closer to the implementation, this turns out to be difficult, because actual computer control algorithms do not actually run all the time, only sporadically every once in a while, albeit sometimes very often. Implementing event-driven control faithfully would, in principle, require permanent continuous monitoring of the state to check whether an event has happened. That is not quite realistic.

Back to the drawing desk. Let us reconsider the original dℒ formula (4) that we started out from for designing the event-driven version in (8).

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

This simplistic formula (4) turned out not to be valid, because its differential equation was not guaranteed to be interrupted when the event $4 \leq h \leq 5$ happens. Consequently, (4) needs some other evolution domain constraint to make sure all continuous evolutions are stopped at some point for the control to have a chance to react to situation changes. Yet, it should not be something like $\ldots \& h \leq 5$ as in (8), because

continuously monitoring for $h \leq 5$ requires permanent sensing of the height, which is difficult to implement.

How else could the continuous evolution of physics be interrupted to make sure the controller runs? By bounding the amount of time that physics is allowed to evolve before running the controller again. Before we can talk about time, the model needs to be changed to include a variable, say t, that reflects the progress of time with a differential equation $t' = 1$.

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1; \\ &\qquad \texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5)\, v := -fv\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{10}$$

In order to bound time by 1, the evolution domain now includes $\ldots \,\&\, t \leq 1$ and the variable t is reset to 0 by $t := 0$ right before the differential equation. Hence, t represents a local clock measuring how long the evolution of the differential equation was. Its bound of 1 ensures that physics gives the controller a chance to react at least once per second. The system could very well stop the continuous evolution more often and earlier, because there is no lower bound on t in (10). Also see Exercise 1.

Before going any further, let's take a step back to notice an annoyance in the way the control in (10) was written. It is written in the style that the original bouncing ball and the event-driven ping pong ball were phrased: continuous dynamics followed by control. That has the unfortunate effect that (10) lets physics happen before control does anything, which is not a very safe start. In other words, the initial condition would have to be modified to assume the initial control was fine. That is a nuisance duplicating part of the control into the assumptions on the initial state. Instead, let's switch the statements around to make sure control always happens before physics.

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(\texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}(4 \leq h \leq 5)\, v := -fv; \\ &\qquad t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{11}$$

Now that dL formula (11) has an upper bound on the time it takes between two subsequent control actions, is it valid?

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

Even though (11) ensures a bound on how long it may take at most until the controller inspects the state and reacts, there is still a fundamental issue with (11). We can try to prove (11) and inspect the non-provable cases in the proof to find out what the issue is. The controller of (11) runs at least after one second (hence at least once per second) and then checks whether $4 \leq h \leq 5$. But if $4 \leq h \leq 5$ was not true when the controller ran last, there is no guarantee that it will be true when the controller runs next. In fact, the ball might very well have been at $h = 3$ at the last controller run, then evolved continuously to $h = 6$ within a second and so missed the event $4 \leq h \leq 5$ that it was supposed to detect (Exercise 2). Worse than that, the ping pong ball has then already become unsafe.

For illustration, driving a car would be similarly unsafe if you would only open your eyes once a second and monitor whether there is a car right in front of you. Too many things could have happened in between that should have prompted you to brake.

Note 5 (Delays may miss events)**.** *Delays in controller reactions may cause events to be missed that they were supposed to monitor. When that happens, there is a discrepancy between an event-driven understanding of a CPS and the real time-triggered implementation. That happens especially for slow controllers monitoring small regions of a fast moving system. This relationship deserves special attention to make sure the impact of delays on a system controller cannot make it unsafe.*

It is often a good idea to first understand and verify an event-driven design of a CPS controller and then refine it to a time-triggered controller to analyze and verify that CPS in light of its reaction time. Discrepancies in this analysis hint at problems that event-driven designs will likely experience at runtime and they indicate a poor event abstraction.

How can this problem of (11) be solved? How can the CPS model make sure the controller does not miss its time to take action? Waiting until $4 \leq h \leq 5$ holds true is not guaranteed to be the right course of action for the controller.

