To appear, ACM Tr. on Cyber-Physical Systems, 2016.

A

Fundamental Limits of Cyber-Physical Systems Modeling[1]

EDWARD A. LEE, EECS Department, UC Berkeley

This paper examines the role of modeling in the engineering of cyber-physical systems. It argues that the role that models play in engineering is different from the role they play in science, and that this difference should direct us to use a different class of models, where simplicity and clarity of semantics dominate over accuracy and detail. I argue that determinism in models that are used for engineering is a valuable property and should be preserved whenever possible, regardless of whether the system being modeled is deterministic. I then identify three classes of fundamental limits on modeling, specifically chaotic behavior, the inability of computers to numerically handle a continuum, and the incompleteness of determinism. The last of these has profound consequences.

1. MODELING

Modeling is central to every scientific and engineering enterprise. Golomb, who has written eloquently about the use of models in science and engineering, emphasizes understanding the distinction between a model and thing being modeled. He famously stated "you will never strike oil by drilling through the map" [Golomb 1971].

For both scientists and engineers, the "thing being modeled" is typically an object, process, or system in the physical world. But it could also be another model. Let us call the thing being modeled the **target** of the model. The **fidelity** of a model is the degree to which it emulates the target.

When the target is a physical object, process, or system, then model fidelity is never perfect. Box and Draper state, "essentially, all models are wrong, but some are useful" [Box and Draper 1987]. In science, the value of a model lies in how well its properties match those of the target. In *engineering*, however, the value of the *target* lies in how well its properties match those of the model. A scientist constructs models in order to help understand the target. An engineer constructs *targets* to emulate the properties of a model. For an engineer, a model provides a **design** and the target is the **implementation**.

These two uses of models are complementary. An engineer will typically use models *both* ways. Good engineering requires doing some science. And at least for experimental science, good science requires doing some engineering.

Although model fidelity is rarely perfect, some models are astonishingly good. A synchronous digital circuit, which is a collection of gates and latches that are clocked, is a model of physical system, a three-dimensional structure of doped semiconductors with an applied voltage. The modeling paradigm of synchronous digital circuits is simple, rooted in boolean logic, and (often, but not always) **deterministic**; if you know the initial state of the circuit and you know the inputs, then there is exactly one behavior specified by the model. Humans have learned to build targets that match the behavior of such models extremely well. We can build a chip that will perform an operation specified by a synchronous digital logic model a billion times per second, and the chip will very likely function without error for years. No other human-engineered artifact has ever reached this level of model fidelity.

A given target may have many useful models. For example, a microprocessor chip may be modeled as a three-dimensional geometry of doped silicon (model A); as differential

[1]This paper reflects several years of work under several sponsored projects, including: the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA; the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies); the National Science Foundation (NSF) awards #1446619 (Mathematical Theory of CPS), #1329759 (COSMOI), #0931843 (ActionWebs), #0720882 (CSR-EHS: PRET), #1035672 (CPS: Medium: Timing Centric Software); and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, supported by the following companies: Denso, IHI, National Instruments, and Toyota.

equations that describe the semiconductor physics (model B); as a synchronous digital circuit (model C); or as a computer program (model D) specifying embedded software for the chip to run. These models are all abstractions of the chip, and they serve very different purposes. Which model to use depends on the goal.

Every model is constructed within some **modeling framework** that provides the **syntax** by which the model is specified (how it is written down or otherwise rendered in physical form) and the **semantics** (what a given rendition means). For example, model A might be given in a language for describing 3D shapes. Model B is given in the framework of the calculus of ordinary and partial differential equations. Model C is given in a hardware description language, such as VHDL. Model D is given in a programming language.

The choice of modeling framework has profound consequences. For example, the language for describing 3D shapes (model A) is not well suited for modeling the **dynamics** of a circuit (how the voltages and currents change over time).

The properties of a modeling framework can also strongly affect the value of a model in that framework. A framework intended for expressing dynamics, for example, may be deterministic or not. In a deterministic framework, if the initial state of the model is known, and if the inputs to the model are known, then the model specifies exactly one behavior. In a nondeterministic framework, the model specifies a family of behaviors.

Note that the role of *inputs* in a model overlaps with the role of nondeterminism. Models that have inputs define a family of behaviors. A deterministic model defines one behavior for each possible input. Inputs are provided by the environment; they are not defined by the model. Inputs can equally well be modeled as nondeterminism, but doing so reduces our ability to design a control system, which observes outputs in order to provide inputs that drive towards some desired behavior.

Note that determinism is a property of a *model*, and not a property of the thing being modeled (unless that thing is also a model).[2] Indeed, asserting determinism in the physical world would put us in the middle of a centuries-old debate in philosophy and physics, and it would probably require us to reject much of modern physics.

Determinism *in models*, however, has proved practical and extremely valuable in the past. Ordinary and partial differential equations, for example, are deterministic, and mechanical, electrical, and civil engineers rely heavily on that determinism. They use these equations to develop controllers, to prove stability, and to analyze robustness to parameter variations. Synchronous digital circuit models are also (often) deterministic, as are single-threaded imperative programs. This determinism is hugely valuable, as it enables very complex and yet reliable designs by enabling definitive analysis. We can make absolute assertions about the behavior of a deterministic model. No such assertions are possible about the physical world (how will the synchronous digital circuit behave while it's being crushed?). The *targets* of deterministic models, however, can only be deterministic if the target itself is a model. The lack of determinism in the physical world does not undermine the value of deterministic models as long as the model fidelity is high.

The determinism of a model depends on what is considered to be an *output* from the model. For example, a terminating single-threaded imperative program is deterministic if the output is defined to be the final state of the machine. However, if we also consider the *time* at which that final state is reached to be an output, then the model is nondeterministic.

Choosing to use deterministic models can have a cost. For example, by choosing to use single-threaded imperative programs, we forgo the ability to exploit the parallelism in a multicore machine. However, this cost is not a consequence of choosing *determinism*, it is a consequence of choosing this *particular* deterministic modeling framework. If we were to choose instead to build our program using Kahn process networks [Kahn and MacQueen

[2]Here, I disagree with the position taken by Furia et al., who state "the notions of determinism and nondeterminism are attributes of the systems being modeled or analyzed" [Furia et al. 2010].

1977] or any of several other deterministic actor modeling frameworks, we would still be working with deterministic models, and the execution of the software would have no difficulty exploiting multicore hardware.

Some problems may seem to *require* nondeterminism as a basic part of the problem statement. For example, if you are building a web server, requests will come in at random times and may be very bursty. Implementing the web server as a single-threaded imperative program is probably not a good choice because every response to a request will be blocked by the handling of all previous arrived requests. But again, we do not have to forgo determinism to handle these requests efficiently. We just have to forgo single-threaded imperative programs. All modern server-side software infrastructure includes support for simultaneous dispatch of multiple independent and mutually isolated request handlers. If the output of the server is defined to be the *set* of responses issued for a *set* of input requests, then it is easy to provide a deterministic model of the server. If the output is defined instead to be a *sequence* of responses, then an efficient server will *cannot* be modeled deterministically. The order in which responses are issued is not a consequence of the inputs. If the design goal is to issue predictable responses to requests, then a properly construed deterministic model is in fact valuable. For example, it can be used to specify regression tests.

1.1. Modeling Cyber-Physical Systems

A **cyber-physical system** (CPS) combines cyber systems (computational systems such as microprocessors and digital communication networks) with other physical systems (electromechanical, chemical, structural, and biological systems). Each of these components have benefited from deterministic modeling frameworks, such as single-threaded imperative programs and discrete-event models for the cyber side, and ordinary and partial differential equations for the physical side. But as I have argued before in [Lee 2015], all widely used combinations of deterministic cyber models and deterministic physical models are nondeterministic. In the same paper I argued that this need not be so; deterministic models for CPS are practical. Given the evident value of determinism in modeling, deterministic frameworks should be used.

In this paper, I reflect on lessons learned from building a deterministic modeling framework for cyber-physical systems called **CyPhySim** [Lee et al. 2015]. CyPhySim is an open-source simulator for CPS based on Ptolemy II [Ptolemaeus 2014]. CyPhySim supports mixed continuous and discrete dynamics using a superdense model of time [Lee and Zheng 2007], modal models and hybrid systems [Lee and Tripakis 2010], and an ability to import functional mockup units (FMUs) [Modelica Association 2014]. CyPhySim integrates five simulation engines:

(1) discrete event simulation (DE) [Lee 1999]
(2) quantized-state solvers (QSS) [Cellier and Kofman 2006]
(3) Runge-Kutta solvers (RK2-3, RK4-5) [Cellier and Kofman 2006]
(4) algebraic loop solvers [Lee et al. 2015]
(5) state machines [Feng et al. 2014]

This combination offers a rich set of modeling choices, and yet, some systems, even simple systems, prove difficult to model in useful ways. This paper studies some of these difficulties with a particular emphasis on identifying the *fundamental* limits of modeling. I focus on three classes of fundamental limits:

(1) chaotic behavior,
(2) the inability of computers to numerically handle a continuum, and
(3) the incompleteness of determinism.

