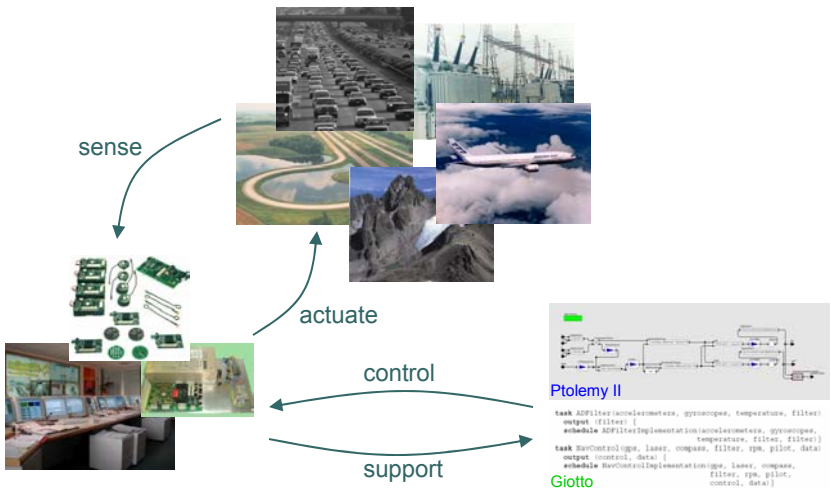
Future Directions

Edward A. Lee

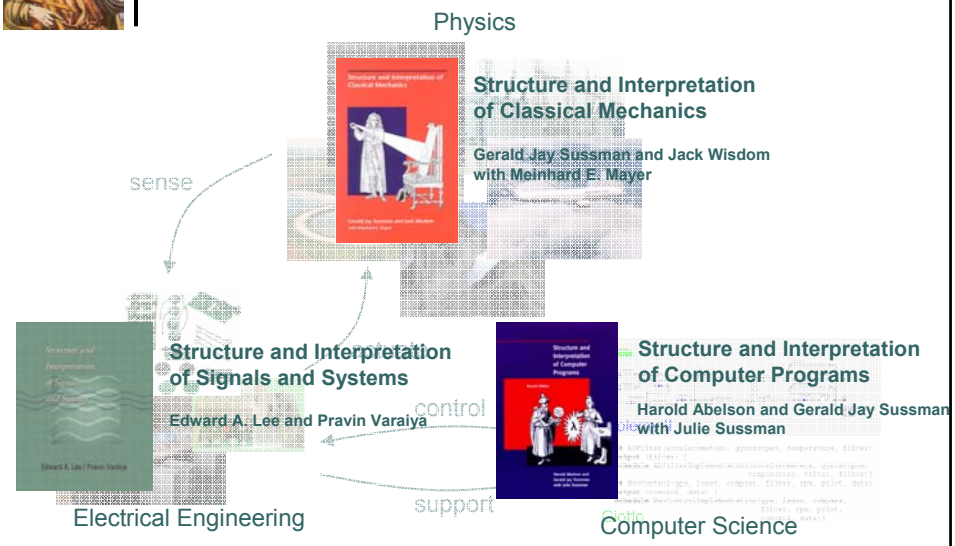
6th Biennial Ptolemy Miniconference

Berkeley, CA
May 12, 2005

A New Computational Platform: Ubiquitous Networked Embedded Systems



A Developing Paradigm: Convergence of Computation and the Physical World



This connection is due to Xiaojun Liu.

Structure and Interpretation of Computer Programs

Preface to the First Edition:

> Underlying our approach to this subject is our conviction that "computer science" is not a science and that its significance has little to do with computers. The computer revolution is a revolution in the way we think and in the way we express what we think. The essence of this change is the emergence of what might best be called *procedural epistemology* – the study of the structure of knowledge from an imperative point of view, as opposed to the more declarative point of view taken by classical mathematical subjects. Mathematics provides a framework for dealing precisely with notions of "what is." Computation provides a framework for dealing precisely with notions of "how to."

Structure and Interpretation of Computer Programs

Preface to the Second Edition:

> This edition emphasizes several new themes. The most important of these is the central role played by different approaches to dealing with time in computational models: objects with state, concurrent programming, functional programming, lazy evaluation, and nondeterministic programming. We have included new sections on concurrency and nondeterminism, and we have tried to integrate this theme throughout the book.

Structure and Interpretation of Computer Programs

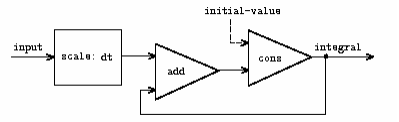


Figure 3.32: The `integral` procedure viewed as a signal-processing system.

Structure and Interpretation of Classical Mechanics

Preface:

> Computational algorithms are used to communicate precisely some of the methods used in the analysis of dynamical phenomena. … Computation requires us to be precise about the representation of mechanical and geometric notions as computational objects and permits us to represent explicitly the algorithms for manipulating these objects.
>
> This book presents classical mechanics from an unusual perspective. … It uses functional mathematical notation that allows precise understanding of fundamental properties of classical mechanics. It uses computation to constrain notation, to capture and formalize methods, for simulation, and for symbolic analysis.

Structure and Interpretation of Classical Mechanics

1.4 Computing Actions

The Lagrangian for a free particle moving in three dimensions:

$$L(t, x, v) = m(v \cdot v)/2, \qquad (1.14)$$

As a procedure:

```
(define ((L-free-particle mass) local)
  (let ((v (velocity local)))
    (* 1/2 mass (dot-product v v))))
```

Structure and Interpretation of Signals and Systems

Preface – Approach

This book is about mathematical modeling and analysis of signals and systems, applications of these methods, and the connection between mathematical models and computational realizations. We develop three themes.