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

The problem with (11) is that its controller is unaware of its own delay. It does not take into account how the ping pong ball could have moved further before it gets a chance to react next. If the ball is already close to the ping pong paddle's intended range of actuation, then the controller had better take action already if it is not sure whether next time will still be fine.

The controller would be in trouble if, in its next control cycle after the continuous evolution, $h > 5$. The continuous evolution can take at most 1 time unit, after which the ball will be at position $h + v - \frac{g}{2}$ as we have observed in Lecture 4 by solving the differential equation. We chose $g = 1$ for the time-triggered case, so the controller could be in trouble in the next control cycle if $h > 5\frac{1}{2} - v$ holds now. Hence, the idea is to make the controller now act based on how it estimates the state might have evolved until the next control cycle. The difference of (6) vs. (7) in the event-driven case indicates that the controller only wants to trigger action if the ball is flying up. Thus, making (11) aware of the future in this way leads to:

$$\begin{aligned} &0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 \geq c \geq 0 \land f > 0 \rightarrow \\ &\quad \big[\big(\texttt{if}(h = 0)\, v := -cv \,\texttt{else if}((h > 5\frac{1}{2} - v) \land v \geq 0)\, v := -fv; \\ &\qquad t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{12}$$

Is conjecture (12) about its future-aware controller valid?

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

The controller in formula (12) has been designed based on the prediction that the future may evolve for 1 time unit. If action will no longer be possible in 1 time unit, because the event $h \leq 5$ has passed in that future instant, the controller in (12) takes action right now already. The issue with that is there is no guarantee that the ping pong ball will fly for exactly 1 time unit before the controller is asked to act again (and the postcondition is checked). The controller in (12) checks whether the ping pong ball could be too far up after one time unit and does not intervene unless that is the case. Yet, what if the ball flies for $\frac{1}{2}$ time units? Clearly, if the ball will be safe after 1 time unit, which is what the controller in (12) checks, it will also be save after just $\frac{1}{2}$ time unit, right?.

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

Wrong! The ball may well be below 5 after 1 time unit but still could have been above 5 in between the current point of time and 1 time unit from now. Recall Fig. 1 to see how this can happen.

In order to understand this further, we use the invariant that we have derived for the bouncing ball in an earlier lecture and then used in Lecture 7 to prove (1).

$$2gh = 2gH - v^2 \land h \geq 0 \land c = 1 \land g > 0 \tag{13}$$

We assume this invariant to hold in the beginning of the ping pong ball's life and also adopt the global assumptions $c = 1 \land g = 1 \land f = 1$ to simplify the arithmetic.

Substituting the critical height 5 for H in (13) for this instance of parameter choices leads to the following condition which indicates that the ball could end up climbing too high

$$2h > 2 \cdot 5 - v^2 \tag{14}$$

Adding this condition to the controller (12) leads to:

$$\begin{aligned} 2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \rightarrow \\ \big[\big(\texttt{if}(h = 0)\, v := -cv \,\texttt{else}\, \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2) \land v \geq 0)\, v := -fv; \\ t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5) \end{aligned} \tag{15}$$

Is dℒ formula (15) about its time-triggered controller valid?

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

Formula (15) is almost valid. But it is still not valid for a very subtle reason. It is great to have proof to catch those subtle issues. The controller in (15) takes action for two different conditions on the height h. However, the ping pong paddle controller actually only runs in (15) if the ball is not at height $h = 0$, for otherwise ground control takes action of reversing the direction of the ball. Now, if the ball is flat on the floor ($h = 0$) yet its velocity so incredibly high that it will rush past height 5 in less than 1 time unit, then the ping pong paddle controller will not have had a chance to react before it is too late, because it does not run on the ground according to (15).

Fortunately, these thoughts already indicate how that problem can be fixed. By turning the nested `if-then-else` cascade into a sequential compositions of `if-then` that will ensure the ping pong paddle controller to run for sure.

$$2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \rightarrow$$
$$\big[\big(\texttt{if}(h = 0)\, v := -cv; \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2) \land v \geq 0)\, v := -fv; \tag{16}$$
$$t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)$$

Now, is formula (16) finally valid, please?