The first two of these are well known phenomena, so I focus on the implications they have for CPS. The third, however, appears to be a relatively new observation, first made in [Lee

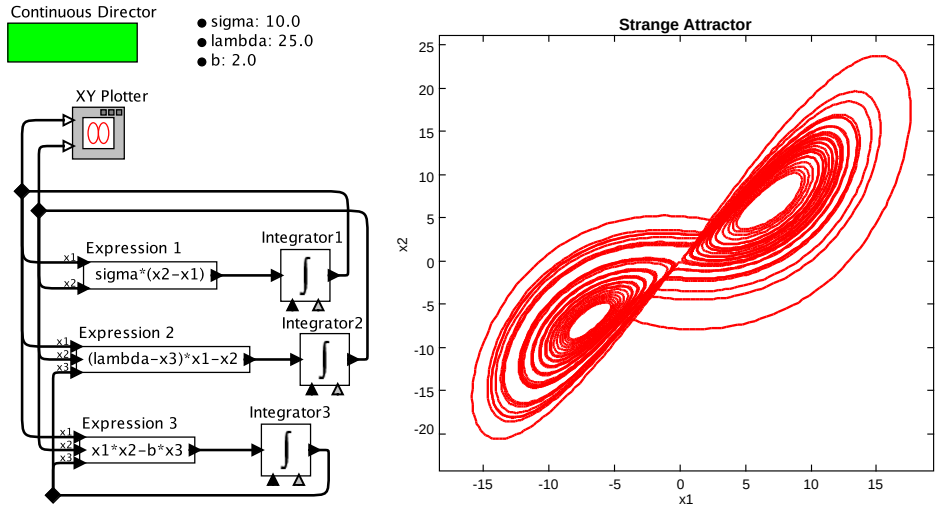

Fig. 1. Lorenz attractor executable model and a plot of x_1 vs. x_2 for an execution.

2014]. It implies that some classes of models have to admit nondeterminism. There is no way to avoid it.

2. CHAOS AND THE BUTTERFLY EFFECT

I have argued that determinism is a valuable property in models. A target in the physical world is not deterministic, but may nevertheless be usefully faithful to the model. Synchronous digital circuits and single-threaded imperative programs are excellent examples of this. Mechanical systems at scales where Newton's laws work well are also good examples.

For a deterministic system, if the initial conditions, inputs, and parameters are known precisely, then the future behavior is known. This would seem to imply that deterministic systems are predictable. But this is not really the case because of the caveat that the initial conditions, inputs and parameters need to be known *precisely*. If the model is chaotic, then arbitrarily small perturbations in any of these can result in arbitrarily different behavior in the future. This phenomenon is sometimes called the **butterfly effect**, invoking a metaphor that the air movement caused by a butterfly wing may eventually cause a hurricane, in that the hurricane would not have occurred had the butterfly not flown.

Chaos often arises with non-linear feedback systems. A well-known example of such a system is the Lorenz attractor, defined by the following set of differential equations:

$$\begin{aligned} \dot{x}_1(t) &= \sigma(x_2(t) - x_1(t)) \\ \dot{x}_2(t) &= (\lambda - x_3(t))x_1(t) - x_2(t) \\ \dot{x}_3(t) &= x_1(t)x_2(t) - bx_3(t) \end{aligned} \tag{1}$$

Augmented with initial conditions, these equations form a deterministic model. Another deterministic model of the same system is shown in Figure 1. That model is a *computational* model, a computer program (given with the graphical syntax of CyPhySim). It approximates equations (1), and hence is a model whose target is another model. Despite the fact that it *approximates* the other model, using Runge-Kutta or QSS numerical integration, it is nonetheless a deterministic model. How faithful is it to its target?

The answer to this question depends on how we measure the discrepancy between the idealized behavior of the differential equations and the computational model. In fact, measuring this discrepancy is difficult, because *every* computational model of the differential

equations will be "wrong" (in the Box and Draper sense). The model is faithful in the sense that the general shape shown in the plot in Figure 1 is the expected shape. It shows two "attractors," points around which the phase-space plot of x_1 vs. x_2 orbits. However, if we try to use this model to predict which of the two attractors we will be near sometime in the future, the utility of the model breaks down. This model is subject to the butterfly effect. An arbitrarily small error in numerical integration, or even in the arithmetic operations of (1), addition, subtraction, and multiplication, will make our prediction useless in the not-too-distant future.

Does this mean that only the computational model is useless, and the symbolic equation model of (1) remains useful for prediction? Probably not. Suppose that (1) is a model of a physical dynamics. Then very likely the parameters σ, λ, and b, and the initial conditions $x_1(0)$, $x_2(0)$, and $x_3(0)$ are physical quantities. Arbitrarily small errors in those physical quantities (which are unavoidable) undermine the predictive value of even the differential equation model.

I know of no CPS application that is reasonably modeled by the Lorenz attractor. However, chaos arises commonly in practical CPS scenarios. For example, Thiele and Kumar show that even simple, commonly used real-time scheduling policies exhibit chaotic behavior [Thiele and Kumar 2015].

A model that is chaotic, or even a model that is not known to be *not* chaotic, needs to be used with caution. It may have *some* predictive value. For example, the Lorenz attractor model suggests that its target system is **stable**. The values of the variables $x_i(t)$ remain within a bounded range. But their actual value at a future time cannot be usefully predicted by the model. A chaotic model may also be predictive over a reasonably short time horizon, though knowing the extent of this horizon could be difficult.

This problem with chaotic models is fundamental, and will be exhibited by *every* modeling framework. It weakens the value of determinism.

3. DISCRETIZING THE CONTINUUM

Most engineering problems occur at a scale where time and space are most usefully modeled as continuums. Continuums enable mathematical models using continuous functions, which are much better behaved and more easily manipulated than discontinuous functions. Within these continuums, however, discrete boundaries will occur. A physical object, for example, has edges that, for most purposes, will be modeled as abrupt. If you delve deeply enough into physics, it is arguable that no such boundary is abrupt. Where is the edge of the outermost electron of the outermost atom comprising an object? Every model of a physical object that has abrupt edges is wrong. But nearly all useful models are wrong in this way.

Cyber-physical systems intrinsically bridge the digital (cyber) world and the physical world. In the domain of software, continuums do not exist. In the world of software, edges are always abrupt. Hence, even if we use continuums to model the physical parts of a CPS, the mere presence of cyber components forces us to integrate continuums with discrete phenomena. An actuator, such as an electrical switch, will abruptly change from on to off and vice versa. The model of the physical side has to be able to deal with the resulting discontinuities.

Even some purely physical systems, with no cyber components, usefully mix discrete and continuous phenomena. Consider for example Newton's cradle, an instance of which is shown in Figure 2. The motion of the balls can be modeled using Newton's second law, which relates force (e.g. gravitation) with acceleration. Let x be the position of an object, v be its velocity, and F be the force on the object. These are all functions of time, which I assume is a continuum. These are related by

$$x(t) = x(0) + \int_0^t v(\tau) d\tau \tag{2}$$

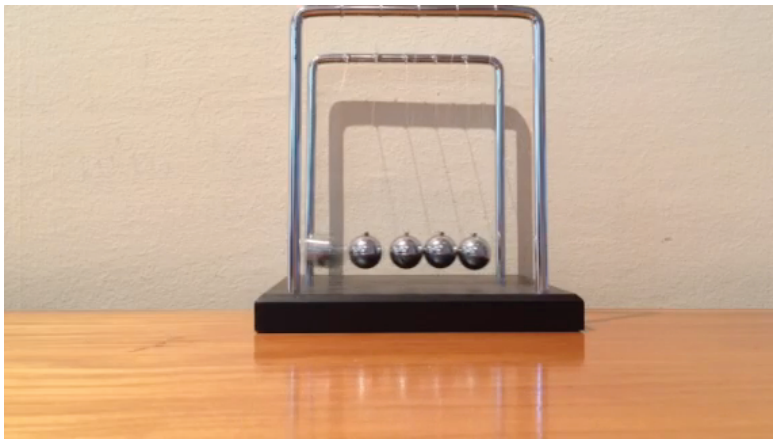

Fig. 2. Poor quality physical realization of Newton's cradle.

$$v(t) \;=\; v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau, \tag{3}$$

where m is the mass of the object. Under mild assumptions on the force function F, the velocity and position will both be continuous. However, for Newton's cradle, the collision between the balls may be better modeled with techniques that will yield discontinuous velocities.

Modeling collisions between rigid objects is a surprisingly difficult thing to do. Stewart gives an excellent overview of approaches that have been used towards solving these problems for collisions and friction between macroscopic physical objects [Stewart 2000]. In this regime, a solution that admits discrete behaviors can use generalized functions, most commonly the Dirac delta function, Lebesgue integration, measure theory, and differential inclusions. Stewart argues for embracing discrete behaviors in models, and shows that a well-known paradox in the study of rigid bodies known as the Painlevé paradox can be resolved by admitting impulsive forces into the model. An impulsive force produces an instantaneous change in momentum, or equivalently, an instantaneous change in velocity. An instantaneous change in momentum at time T of magnitude P can be modeled as follows,

$$v(t) = v(0) + \frac{1}{m}\int_0^t (F(\tau) + P\delta(\tau - T))d\tau \tag{4}$$

where δ is the Dirac delta function. Such a model for Newton's cradle imposes impulsive forces at the instants of the collisions between the balls. These impulsive forces result in instantaneous transfer of momentum from one ball to another and discontinuities in the velocity functions.