The first theme is the use of sets and functions as a universal language to describe diverse signals and systems.

The second theme is that complex systems are constructed by connecting simpler subsystems in standard ways – cascade, parallel, and feedback.

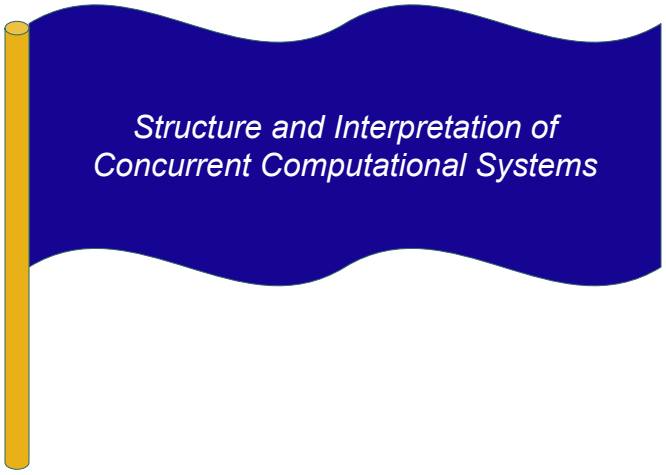
Our third theme is to relate the declarative view (mathematical, "what is") with the imperative view (procedural, "how to").

Putting these side by side …

Data Structures Array, List, Tree, Stream, …	*Algorithms* Sort, Map, Traverse, Filter, …
Physical Phenomena Force, Electromagnetic Field, …	*Physical Processes* $F = ma$, Maxwell's Equations, …
Signals Audio, Image, …	*Systems* Filter, Compress, …

This connection is due to Xiaojun Liu.

A Banner for the Ptolemy Project



Our Current Projects

- Abstract semantics (Cataldo, Liu, Matsikoudis, Zheng)
  - Domain polymorphism
  - Actor semantics (prefire, fire, postfire)
  - Compositional directors
  - Time semantics and backtracking
- Distributed computing (Feng, Zhao)
  - Robust distributed consensus
  - Data coherence (distributed caches)
  - Time synchronization
  - Stochastic models
- Real-time software (Cheong, Zhou, Zhou)
  - Time-based models vs. dataflow models
  - Deterministic, understandable multitasking
  - Aspect-oriented multi-view modeling
  - Code generation

Future Project Proposal: Adaptive Networked Infrastructure

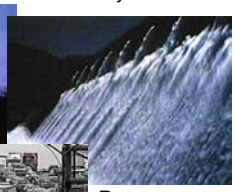
Core partners: Berkeley (lead), Cornell, Vanderbilt

Outreach partners: San Jose State, Tennessee Tech, UC Davis, UC Merced.

Principal investigator: Edward A. Lee

Enabling technologies: wireless networked embedded systems with sensors and actuators

Approach: Engineering methods for integrating computer-controlled, networked sensors and actuators in societal-scale infrastructure systems.



Resource management test beds:

- electric power
- transportation
- water

The ANI ERC

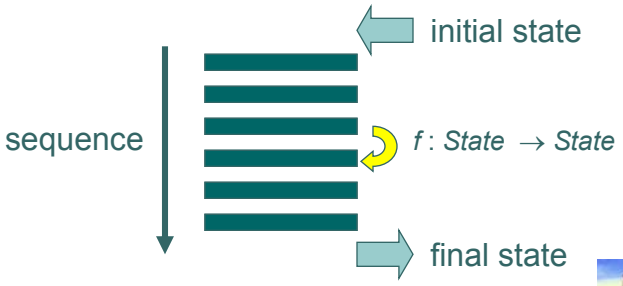
Deliverables: Engineering Methods, Models, and Toolkits for:

- design and analysis of systems with embedded computing
- computation integrated with the physical world
- analysis of control dynamics with software and network behavior
- programming the ensemble, not the computer
- computer-integrated systems oriented engineering curricula

Closing the Loop: The Key Issues

- Time
- Concurrency

20-th Century Computing Abstraction



- Time is irrelevant
- All actions are ordered
- Nontermination is a defect
- Concurrency is an illusion

Computation

$$f: \{0,1\}^{\omega} \rightarrow \{0,1\}^{\omega}$$

Everything Else is "Non-functional"

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

But the word choice is 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?

Exploiting the 20-th Century Computation Abstraction

- Programming languages
- Virtual memory
- Caches
- Dynamic dispatch
- Speculative execution
- Memory management (garbage collection)
- Multitasking (threads and processes)
- Networking (TCP)
- Theory (complexity)

APOT

The question: What would have to change to achieve *absolutely, positively on time* (APOT)?

The answer: *nearly everything.*

What to do?

- Put time into programming languages
  - *Promising start*: Simulink, Giotto, DE domain, TM domain
- Rethink the OS/PL split
  - *Promising start*: TinyOS/nesC, VIPTOS
- Rethink the hardware/software split
  - *Promising start*: FPGAs with programmable cores + SDF/HDF
- Memory hierarchy with predictability
  - *Promising start*: Scratchpad memories vs. caches + SDF/HDF
- Memory management with predictability
  - *Promising start*: Bounded pause time garbage collection
- Predictable, controllable deep pipelines
  - *Promising start*: Pipeline interleaving + SDF/HDF
- Predictable, controllable, understandable concurrency
  - *Promising start*: Synchronous languages, SR domain
- Networks with timing
  - *Promising start*: Time triggered architectures, time synchronization
- Computational dynamical systems theory
  - *Promising start*: Hybrid systems

Conclusion

The time is right to create the 21-st century theory of (embedded) computing.