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

Yes, formula (16) is valid and can be proved with the invariant

$$2h = 2H - v^2 \land h \geq 0 \land h \leq 5 \tag{17}$$

Yet, is the controller in (16) useful? That is where the problem lies now. The condition (14) checks whether the ping pong ball could possibly ever fly up to height 5. If this is ever true, it might be true long before the bouncing ball approaches the critical control cycle where ping pong paddle action needs to be taken. In fact, if (14) is ever true, it will also be true in the beginning. After all, the formula (13), from which (14) derived, is an invariant. That would cause the controller in (16) to take action right away even if the ping pong ball is still close to the ground and far away from height 5. That would make the ping pong ball safe, after all (16) is valid, but also rather conservative, and would not allow the ping pong ball to bounce around as much as it would have loved to. How can the controller in (16) be modified to resolve this problem?

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

Restrict the use of condition Exercise 1 to slow velocities to only make up for the occasions that the first controller condition $h > 5\frac{1}{2} - v$ misses. Only with slow velocities ($v < 1$) does the ball move so slowly that it is near its turning point to start falling down, and only then could the first condition miss out on the ball being able to have evolve above 5 before 1 time unit.

$$\begin{aligned}2h = 2H - v^2 \land 0 \leq h \land h \leq 5 \land v \leq 0 \land g = 1 > 0 \land 1 = c \geq 0 \land 1 = f > 0 \rightarrow \\ \big[\big(\texttt{if}(h = 0)\, v := -cv; \texttt{if}((h > 5\frac{1}{2} - v \lor 2h > 2 \cdot 5 - v^2 \land v < 1) \land v \geq 0)\, v := -fv; \\ t := 0; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0 \land t \leq 1\big)^*\big](0 \leq h \leq 5)\end{aligned} \tag{18}$$

This $\mathsf{d}\mathcal{L}$ formula is valid and provable with the same invariant (17) that was used to prove (16). It has a much more aggressive controller than (16), though, so it is more fun for the ping pong ball to bounce around with it. Recall that (global) invariants need to be augmented with the usual assumptions about the unchanged variables, like $g = 1 \land 1 = c \land 1 = f$.

Note 6 (Time-triggered control)**.** *One common paradigm for designing controllers is time-triggered control, in which controllers run periodically or pseudo-periodically with certain frequencies to inspect the state of the system. Time-triggered systems are closer to implementation than event-driven control. They can be harder to build, however, because they invariably require the designer to understand the impact of delay on control decisions. That impact is important in reality, however, and, thus, effort invested in understanding the impact of time delays usually pays off in designing a safer system that is robust to bounded time delays.*

Exercises

Exercise 1. The HP in (11) imposes an upper bound on the duration of a continuous evolution. How can you impose an upper bound 1 and a lower bound 0.5?

Exercise 2. Give an initial state for which the controller in (11) would skip over the event without noticing it.

Exercise 3. The formula (18) with the time-triggered controller of reaction time at most 1 time unit is valid. Yet, if a ball is let loose a wee bit above ground with a very fast negative velocity, couldn't it possibly bounce back and exceed the safe height 5 faster than the reaction time of 1 time unit? Does that mean the formula ought to have been falsifiable? No! Identify why and give a physical interpretation.

Exercise 4. The event-driven controller we designed in Sect. 3 monitored the event $4 \leq h \leq 5$. The time-triggered controller in Sect. 4, however, ultimately only took the upper bound 5 into account. How and under which circumstances can you modify the controller so that it really only reacts for the event $4 \leq h \leq 5$.

Exercise 5. Devise a controller that reacts if the height changes by 1 when comparing the height before the continuous evolution to the height after. Can you make it safe? Can you implement it? Is it an event-driven or a time-triggered controller? How does it compare to the controllers developed in this lecture?

References

[Pla10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14509-4.

[Pla12a] André Platzer. The complete proof theory of hybrid systems. In *LICS*, pages 541–550. IEEE, 2012. doi:10.1109/LICS.2012.64.

[Pla12b] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012. arXiv:1205.4788.