For the Newton's cradle example, modeling collisions as discrete gets tricky. Consider the simplest use of Newton's cradle, where you lift the left ball and release it. Let us number the balls 1 to 5 from left to right. Suppose that ball 1 collides with ball 2 at time T. Assume that the balls have exactly the same mass and that the collisions are perfectly elastic, where no kinetic energy is lost. Then at time T, ball 1 will instantaneously transfer its momentum to ball 2. Then, without any time elapsing, still at time T, ball 2 will collide with ball 3, and transfer its momentum. Ball 3 will then collide with ball 4, and ball 4 will collide with ball 5, all at time T. Ball 5 has nothing to collide with, so its acquired momentum becomes motion, and it lifts from the pack.

The chain of momentum transfers at an instant T creates mathematical difficulties when time is modeled using only the real line. The reader is referred to [Lee 2014] for an explanation of how a superdense model of time solves these problems, at least for some models. But this topic is beyond the scope of this paper, which focuses on problems that *cannot* be solved.

Figure 2 is a photograph of a particularly poor quality implementation of Newton's cradle. For this instance, the motion of the balls is sloppy, not behaving at all like predicted by the idealized model. The model is not faithful to its target. Which is flawed, the model or the target? To an engineer, the target is flawed. To a scientist, the model is flawed. Both perspectives are valid, but for different purposes.

Let us consider the scientist's perspective. To construct a more faithful model, we will need more than just Newton's laws of motion. Collisions between rigid objects, for example, involve localized plastic deformation, viscous damping in the material, and acoustic wave propagation. Much experimental and theoretical work has been done to refine models of such phenomena, leading to considerable insight into the underlying physical phenomena. How good is the predictive value of such models? The steel in the balls will have imperfections that will affect the plastic deformation, damping, and acoustic wave propagation. Imperfections in the spherical shape will affect the acoustic wave propagation. Elasticity and imperfections in the lengths of the strings will determine the points at which collisions occur. There are so many physical parameters here that there is little hope of knowing them all precisely. Moreover, the equations will likely become non-linear and exhibit chaos. Such more detailed models are unlikely to be any more faithful than the idealized (and much simpler) model. At the macroscopic scale, the predictive value of the simpler model is probably just as good (or bad) as that of the more complex model. And for the purposes of engineering, the simple idealized discrete model provides a *specification*, a benchmark against which the quality of a physical realization can be assessed. By itself, this is a valuable role for a model.

We could go part way towards a more complex model, and approximate the physics of the collision using a stiff spring between the balls. With some care, this results in continuous velocities, though it still has instantaneous discontinuities in acceleration. I will say more about this later, in Section 4, but for now I will just assert that over most operating conditions, such a model exhibits very similar behavior to the simpler discrete model, and where the behaviors differ, the model is extremely sensitive to parameters that are difficult to measure or know.

3.1. Computational Models Mixing Discrete and Continuous Behaviors

Equation (4) provides a mathematical model that mixes discrete and continuous behaviors. How can we create *computational* models for such mixtures? Such models have to embrace numerical approximation of continuous systems, using for example numerical integration. But that is not the only problem. It is also challenging to properly embrace discrete events.

Numerical integration and solving techniques for differential algebraic equations (DAEs) is an old and quite sophisticated topic. Excellent languages and tools are available for analyzing and numerically solving DAEs. One of the best is Modelica [Tiller 2001], a widely used language with well-supported libraries of models for a large variety of physical systems. Although the technology continues to improve, even such state-of-the-art tools have had difficulty mixing in discrete behaviors. Otter, et al. in [Otter et al. 2005] state that "at the moment, it is not possible to implement the solution with impulses ... in a generic way in Modelica." They offer continuous approximations as an alternative, categorizing three approaches for collisions: impulsive, spring-damper ignoring contact area, and spring-damper including contact area. They describe a library in Modelica that uses the latter two approaches.

A CyPhySim model for equations (2) and (4) is shown in Figure 3. A key feature of that model is that the impulsive force, given as an instantaneous change in momentum P, is

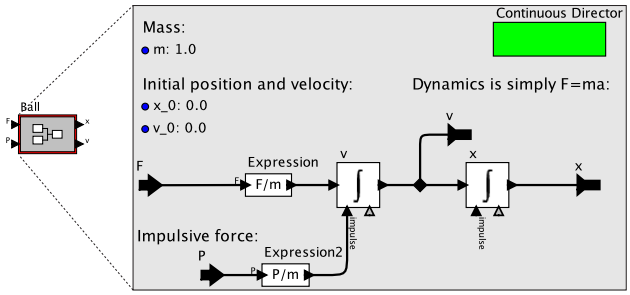

Fig. 3. Ball dynamics.

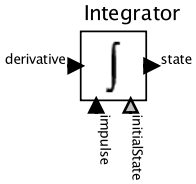

Fig. 4. CyPhySim Integrator has "impulse" input.

provided on a separate input port from the continuous force F. There are two reasons for doing this. First, the Dirac delta functions are discrete events, absent most of the time and present only at the instants of collisions. The *value* of the force at that instant is not finite, but the *weight* of the delta function, which has units of momentum, is finite. The semantics of these two inputs, F and P, are sufficiently different to justify providing them separately.

Second, and perhaps more interestingly, P is provided on a distinct input port because the causality relationship between $P(t)$ and $v(t)$ is not the same as the causality relationship between $F(t)$ and $v(t)$. In particular, because of the fundamental properties of integration, at a time t, $v(t)$ does not depend on $F(t)$.[3] It only depends on earlier values of F. However, $v(t)$ *does* depend on $P(t)$. The impulse immediately affects the velocity. There is **direct feedthrough** from the input P to v, but not from F to v. We will see below that this direct feedthrough constrains how we can construct feedback models.

Examining closely the CyPhySim integrator component shown in Figure 4, we see that indeed the weight of an impulse is provided on a distinct input port labeled "impulse." The signal provided to this port is required to be **discrete** in a very technical sense elaborated in [Lee 2014], but beyond the scope of this paper. For our purposes, it is sufficient to use an intuitive description: the signal provided to the "impulse" input is **absent** almost everywhere, and present with a weight at the time of a Dirac delta function.

3.2. Bouncing Ball Model

To better understand the implications of the causality relation between an impulse and the integrated output, consider the well-studied bouncing-ball problem. Consider a ball that is subject to the constant downward force of gravity. The gravitational force is shown on the left in Figure 5, with value $-9.8m$, where m is the mass. The `Ball` in the figure is the one in Figure 3. This force causes it to accelerate until it collides with a rigid surface.

Let's assume a partially inelastic collision that loses some energy so that each bounce of the ball rises to a lesser height than the previous bounce. When it collides with the surface,

[3]Note that *implicit* numerical integration techniques introduce additional dependencies, and when using such techniques, it is necessary to know (or estimate) $F(t)$ in order to calculate $v(t)$. This dependence, however, is not present in the fundamental model from calculus nor in *explicit* numerical integration techniques.

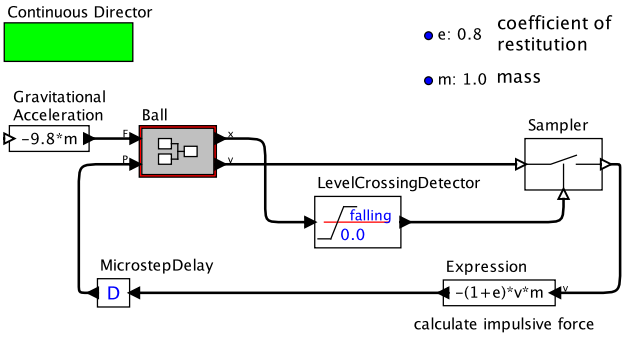

Fig. 5. Bouncing ball model.

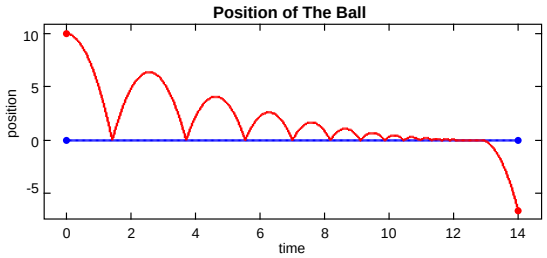

Fig. 6. Bouncing ball execution.

we can calculate an impulsive upwards force that will instantly reverse the velocity of the ball. Figure 5 shows this model. An execution of this model is shown in Figure 6.

There are two interesting conundrums about this simple model. First, at the time of the collision, there is a potential causality loop. That causality loop is broken in Figure 5 by the `MicrostepDelay` component, whose output is the same as its input, but delayed by one microstep in superdense time. If this `MicrostepDelay` were omitted from the model, we would have a causality loop. At the time t of a collision, the velocity $v(t)$ would depend on the impulse force $P(t)$, which in turn would depend on $v(t)$. We would be unable to calculate $v(t)$ or $P(t)$ because of this circular dependence. This would be an example of a **non-constructive** model [Berry 1999].

The fact that a non-constructive model emerges from a discrete description of the physics of the problem is disturbing. I will say more about this problem below, but first, there is another equally disturbing problem with this model. The time between bounces shrinks rapidly enough that, in theory, an infinite number of bounces will occur in finite time. This is called a **Zeno** condition. But in the executable model, instead of getting an infinite number of bounces, we see at the right of the plot in Figure 6 that the ball tunnels through the surface and goes into a free fall below the surface!

The velocity and position of the ball lie in a continuum. The surface is modeled as discrete. The position function crosses a discrete threshold, which in an idealized sense, requires infinite precision. But level-crossing can only be done computationally up to some error, and the resulting error will inevitably be large enough that the ball tunnels through the surface.

