Models of Time for a Computational World

Edward A. Lee[*]
[*]University of California, Berkeley, eal@eecs.berkeley.edu

Humans have struggled for centuries to develop a good notion of time. For example, the uniformly advancing fabric of time that Newton used to describe the dynamics of physical systems was shattered by Einstein, whose model allows time to advance at different relative rates, depending on the frame of reference. From a practical perspective, measurement of time has historically been the measurement of the position of a point on the surface of the earth relative to the sun and other celestial bodies. Indeed, this association makes measuring time extremely valuable, because of the ability of the solve the inverse problem; if you know the current time and the positions of the celestial bodies, then you know your position on the surface of the earth [6]. In all of these struggles, the *correctness* of any notion of time has been judged by its relation to observable physical dynamic systems.

The world of computation, by contrast, is deeply formal, and depends far less on a relation to observable physical dynamic systems. A formal world is rooted in symbols and axioms (statements that are "true" by definition). In the most useful formal systems (including all computational systems), the axioms provide "truth preserving" operations that turn one "true" statement into another "true" statement. The "truth" of a statement in a formal world depends on whether it is consistent with the axioms. The *provability* of a statement depends on whether there is a terminating sequence of truth-preserving transformations that reduces the statement to a set of axioms. The formal worlds in mathematics and in computing have deep parallels, for example in the relation between provability and computability, as developed by Turing and Church.

Working in a formal world has a distinct advantage over working in the physical world. By constructing axioms and proofs, we can make statements that are definitively "true," invulnerable to any further developments of understanding. The *usefulness* of these statements may change depending on their relationship to physical systems, but not their truth. In this position paper, I will argue that models of time in computational systems should be viewed primarily as formal models, and not be judged primarily by their relationship to physical dynamic systems. That is, before we attempt to deliver "real-time systems" that are *useful*, we must first deliver "real-time systems" that are *true*. Only if they can be judged to be true should we evaluate their usefulness.

This work was supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CSR-EHS: PRET), #0931843 (ActionWebs), and #1035672 (CSR-CPS Ptides)), the U. S. Army Research Laboratory (ARL #W911NF-11-2-0038), the Air Force Research Lab (AFRL), the Multiscale Systems Center (MuSyC), one of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program, and the following companies: Bosch, National Instruments, Thales, and Toyota.

This perspective contrasts with the dominant view in the world of "real-time computing," which is primarily judged by its ability to deliver computational actions in relation to physical clocks whose task is primarily to measure the position of a point on the surface of the earth relative to the celestial bodies. This world is dominated by scheduling theory, and has developed many sophisticated and useful techniques [13]. But these techniques do not deliver truth in an absolute sense.

Thinking of time as a logical and formal notion in computational systems rather than a physical measurement was eloquently posited by Lamport [9]. This work put emphasis on consistent views of the state of a system rather than on the accuracy of the measurement of time. Lamport introduced the idea that consistency could be guaranteed (it could be "true") in certain circumstances, whereas measurements are never true. He also emphasized the *partial* ordering of events, rather than their total ordering, a concept that was then extended in very interesting ways by many researchers (see [2], [1]). Purely logical notions of time become more *useful* (not more correct) when one can establish a connection with physical measurement while preserving truth, as done in the field of hybrid systems, for example [7].

I have previously argued that time needs become part of the *semantics* of programs, rather than being merely a measure of performance [10]. Here, I discuss the form that models of time should take in the formal systems of distributed programs. Specifically, I address the issues of precision, causality, simultaneity, and multiform time.

I. Numerical Representation

In any practical computational system, time must be represented by members of a discrete set. It is naive to simply assert that a time value is a real number, for example, since with any fixed digital encoding, a randomly chosen real number is not representable with probability one. We could just fall back on what is done to represent physical quantities, where the state-of-the-art is IEEE double precision floating point numbers. This is a reasonable choice if time is a measurement, but it is not reasonable within a formal system, for two reasons.

First, the *precision* with which time is represented by a floating-point number depends on its magnitude. Define the precision p of a time t_1 to be $p = t_2 - t_1$, where t_2 is the next larger representable time. With floating point numbers, p gets larger as t_1 gets larger. Second, floating-point time makes the notion of *simultaneity* complex. With floating-point numbers, given two time values t_1 and t_2, we may have that $t_1+t_2=t_1$. This would suggest that an event that occurs at time t_1+t_2 is simultaneous with an event that occurs at time t_1.

Taken together, these problems can result in a caused event being simultaneous with the causing event. Suppose the caused event occurs at time t_1+t_2 and is caused by an event occurring at time t_1. Then as long as t_1 remains small, the time of the caused event will differ from the time of the causing event. But when t_1 gets large enough, they become simultaneous.

The time representation used in the IEEE 1588 standard [4] does not suffer from this problem, but it does suffer from a "Y-2k" problem. There are a finite number of representable times. The model of time used in Ptolemy II solves this by having an infinite number of representable times, at the cost that the memory required to store a time value is not bounded [12]. These are practical tradeoffs that can be evaluated, and such evaluation can work entirely within a formal system that preserves "truth" during any manipulations of time.

