A New System Science in Research and Education

Presented by
Edward A. Lee
Chess, UC Berkeley

A Traditional Systems Science – Feedback Control Systems

• Models of continuous-time dynamics
• Sophisticated stability analysis
• But not accurate for software controllers
Discretized Model – A Step Towards Software

- Numerical integration techniques provided sophisticated ways to get from the continuous idealizations to computable algorithms.
- Discrete-time signal processing techniques offer the same sophisticated stability analysis as continuous-time methods.
- But it’s still not accurate for software controllers

**In general,** $z$ is an $N$-tuple, $z = (z_1, \ldots, z_N)$, where $z_2: \text{Reals} \rightarrow \text{Reals}$. The derivative of an $N$-tuple is simply the $N$-tuple of derivatives, $\dot{z} = (\dot{z}_1, \ldots, \dot{z}_N)$. We know from calculus that

$$\frac{dz(t)}{dt} = \lim_{\delta \to 0} \frac{z(t + \delta) - z(t)}{\delta}$$

and so, if $\delta > 0$ is a small number, we can approximate this derivative by

$$\frac{dz(t)}{dt} \approx \frac{z(t + \delta) - z(t)}{\delta}$$

Using this for the derivative in the left-hand side of (5.50) we get

$$z(t + \delta) - z(t) = \delta (z(t), \dot{z}(t)). \quad (5.51)$$

---

Hybrid Systems – Reconciliation of Continuous & Discrete

- UCB researchers have contributed hugely to the theory and practice of blended discrete & continuous models.
- But it’s still not accurate for software controllers

This model gives two separate ordinary differential equations, one for each point mass attached to a spring. The ZeroCrossingDetector does the event detection of the point masses and emits the “touched” event.
Timing in Software is More Complex Than What the Theory Deals With

An example, due to Jie Liu, models two controllers sharing a CPU under an RTOS. Under preemptive multitasking, only one can be made stable (depending on the relative priorities). Under non-preemptive multitasking, both can be made stable.

Where is the theory for this?

Another Traditional Systems Science - Computation, Languages, and Semantics

Everything “computable” can be given by a terminating sequential program.

- Functions on bit patterns
- Time is irrelevant
- Non-terminating programs are defective

\[ f : \text{States} \rightarrow \text{States} \]

States = Bits\(^*\)

sequence

results + state out
Current fashion – Pay Attention to “Non-functional properties”

• Time
• Security
• Fault tolerance
• Power consumption
• Memory management

But the formulation of the question is very telling:

How is it that when a braking system applies the brakes is any less a function of the braking system than how much braking it applies?

Processes and Process Calculi

Infinite sequences of state transformations are called “processes” or “threads”

Various messaging protocols lead to various formalisms.

In prevailing software practice, processes are sequences of external interactions (total orders).

And messaging protocols are combined in ad hoc ways.

Berkeley, Vanderbilt, Memphis 7

Berkeley, Vanderbilt, Memphis 8
Software realizing these interactions is written at a very low level (semaphores and mutexes). Very hard to get it right.

An aggregation of processes is not a process (a total order of external interactions). What is it?

Many software failures are due to this ill-defined composition.
Promising Alternatives

- Synchronous languages (e.g. Esterel)
- Time-driven languages (e.g. Giotto)
- Hybrid systems
- Timed process networks
- Discrete-event formalisms
- Timed CSP

We are working on interface theories and meta models that express dynamic properties of components, including timing.

Intellectual Groupings in EECS
Berkeley has required sophomore course that addresses mathematical modeling of signals and systems from a computational perspective.

The web page at the right illustrates a broad view of feedback, where the behavior is a fixed point solution to a set of equations. This view covers both traditional continuous feedback and discrete-event systems.

Themes of the Course

• The connection between imperative and declarative descriptions of signals and systems.

• The use of sets and functions as a universal language for declarative descriptions of signals and systems.

• State machines and frequency domain analysis as complementary tools for designing and analyzing signals and systems.

• Early and often discussion of applications.
Conclusion

We are on the line to build a new system science that is at once physical and computational.

It will form the foundation for our understanding of computational systems that engage the physical world.

And it will change how we teach and research the engineering of systems.