Figure 7 explains why this happens. To detect the level crossing, this particular detector has a parameter controlling the precision, and it allows the ball to penetrate the surface a certain amount when detecting a collision. Hence, eventually, the ball will not have enough

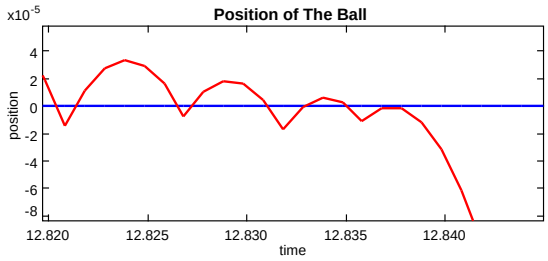

Fig. 7. Bouncing ball execution closeup showing tunneling.

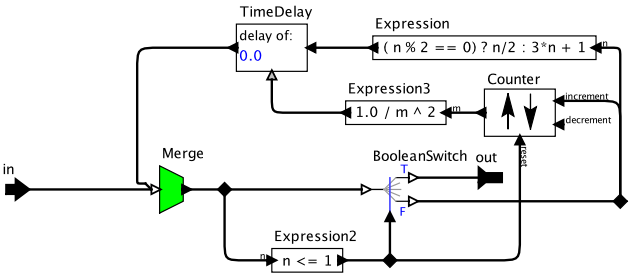

Fig. 8. Model that is Zeno if the Collatz conjecture is false (due to Ben Lickly).

energy to rise above the surface. Any implementation that allows penetration and models the surface as a discrete threshold will suffer this tunneling.

As with Newton's cradle, we might conclude that using discrete models isn't the right choice. We could construct a more detailed model of the physics, but as I've observed before, the resulting model may not be any more faithful, and the cost in modeling complexity can be high. We will see below that there is an atlernative solution that further embraces discreteness, which is to model this system as a *modal model*.

Although the computational model in Figure 5 is deterministic, it is difficult to predict when the tunneling will occur. This will depend on the exact details of numerical computation, including the precision with which real numbers are represented and the algorithms used to perform arithmetic on them. At this point, it might be more useful to consider the model to be nondeterministic, and not even attempt to include in the model such computational details. An idealized model, one that embraces the continuum, becomes a deterministic ideal against which the nondeterministic approximations are evaluated. This seems to suggest a tradeoff between precision and determinism, in that higher precision could get us "closer" (in some sense) to the deterministic ideal. However, as I will show below, this line of reasoning breaks down because determinism is incomplete. Even letting the precision approach infinity does not solve the problem.

3.3. Detecting Zeno Behavior is Hard

The bouncing ball model above exhibits Zeno behavior, where an infinite number of events occur in finite time. Any modeling framework that is sufficiently expressive to describe such models will have challenges handling the models. A framework that handles all events in chronological order will either get stuck at the Zeno point or suffer from unexpected behavior due to quantization errors near the Zeno point (Figures 6 and 7 illustrate the latter). Techniques exist for getting past the Zeno point in practice (see for example [Zheng et al. 2006]) and in theory (see [Matsikoudis and Lee 2013], which uses transfinite induction), but these techniques are either incomplete (they fail on some models) or not (directly) computa-

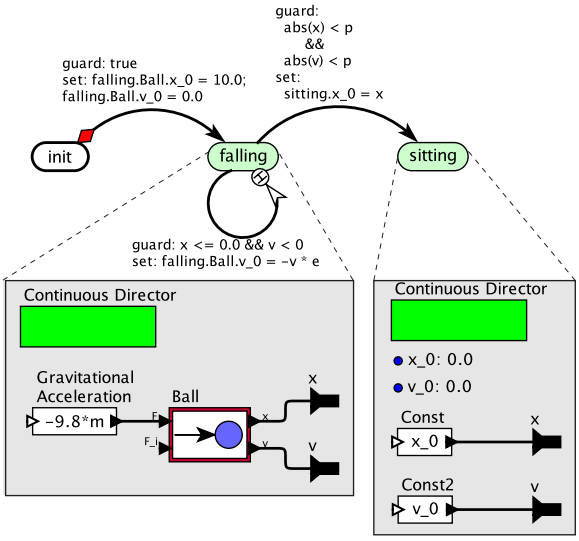

Fig. 9. Modal model of the bouncing ball that does not tunnel.

tional. It would be desirable if Zeno models could be recognized and rejected to circumvent such problems, but unfortunately, this is not possible. Figure 8 shows a model (due to Ben Lickly) that may or may not have Zeno behavior. Whether it has Zeno behavior depends on whether a famous conjecture is true or false. Specifically, the **Collatz conjecture** states:

> For any natural number $n \geq 1$, if n is even, divide it by 2; if n is odd multiply it by 3 and add 1. Repeat the process indefinitely. The conjecture is that no matter what number you start with, you will always eventually reach 1.

This conjecture is as yet unproven true or false. If the conjecture is false, then the model in Figure 8 is a Zeno model when the input is a counterexample to the conjecture. An infinite number of events will circulate in the loop with spacing between them equal to $1/m^2$, where m is the count of events from the input that is the counterexample. Otherwise, the model does not exhibit Zeno conditions for any input. This shows definitively that detecting Zeno systems is hard.

3.4. Modal Models

One possible interpretation of the problem with the bouncing ball is that the model that detects a collision with the surface becomes invalid when the ball motion drops below some threshold. Indeed, in the physical world, a ball won't bounce an infinite number of times. It will stop. The model of a stopped ball is quite different from the model of a bouncing ball (and much simpler!).

Modal models (generalized hybrid systems) split models into *modes*, and provide a transition system ensures that a mode is active only when the model in that mode is valid. A modal model for the bouncing ball is shown in Figure 9. Details of the notation in this (executable) model are given in [Ptolemaeus 2014], but for our purposes, it is enough to notice that guarded transition moves the model from `falling` to `sitting` when the position and velocity drop below a threshold. Interestingly, the behavior of the model is not very sensitive to the choice of this threshold, as long as the transition is taken before the ball tunnels.

A plot of the new execution is shown in Figure 10. This model has more discrete elements than the original model, in that the guarded transitions are (semantically) instantaneous.

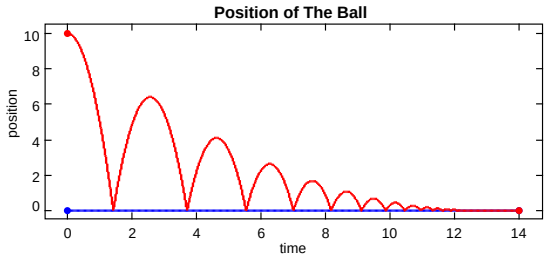

Fig. 10. Trajectory of the bouncing ball that does not tunnel.

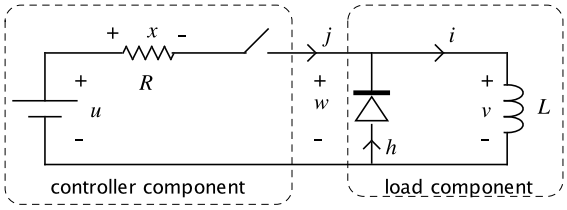

Fig. 11. A model of a physical system where causality changes when modes change.

By making the model more discrete, we are also able to make it simpler. And we no longer confront the tradeoff between determinism and precision.

3.5. Modal Causality

In this section, I study a modal model where the causality relationship between variables changes between modes. Consider the example shown in Figure 11, which was also considered in [Lee 2014; Mosterman and Biswas 1995]. This is a very practical circuit, representing a pattern that is widely used in CPS when a switch is used to turn on and off an inductive load, such as a motor. The dashed lines abstract this circuit as two components, a controller on the left and the load on the right. When the switch is closed, the controller supplies a voltage w and current j to the load. There are several variables in this model, currents h, j, and i, and voltages u, v, and w. To build a constructive model, we would like to identify some of these variables as inputs and outputs of the two components.

The notion of constructive models is defined formally in [Berry 1999], but loosely a model is constructive if its behavior can be found by a terminating procedure that changes variables from unknown to known in a sequence that is based on their causal relationships; i.e., if a causes b, then a must become known before b can become known.

An inductor has memory. It has the property that current flowing through it at a time t does not depend on the voltage across it at time t, but rather only depends on the history of past voltages. Specifically, for the currents and voltages in the figure,

$$i(t) = \frac{1}{L}\int_0^t v(\tau)d\tau, \tag{5}$$

where L is the inductance. This assumes that the initial voltage $v(0)$ is zero at time zero; i.e., we assume that the system starts in the off position (the switch is open) and that the inductor is storing no energy. It is a basic property of integration in calculus that $i(t)$ in (5) does not depend on $v(t)$ at any time t.

During normal operation of this circuit, assuming that the battery voltage u is positive and the switch is closed, the diode will be reverse biased, meaning that no current flows through it (I neglect leakage). At such times t, $h(t) = 0$, and consequently, $j(t) = i(t)$.

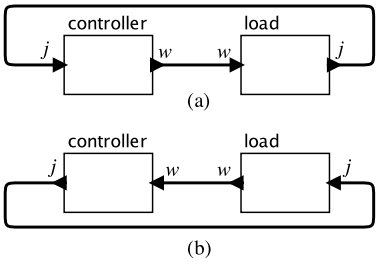

Fig. 12. Causality relationships of the load and controller components of Figure 11. In (a) the diode is reverse biased and the switch is closed. In (b), the diode is forward biased and the switch is open.