II. Causality, Simultaneity, and Superdense Time

Intuitively, we think of an event that is caused by another as occurring later.[1] But how much later? If you push one end of a rigid rod, how much later does the other end move? A useful conceptual model must admit the possibility that the caused event is simultaneous with the causing event. Fortunately, this possibility is allowed by a *superdense* notion of time, introduced by Manna and Pnueli [14]. A superdense time value is a two-tuple, $t=(t_1,n)$, where t_1 is metric time and n is an index. Manna and Pnueli proposed that t_1 be real, but given the argument in the previous section, we would prefer an IEEE 1588 or Ptolemy II representation. The number n is a natural number, where again it can be practically represented by either a finite set (e.g. the set of 32 bit ints) or an infinite set (e.g. the Java BigNum), with evaluable tradeoffs in expressiveness and memory requirements. Superdense time easily handles the situation where caused events are simultaneous with causing events by having the caused event occur at (t_1,n_1) and the causing event occur at (t_1,n_2), where $n_1>n_2$. This solution is used in the VHDL language for hardware description, and can be further refined to become more modular.

III. Multiform Time

To be useful, even a formal model of time often needs to have some correspondence with physical measurements of time. Distributed systems accomplish this using a multiplicity of clocks, each measuring time at a different physical location. How can these measurements of time coexist with a formal notion of time? This problem is addressed in the PTIDES project [3], where a strictly formal notion of time is bound to physical measurements of time at sensors and actuators.

[1]In [15], Price and Corry argue that this notion of causality has no basis in physics. It is a human construct. Fortunately, synchronous fixed-point semantics and equational languages such as Modelica [5] admit this possibility.

Physical measurements of time, however, diverge. Time synchronization technologies help [4], [8], but they are imperfect, which can compromise preservation of "truth" in a formal system. The PTIDES project resolves this problem by showing that if the absolute discrepancy between two clocks can be bounded, then formal operations on time become possible. The notion of an "absolute discrepancy" is a problematic one, since it requires itself a physical notion of simultaneity. You have to be able to simultaneously observe the values of two physically separate physical clocks, something that is not physically possible. But it can be approximated, and given such an approximation, PTIDES provides a formal and deterministic notion of time that allows for clocks to progress at different rates, as long as their divergence can be bounded.

Another version of multiform time is given in the modal models of Ptolemy II [11], where parts of the system can go into "suspended animation," where logical time stops progressing altogether. The passage of time is, in fact, a changing state of a system. If a subsystem is inactive, its state should not change. Ptolemy II accomplishes this in a strictly formal and deterministic way.

I therefore argue that we can and must extend our formal notions of computation with a formal notion of time. The result will be not merely *useful*, but also *correct*.

References

[1] R. Alur and T. Henzinger. Logics and models of real time: A survey. In J. W. De Bakker, C. Huizing, W. P. De Roever, and G. Rozenberg, editors, *REX Workshop*, volume LNCS 600, pages 74–106, Mook, The Netherlands, 1991. Springer.

[2] M. Broy. Refinement of time. *Theoretical Computer Science*, 253(1):3–26, 2001.

[3] J. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou. Distributed real-time software for cyber-physical systems. *Proceedings of the IEEE (special issue on CPS)*, 100(1):45–59, 2012.

[4] J. C. Eidson. *Measurement, Control, and Communication Using IEEE 1588*. Springer, 2006.

[5] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley, 2003.

[6] P. Galison. *Einstein's Clocks, Poincaré's Maps*. W. W. Norton & Company, New York, 2003.

[7] T. A. Henzinger. The theory of hybrid automata. In M. Inan and R. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series F: Computer and Systems Sciences*, pages 265–292. Springer-Verlag, 2000.

[8] S. Johannessen. Time synchronization in a local area network. *IEEE Control Systems Magazine*, pages 61–69, 2004.

[9] L. Lamport, R. Shostak, and M. Pease. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[10] E. A. Lee. Computing needs time. *Communications of the ACM*, 52(5):70–79, 2009.

[11] E. A. Lee and S. Tripakis. Modal models in Ptolemy. In *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT)*, volume 47, pages 11–21, Oslo, Norway, 2010. Linköping University Electronic Press, Linköping University.

[12] E. A. Lee and H. Zheng. Operational semantics of hybrid systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control (HSCC)*, volume LNCS 3414, pages 25–53, Zurich, Switzerland, 2005. Springer-Verlag.

[13] J. W. S. Liu. *Real-Time Systems*. Prentice-Hall, 2000.

[14] Z. Manna and A. Pnueli. Verifying hybrid systems. In *Hybrid Systems*, volume LNCS 736, pages 4–35, 1993.

[15] H. Price and R. Corry, editors. *Causation, Physics, and the Constitution of Reality*. Clarendon Press, Oxford, 2007.