Moreover, at all times t, $v(t) = w(t)$. Hence, when the diode is reverse biased, the load component is defined by

$$j(t) = \frac{1}{L} \int_0^t w(\tau) d\tau, \tag{6}$$

With this definition, we must consider w the input and j the output, because the environment of the load component cannot arbitrarily set the current j independent of the history. Figure 12(a) shows the load component with input w and output j.

For the controller, because of the resistor, w depends on the current j that the load draws. Hence it is reasonable to consider j to be an *input* and w to be an *output*, as shown in Figure 12(a) The battery symbol on the far left of Figure 11 represents an ideal voltage source producing voltage u, so the battery together with the resistor model a practical battery.

Notice that for the controller, the voltage $w(t)$ depends instantaneously on the current $j(t)$ at time t, by Ohm's law. There is no delay from the input $j(t)$ to the output $w(t)$, so there is *direct feedthrough*. Put another way, we cannot know the output $w(t)$ unless we know the input $j(t)$. By contrast, for the load component, the output $j(t)$ does not immediately depend on the input $w(t)$, because of (6). Hence, that component does not have direct feedthrough. As a consequence, the feedback loop in Figure 12(a) is constructive. At any time t, the output $j(t)$ of the load can be determined without knowing the input $w(t)$. A **causality loop** is averted. We therefore have a **constructive procedure** for finding the values of both $w(t)$ and $j(t)$ at time t.

When the switch is open, the situation is very different. In this case, the current j is forced by the switch to be identically zero. This is what a switch does. Hence, the current j is no longer an input to the controller component. It is an output. Moreover, since the switch is open, the controller no longer has any control over the voltage w. Therefore, that voltage is no longer an output, and can only be considered an input.[4] This reversal of causality is shown in Figure 12(b).

For the load component, with the switch open, the current j has now become an input. We have already established that j cannot be an input if the diode is reverse biased. Indeed, in this circuit, when the switch is opened, the diode becomes forward biased, and current can flow through it. In this state, the current h through the diode is equal to i, the current through the inductor. The voltage w is now determined by the diode, and hence is an output of the load component (this voltage is a property of the diode technology, but we can assume for our purposes an ideal diode where this voltage is zero). Figure 12(b) shows a revised

[4]Here, I assume that every variable must be either an input or an output. In this case, the input has no effect, but since it cannot be an output, it must be an input.

load component, where the current j is an input (forced to zero by the controller) and the voltage w is an output.

Figure 12(b) also averts a causality loop. For the controller, the current j, the output, does not depend on the voltage w, the input, and hence, in this mode, there is no direct feedthrough from input to output. We do not need to know $w(t)$ at time t to determine $j(t)$. The circuit is again constructive.

The behavior of this circuit is naturally modeled using a modal model. There are potentially four modes, because each of the two components has two modes. The controller component has a switch that is either open or closed, and the load component has a diode that is either forward or reverse biased. A detailed model is given in [Lee 2014]. In that model, two of the four modes (switch is closed and diode is forward biased, or switch is open and diode is reverse biased) are **transient**. The model spends no time in them. Such transitory modes are called "mythical modes" by Mosterman and Biswas [Mosterman and Biswas 1995]. In fact, in two of the four modes, the model is not constructive. It has a causality loop.

The most interesting feature of this model, however, is not the mythical modes. It is the reversal of causality when switching between modes. What is an input in one mode becomes an output in the other. Moreover, the direct feedthrough property changes. These changes in causality appear to be a direct consequence of the physics of the model.

A continuous model of this system would have to get much more detailed about the physics, delving into the mobility of charged particles. Again, there is risk that uncertainty about material details of the system being modeled will undermine the fidelity of the model, resulting in a model that is no more trustworthy than the discrete model. An alternative is to reject the notion of causality altogether, something that is advocated by some philosophers.[5] But rejecting the notion of causality just so that we can reject the notion of discreteness is simply too high a price to pay. This author prefers to accept discreteness.

Note that even languages that embrace acausal models, such as Modelica, go to great lengths to convert the models to causal form, e.g. using the Pantelides algorithm [Pantelides 1988]. Only when such conversion is not possible are algebraic loop solvers used, and in such cases, considerable modeling care is required to interpret the results, since solutions may not exist or may not be unique.

3.6. Causality Loops

In equation (6), the reader may have noticed a sleight of hand. I assumed that the initial voltage $w(0)$ was zero. Otherwise, the correct equation is a little bit different,

$$j(t) = \frac{1}{L}\left[w(0) + \int_0^\tau w(\tau)d\tau\right]. \tag{7}$$

Now we have a problem. At time zero, the integral is zero, so

$$j(0) = \frac{1}{L}w(0). \tag{8}$$

The current $j(0)$ depends immediately on the voltage $v(0)$. If the voltage is an input and the current is an output (for the load component), then there *is* direct feedthrough, though only at time zero. Since the controller component on the left also has direct feedthrough, the voltage $w(0)$ depends immediately on the current $i(0)$, we have a **causality loop**. If we do not know $v(0)$, then can't find $i(0)$, and if we do not know $i(0)$, we cannot find $v(0)$.

This problem is resolved in [Lee 2014] by ensuring that whenever we enter the mode where the diode is reverse biased, the voltage across the inductor gets set as an **initial condition**.

[5]See [Price and Corry 2007], which includes an essay arguing that there is no basis in physics for the notion of causality [Norton 2007]. Causality, it claims, is a human cognitive construction.

At the very start of the execution of the model, if the initial mode is such that the diode is reverse biased, then that initial voltage is a parameter of the model. Upon subsequent entries into this mode, the initial voltage is simply set equal to the voltage w in the *previous* mode. This setting is part of the **mode transition function**.

There are circuits, however, where the resolution to such transient causality loops is not so obvious (the causality loop is transient because it lasts zero time; after any infinitesimal time has passed, there is no longer a causality loop). See [Cellier and Kofman 2006], page 582, for an example that intrinsically includes a causality loop. To support such models, CyPhySim includes three **algebraic loop solvers** that can be tried [Lee et al. 2015]. However, with such causality loops, we have no assurance of the existence or uniqueness of a solution, so considerable care in modeling is required.

3.7. Representing a Continuum

Computers cannot directly deal with quantities in a continuum. Every object manipulated by a digital computer is a member of a countable set (or strictly speaking, a *finite* set, since memory is always bounded; but often the bound is so large that we can neglect the finiteness). When modeling physical behaviors in a computer, we have to approximate the continuum somehow. Furia et al. state "normal numerical algorithms deal with rational numbers since they can approximate real numbers" [Furia et al. 2010]. However, rational numbers are in fact quite expensive to compute with. Representing a rational number (fully) requires two unbounded integers, one for the denominator, and one for the numerator. Adding two rational numbers requires finding the least common multiple of the denominators, scaling the numerators and denominators, and then adding. If we then want to reduce each rational number to a canonical form, we have to factor the numerator and denominator of the result. As a consequence, numerical algorithms that deal with rational numbers are rare indeed.

The usual approximation to real numbers in computer programs is floating-point numbers conforming to the IEEE 754 standard. The sets of `float` or `double` numbers, however, are not dense in the reals (there are real numbers that are not limits of sequences of floating-point numbers). Moreover, these sets are in fact *finite*. Hence, the properties of these sets are distinctly different from the properties of real numbers, and it is not correct to simply assume that one can arbitrarily closely approximate real numbers.

In many physical problems, it makes little sense to talk about *equality* of two distinct quantities in a continuum. For example, we would be on shaky ground to assert that the voltage across two distinct capacitors is equal. Any such assertion would force our model down to the resolution where quantum mechanics applies, and at such scales, the assertion is prohibited.

Consider a bouncing ball model that places the position of the ball in a continuum. If the position of the falling ball is continuous, then there must be instant in time (where time is also modeled as a continuum) when the position of the ball *equals* the position of the surface. In the idealized model using continuums, this is the instant at which the ball bounces.

Such equality is problematic in physics and even more problematic in computation. Neither the position nor the time can be represented in the computer as members of a continuum. Hence the need for approximation. Even if the position of the falling ball is continuous in the continuum model, in any computational model, the position of the ball may progress from above the surface to below it without ever being equal.

But there is an even more subtle phenomenon that we cannot ignore: when we represent quantities using a finite set, such as `double`, it *is* possible for two quantities to be equal. It is essential for any modeling framework that approximates a continuum using such a finite set to give a semantic meaning to such equality. Equality *will* occur sometimes. It needs to mean something. It cannot be ignored.

Consider representing time using a `double`. Many modeling and simulation frameworks do this (CyPhySim is an exception [Ptolemaeus 2014], as is, for example, VHDL). Since two doubles can be equal, such modeling frameworks allow for **simultaneity** (the times of two events can be equal). But does this simultaneity have any semantics? Broman et al. argue that floating-point numbers are a poor choice for modeling time because we cannot assign any significant semantics to equality [Broman et al. 2015]. The core reason for this is that addition is not associative and resolution is not independent of magnitude for floating-point numbers.[6] In fact, most bug checkers for software (such as Coverity) will flag tests for equality of floating-point numbers as a potential bug.

As a consequence, computational models that approximate continuums should use care in how they do so. CyPhySim represents time as an (unbounded) integer multiple of a global **resolution**, a parameter of the model. As a consequence, addition is associative, resolution is independent of magnitude, and simultaneity is independent of the accidental ordering of simple arithmetic operations. See [Ptolemaeus 2014] for details.

4. LIMITS OF DETERMINISM

In this final section, I observe a fundamental limit that was, to my knowledge first described in [Lee 2014]. This limit is more speculative than the ones discussed earlier in this paper and its consequences more profound.

Specifically, consider a set of deterministic models that is rich enough to encompass at least models like the Newton's cradle with discrete collisions. That is, the modeling framework needs to include an ability to model Newton's second law augmented with impulsive forces. Then we can construct a sequence of deterministic models whose behavior gets closer and closer together, but the sequence has no limit in the set of deterministic models. Every model in the sequence is deterministic, and the models in the sequence get arbitrarily "close" to each other, but the limit is not deterministic. This suggests that the set of deterministic models is not complete, though defining "completeness" in general seems tricky. As a practical consequence, this result suggests that we cannot be complacent when building deterministic models, because there could be corner cases, parameter values for the models, where their determinism breaks down. These cases may be unavoidable, at least when discrete behaviors are modeled discretely.

Formalizing this in general is not trivial, as it requires defining a "limit" of a sequence of models and of "closeness" of models. Here, I will make the case for a family of very simple models, but I fully acknowledge that it is a leap of faith to assert that determinism is incomplete for modeling in general. Nevertheless, it seems defensible on philosophical grounds and consistent with modern physics (which is all about models).

4.1. Collisions: Mixing Discrete and Continuous

To simplify Newton's cradle, we can reduce it to one dimension. Consider the scenario depicted in Figure 13. An ideal billiard ball in motion approaches a second ball on a frictionless surface. Let us model the collisions using impulsive forces.

Assuming the balls are perfectly elastic, then after the collision, both momentum and energy should have been conserved. Let v_1 and v_1' be the speed of the first ball before and after the collision, respectively. Let v_2 and v_2' similarly represent the speed of the second ball before and after the collision. Conservation of momentum requires

$$m_1 v_1' + m_2 v_2' = m_1 v_1 + m_2 v_2, \tag{9}$$

[6] **Resolution** refers to the ability to distinguish or "resolve" two quantities. The gap between one floating-point number and the next larger floating-point number depends on the magnitude of the number.

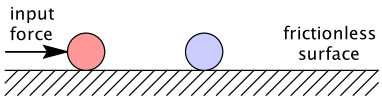

Fig. 13. Two balls.

where m_1 and m_2 are the masses of the two balls, respectively. Conservation of kinetic energy requires

$$\frac{m_1(v_1')^2}{2} + \frac{m_2(v_2')^2}{2} = \frac{m_1(v_1)^2}{2} + \frac{m_2(v_2)^2}{2}. \tag{10}$$

Assuming we know the starting speeds v_1 and v_2 and the masses, then we have two equations and two unknowns, v_1' and v_2'. This is a quadratic problem with two solutions.
Solution 1: $v_1' = v_1$, $v_2' = v_2$ (ignore the collision).
Solution 2:

$$v_1' = \frac{v_1(m_1 - m_2) + 2m_2v_2}{m_1 + m_2} \tag{11}$$

$$v_2' = \frac{v_2(m_2 - m_1) + 2m_1v_1}{m_1 + m_2}. \tag{12}$$

Note that if $m_1 = m_2$, then the two masses simply exchange velocities (as in Newton's cradle). Note further that solution 1 corresponds to tunneling, albeit for a very different reason than in the bouncing ball example.

Now consider a more elaborate scenario, shown in Figure 14. Two (ideal) billiard balls approach a third stationary ball from opposite sides on a frictionless surface and collide with the stationary ball simultaneously. How should they react?

If we consider only conservation of momentum and energy, then there is more than one possible outcome. The conservation laws give us two equations, but there are now *three* unknowns. Embracing discrete models, at the time of the collision, there are *two* collisions, each with two outcomes. If we reject the tunneling solutions, then it would seem that only one outcome remains. But it is not obvious how to combine the two non-tunneling outcomes.

A first (naive) solution, using what is known as **Newton's hypothesis**, just superimposes the resulting momentums of the two non-tunneling outcomes. If the balls all have the same mass, then the left ball will transfer its momentum to the middle ball, the right ball will also transfer its momentum to the middle ball, and the equal and opposite momentums will cancel. All balls stop. Momentum is conserved, but not energy. This solution is shown at the top of Figure 15. Since there is no mechanism for energy dissipation in this model, this resolution is not satisfactory.

An alternative solution, using what is known in the literature as **Poisson's hypothesis**, introduces a form of superdense time. At the time of the collision, the kinetic energy of the balls is instantly converted to potential energy by compressing the middle ball. Then, without time elapsing, in a second microstep, the potential energy is reconverted to kinetic energy by the middle ball expanding. But how should this ball apportion the kinetic energy to the two outer balls? As it happens, if the masses of the balls are equal, there is

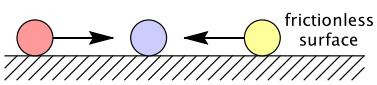

Fig. 14. Simultaneous collisions where one collision does not cause the other.

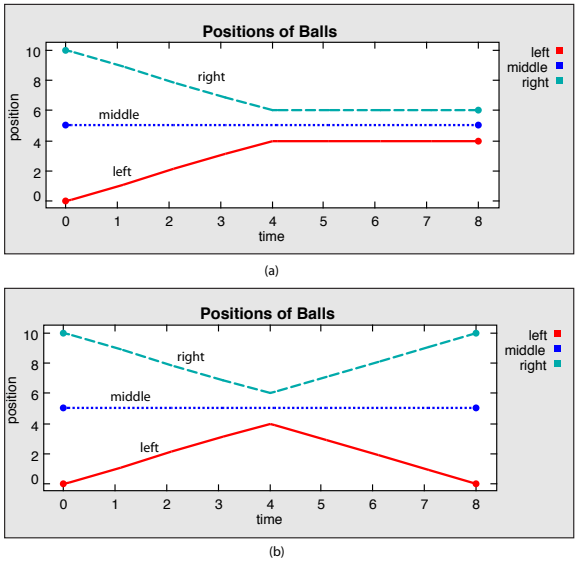

Fig. 15. Newton's hypothesis vs. Poisson's hypothesis.

only one solution, shown at the bottom of Figure 15. But if the masses are unequal, then there are many solutions that conserve both momentum and energy! We seem to have no basis for picking one solution over another, so the model appears to become intrinsically nondeterministic.

Perhaps we can resolve this conundrum by questioning the notion of *simultaneity*. Let's assume that the collisions occur in some order. To embrace discreteness and leverage superdense time, let's assume that no time elapses between collisions. As shown in Figure 16, when the collisions occur, we arbitrarily pick one to handle, the left one in the figure, and ignore the other collision. We handle that collision, rejecting the tunneling solution. Without time elapsing, we find ourself in state (c) in the figure, where the middle and right ball are traveling towards one another and colliding. Now there is only one collision, so we handle it in (d), leaving us in state (e). Again, without time elapsing, there is a new collision, which we handle in (f), leaving us in state (g). After time elapses, we find ourselves in state (h).

The balls move away at equal speed, but only if their masses are the same! If the masses are different, then the choice of which collision we handle first affects the final outcome, as depicted in Figure 17. More than one final state conserves both momentum and energy.

Arbitrary interleaving of the collisions yields the right result (for any choice of interleaving), in that momentum and energy are conserved. But if the masses are not the same, even after excluding tunneling, there is more than one "right" result, as shown for a particular example in Figure 17. There, three balls with masses 0.2, 1.0, and 5.0 collide simultaneously. Depending on which collision is handled first, the trajectories differ. Both shown outcomes have the same momentum and energy after the collision as before.

Inevitably, when I present this example, someone asks me how the "real world" behaves in this example. This is a difficult question. Recall that the Heisenberg uncertainty principle states that we cannot simultaneously know the position and momentum of an object to arbitrary precision. But discrete modeling of these collisions depends on knowing position and momentum precisely. Interestingly, quantum mechanics resolves this conundrum with probabilistic models, which like our arbitrary interleaving and nondeterminism, admit more than one behavior.

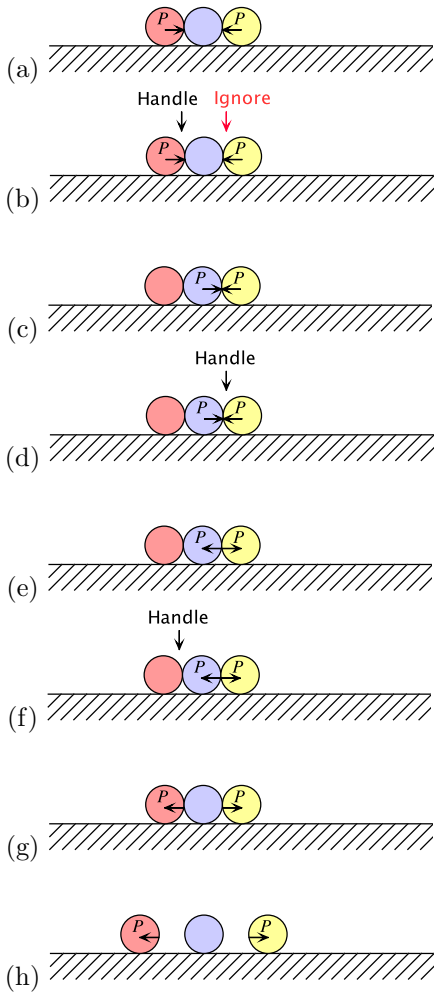

Fig. 16. One of two orderings for handling collisions.

It probably seems odd to be invoking quantum mechanics on macro-scale physics problems, where Newtonian mechanics usually works just fine. But we are talking about *models*, and in our model, position and velocity are real numbers, and therefore have infinite precision, in violation of the Heisenberg uncertainty principle. Arguably, physics tells us that when the time between collisions gets small enough, our Newtonian model is no longer valid, and we need to switch to some other model.

Another way to resolve this conundrum would be to switch to a model that is more detailed about the physics of collisions. A relatively small step in that direction, short of modeling localized plastic deformation, viscous damping in the material, and acoustic wave propagation, would be to model the collisions using stiff springs at the boundaries of the balls. Such a model is deterministic, and the velocities of the balls are continuous functions of time, but can we have confidence in the result? If we assume the stiffness of the springs is high, then the behavior exhibited by the model will be very sensitive to the time between collisions. Figure 18 shows multiple trajectories under this stiff-spring model where the

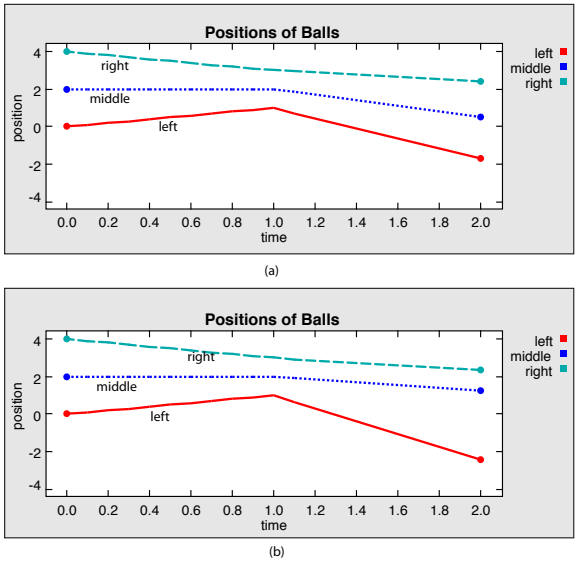

Fig. 17. If the masses are different, the behavior depends on which collision is handled first.

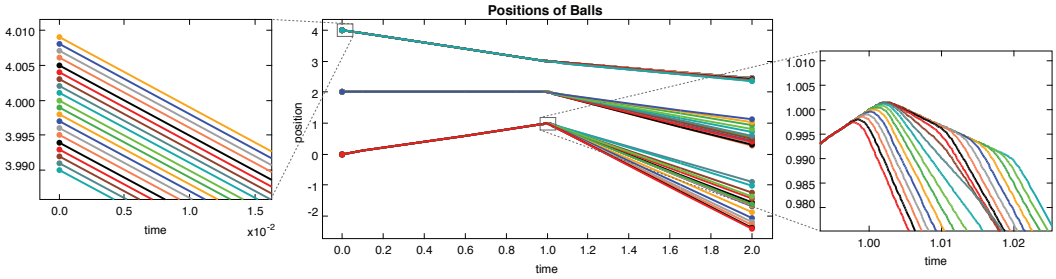

Fig. 18. Trajectories with a continuous, stiff-spring model where the initial position varies over a small range.

initial position of the right ball is varied slightly as shown in the figure. Each trajectory is the result of a deterministic model, but the final behavior is very sensitive to the initial position. As with chaotic models, the predictive power of this deterministic model is suspect, and may in fact be no better than the predictive power of the discrete nondeterministic model. Moreover, the determinism of the model could be misleading, inducing a false sense of confidence in the model.

Since the time of the collisions cannot be known precisely, we could elaborate our model with probabilities that quantify our uncertainty about the parameters. However, if the stiffness of the springs is high enough, these probabilistic models likely become bimodal, reflecting a temporal ordering between collisions. In this case, the probabilistic model becomes remarkably similar to the discrete nondeterministic model, though at considerable cost in complexity.

As with all modeling exercises, we have many choices for how to model this system. Which model to choose will depend on our modeling goals. Do we want a typical behavior? The set of all possible behaviors? The set of possible behaviors weighted by probabilities? The extreme behaviors? Any plausible behavior? This choice should be explicit. And we should always choose the simplest model that meets our objectives.

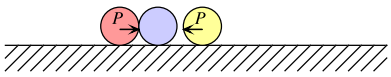

Fig. 19. Non-simultaneous collisions.

4.2. Is Determinism Incomplete?

The above example has the odd property that a sequence of deterministic models "converges" (in some sense) to a nondeterministic one. This suggests that the set of deterministic models is incomplete (in some sense), at least when discrete behaviors are allowed. A general statement about the incompleteness of determinism is difficult to make, because the set of all models is ill defined. Also, there are many subtly different notions of completeness in mathematics. If we restrict our attention to a well-defined subset, and choose a particular notion of completeness, then we can argue this incompleteness.

Specifically, I will study a particularly small set of well-defined models. Consider the set M of models describing one-dimensional motion of $N = 3$ ideal elastic balls subject to Newton's second law, where collisions are handled with impulsive forces, and all behaviors that conserve momentum and energy are allowed except for tunneling. The equations of motion are (2) and (4). There are no external forces, so all behaviors are a consequence of the initial conditions. We can assume that these equations are solved ideally, since they are simple enough that given any initial condition, all eventualities can be derived in closed form. The set M includes three balls with masses $m_1 = 0.2$, $m_2 = 1.0$, and $m_3 = 5$ (these are the masses that generate the behaviors shown in Figure 17). Assume initial positions $x_i(0) \in \mathbb{R}$, and initial velocities $v_i(0) \in \mathbb{R}$ given as follows:

$$\begin{aligned} x_1(0) &= -1 \\ v_1(0) &= 1 \\ x_2(0) &= 0 \\ v_2(0) &= 0 \\ x_3(0) &= 1 + \Delta \\ v_3(0) &= -1 \end{aligned}$$

where $\Delta \in \mathbb{R}$ is a real number. That is, there are three balls, with the two outer ones moving towards the stationary middle one. The set M is now well defined. The only distinguishing feature between models in M is the value of the parameter Δ. When $\Delta \neq 0$, the collisions will not be simultaneous, as illustrated in Figure 19.

Following [Lee and Sangiovanni-Vincentelli 1998], we formally define a **model** as a set of **behaviors**. We will consider the behavior of each model over the time interval $[0, 2]$ only, since this is sufficient for our purposes. Specifically, let B be the set of all functions of the form $b\colon [0, 2] \to \mathbb{R}^3$. For a particular model $A \in M$ with parameter Δ_A, we say that $x \in B$ is a **behavior** of A if for all $t \in [0, 2]$, $x_i(t)$ is the position of ball i at time t, for $i \in \{1, 2, 3\}$. These x_i are functions of time that satisfy the equations of motion and conservation laws. In words, a behavior of a model $A \in M$ is three functions giving the positions of the three balls as a function of time.

A particular model $A \in M$ may have more than one behavior, in which case the model is nondeterministic [Lee and Sangiovanni-Vincentelli 1998]. In particular, as shown by example above, there is more than one behavior that satisfies the equations of motion and conservation laws when $\Delta_A = 0$. In fact, in our space of models, that is the only nondeterministic model.

Consider the subset $D \subset M$ of deterministic models. We can define a metric on D that allows us to talk about models being "close." It does not matter much which metric we

choose. To be concrete, consider two models $A, A' \in D$. Since these models are deterministic, each has exactly one behavior. Let those behaviors be $x, x' \in B$, respectively. We can define a distance function d as follows,

$$d(A, A') = \frac{1}{2} \int_0^2 ||x(t) - x'(t)|| dt, \tag{13}$$

where $||x||$ is the L1 norm of a real vector x. It is easy to show that d is a metric.

Consider a sequence of models $A_i \in D$, $i \in \{1, 2, \cdots\}$ where

$$\Delta_{A_i} = 1/i^2.$$

It is easy to show that this sequence is Cauchy, which means that for any $\epsilon > 0$, we can find a positive integer N such that for all positive integers $i, j > N$,

$$d(A_i, A_j) < \epsilon.$$

However, this sequence has no limit in D, which means that the metric space D of deterministic models is incomplete. It does not contain all its limit points. Every model in the sequence is deterministic, and the models in the sequence get arbitrarily close to one another. But there is no limit that is a deterministic model.

In [Lee 2014], I show that a direct description of this scenario results in a *non-constructive* model [Berry 1999; Mendler et al. 2012] when Δ reaches zero. To make the model constructive, we have to arbitrarily choose one of the possible behaviors. My formulation assumes that this choice is handled nondeterministically. Does this assumption alone predispose the result? For example, we could define the universe of models so that simultaneous collisions are handled left to right, and then all models in M become deterministic. However, the above argument still works, because even though the above sequence of models will converge to a deterministic model, a second Cauchy sequence where

$$\Delta_{A_i} = -1/i^2$$

will not converge to any model in the set.

Considering the collisions to be simultaneous appears to be problematic. But our set of Newtonian models allows the left collision to occur before the right collision, and also allows the right collision to occur before the left. If time and space are continuums, then these two scenarios must be able to cross one another. At the point of crossing, in this set of models, the collisions are simultaneous. To avoid this simultaneity, we need to either introduce a "hole" in the set of models (hence the incompleteness) or take more drastic measures such as rejecting the time or space continuum. These more drastic measures amount to rejecting Newtonian physics. Hence, from principles of logic, we must either accept that determinism is incomplete or reject Newtonian physics.

An alternative is to reject *discreteness*. However, this does not solve the problem. Consider the stiff-spring models illustrated in Figure 18. In this stiff-spring model, every model is deterministic. But again, we can construct a Cauchy sequence of models of this type that does not converge to a deterministic model. Just let the spring stiffness approach infinity while simultaneously letting the time between collisions approach zero. The same arguments as above apply.

Generalizing this result to arbitrary sets of models and unbounded time horizons appears treacherous. Nevertheless, the above argument shows definitively the incompleteness of determinism for sets of models rich enough to encompass Newton's laws and discrete elastic collisions. It would probably be wise to assume determinism is incomplete for any modeling framework that is rich enough to help design and understand cyber-physical systems, where discrete and continuous behaviors inevitably mix. As a practical matter, it means that nondeterminism is unavoidable. I caution the reader, however, to not use this argument to

dispense with deterministic models altogether. Death is also unavoidable, but that doesn't mean we should give up on life.

5. CONCLUSIONS

Science and engineering use models in different ways, and the differences affect the choice of models and how they are used. For engineering, simplicity and understandability of the models is essential for their usefulness, and the goal becomes to create physical systems that are faithful to the model. In science, the physical systems are given, and simplicity may need to be sacrificed to achieve fidelity.

To achieve simplicity and understandability, clear, deterministic modeling semantics have proven extremely valuable in the past in modeling and design of both physical systems and cyber (computational) systems. But when these modeling frameworks are combined to engineer cyber-physical systems, determinism is usually lost. To improve this situation, we face a number of challenges. One is that widely used modeling techniques on the cyber side abstract away time and cannot directly deal with the continuums of the physical world. Another is that widely used modeling techniques on the physical side do not deal well with discreteness.

Although I assert that deterministic models are useful, I am *not* asserting that nondeterministic models are not useful. They are. One common use of nondeterministic models is to provide simpler abstractions of deterministic models. In this case, the target of the nondeterministic model is a deterministic model. If the abstraction is sound, then statements about the nondeterministic model are also true of the deterministic model. Since the nondeterministic model may be much smaller (fewer states, for example), algorithmic techniques such as model checking may be more effective.

Recognizing the differing emphasis in modeling for engineering purposes, we can make a number of improvements in engineering practice. One is to embrace discrete behaviors, even when modeling physical systems. Another is to use modal models, ensuring that a model is used only within its regime of validity. Even with such improvements, however, there are fundamental limits that can restrict the utility of models. One is chaos, which limits the use of models for prediction, even if the model is deterministic. Another is that discrete behaviors can introduce Zeno conditions and non-constructive models. In the latter case, models may of necessity become nondeterministic, again limiting their predictive power. And finally, we are forced to accept that any set of deterministic models rich enough to model Newton's laws is incomplete, in that the set does not include all its limit points. We will always be faced with possible corner cases that yield nondeterministic models.

ACKNOWLEDGMENTS

I thank Eleftherios Matsikoudis for first suggesting to me that determinism might be incomplete in models and both Matsikoudis and Gil Lederman for helpful comments on a draft of the paper. I also thank Chris Paredis and Yaakov Bar-Shalom for suggestions that have (hopefully) improved this paper. Finally, I thank three anonymous reviewers for very helpful comments that led to significant improvements in the paper. Any remaining errors and opinions are entirely the responsibility of this author.

REFERENCES

Gerard Berry. 1999. *The Constructive Semantics of Pure Esterel - Draft Version 3*. Book Draft. http://www-sop.inria.fr/meije/esterel/doc/main-papers.html

George E. P. Box and Norman R. Draper. 1987. *Empirical Model-Building and Response Surfaces*. Wiley.

David Broman, Lev Greenberg, Edward A. Lee, Michael Masin, Stavros Tripakis, and Michael Wetter. 2015. Requirements for Hybrid Cosimulation Standards. In *Hybrid Systems: Computation and Control (HSCC)*.

François E. Cellier and Ernesto Kofman. 2006. *Continuous System Simulation*. Springer.

Thomas Huining Feng, Edward A. Lee, Xiaojun Liu, Christian Motika, Reinhard von Hanxleden, and Haiyang Zheng. 2014. *Finite State Machines*. Ptolemy.org, Berkeley, CA. http://ptolemy.org/books/Systems

Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, and Matteo Rossi. 2010. Modeling Time in Computing: A Taxonomy and a Comparative Survey. *Computing Surveys* 42, 2 (2010), 6:1–6:59.

Solomon Wolf Golomb. 1971. Mathematical models: Uses and limitations. *IEEE Transactions on Reliability* R-20, 3 (1971), 130–131. DOI:http://dx.doi.org/10.1109/TR.1971.5216113

Gilles Kahn and D. B. MacQueen. 1977. Coroutines and Networks of Parallel Processes. In *Information Processing*, B. Gilchrist (Ed.). North-Holland Publishing Co., 993–998.

Edward A. Lee. 1999. Modeling Concurrent Real-time Processes Using Discrete Events. *Annals of Software Engineering* 7 (1999), 25–45. DOI:http://dx.doi.org/10.1023/A:1018998524196

Edward A. Lee. 2014. Constructive Models of Discrete and Continuous Physical Phenomena. *IEEE Access* 2, 1 (2014), 1–25. DOI:http://dx.doi.org/10.1109/ACCESS.2014.2345759

Edward A. Lee. 2015. The Past, Present, and Future of Cyber-Physical Systems: A Focus on Models. *Sensors* 15, 3 (2015), 4837–4869. DOI:http://dx.doi.org/10.3390/s150304837

Edward A. Lee, Mehrdad Niknami, Thierry S. Nouidui, and Michael Wetter. 2015. Modeling and Simulating Cyber-Physical Systems using CyPhySim. In *International Conference on Embedded Software (EMSOFT)*. ACM, Amsterdam, The Netherlands.

Edward A. Lee and Alberto Sangiovanni-Vincentelli. 1998. A Framework for Comparing Models of Computation. *IEEE Transactions on Computer-Aided Design of Circuits and Systems* 17, 12 (1998), 1217–1229.

Edward A. Lee and Stavros Tripakis. 2010. Modal Models in Ptolemy. In *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT)*, Vol. 47. Linköping University Electronic Press, Linköping University, Oslo, Norway, 11–21. http://chess.eecs.berkeley.edu/pubs/700.html

Edward A. Lee and Haiyang Zheng. 2007. Leveraging Synchronous Language Principles for Heterogeneous Modeling and Design of Embedded Systems. In *EMSOFT*. ACM, Salzburg, Austria, 114 – 123. DOI:http://dx.doi.org/10.1145/1289927.1289949

Eleftherios Matsikoudis and Edward A. Lee. 2013. On Fixed Points of Strictly Causal Functions. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, Vol. LNCS 8053. Springer-Verlag, Buenos Aires, Argentina, 183–197. DOI:http://dx.doi.org/10.1007/978-3-642-40229-6_13

Michael Mendler, Thomas R. Shiple, and Gérard Berry. 2012. Constructive Boolean circuits and the exactness of timed ternary simulation. *Formal Methods in System Design* 40, 3 (2012), 283–329. DOI:http://dx.doi.org/10.1007/s10703-012-0144-6

Modelica Association. 2014. *Functional Mock-up Interface for Model Exchange and Co-Simulation*. Report Version 2.0. https://www.fmi-standard.org/downloads

Pieter J. Mosterman and Gautam Biswas. 1995. Modeling Discontinuous Behavior with Hybrid Bond Graphs. In *Ninth Qualitative Reasoning Workshop*. Amsterdam, 139–147.

John D. Norton. 2007. Causation as Folk Science. In *Causation, Physics, and the Constitution of Reality*, Huw Price and Richard Corry (Eds.). Clarendon Press, Oxford, 11–44.

Martin Otter, Hilding Elmqvist, and José Daz López. 2005. Collision Handling for the Modelica MultiBody Library. In *Modelica Conference*. Hamburg, Germany, 45–53. http://elib.dlr.de/12299/1/otter2005-modelica-collision.pdf

Constantinos C. Pantelides. 1988. The Consistent Initialization of Differential-Algebraic Systems. *SIAM Journal of Scientific and Statistical Computing* 9, 2 (1988), 213–231. DOI:http://dx.doi.org/10.1137/0909014

Huw Price and Richard Corry (Eds.). 2007. *Causation, Physics, and the Constitution of Reality*. Clarendon Press, Oxford.

Claudius Ptolemaeus (Ed.). 2014. *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, Berkeley, CA. http://ptolemy.org/books/Systems

David E. Stewart. 2000. Rigid-Body Dynamics with Friction and Impact. *SIAM Rev.* 42, 1 (2000), 3–39. DOI:http://dx.doi.org/10.1137/S0036144599360110

Lothar Thiele and Pratyush Kumar. 2015. Can Real-Time Systems be Chaotic?. In *EMSOFT*. ACM, Amsterdam, The Netherlands, 21–30.

Michael M. Tiller. 2001. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers.

Haiyang Zheng, Edward A. Lee, and Aaron D. Ames. 2006. Beyond Zeno: Get on with It!. In *Hybrid Systems, Computation, and Control (HSCC)*, J Hespanha and A. Tiwari (Eds.), Vol. LNCS 3927. Springer-Verlag, 568–582. DOI:http://dx.doi.org/10.1007/11730637_42