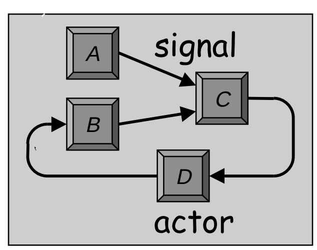
Models of Computation
for Embedded, Real-Time Systems

Tower of Babel, in painting by Bruegel, 1563

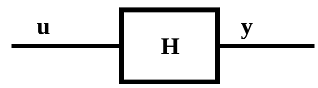
Edward A. Lee
Professor
UC Berkeley

Actors and Signals

- Actors represent functionality
  - May have inputs and outputs, or not
  - May be implemented concurrently, or not
  - Are conceptually concurrent
- Signals represent shared information
  - Shared variables
  - Functions of time
  - Sequences of tokens
  - Events in time



Specifying Actors



- Denotationally:
  - A relation between signals (constraints on acceptable signals)
  - A function mapping input signals to output signals
  - A set of constraints
  - e.g.

$$Y(z) = H(z)U(z)$$

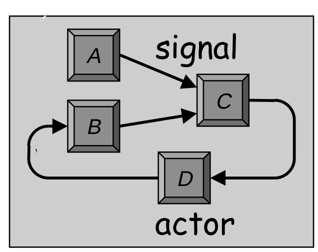
- Operationally:
  - Given observations of some signals, how do we change others?
  - e.g.

$$\vec{x}(n+1) = A\vec{x}(n) + bu(n)$$
$$y(n) = c^T\vec{x}(n) + du(n)$$

Semantics

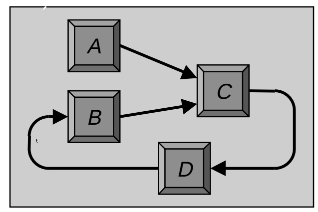
The meaning of an interconnection of actors

- Denotational semantics:
  - The set of properties that signals must have in a particular interconnection
- Operational semantics:
  - How to compute the signal values for an interconnection



Useful Concurrent Semantics

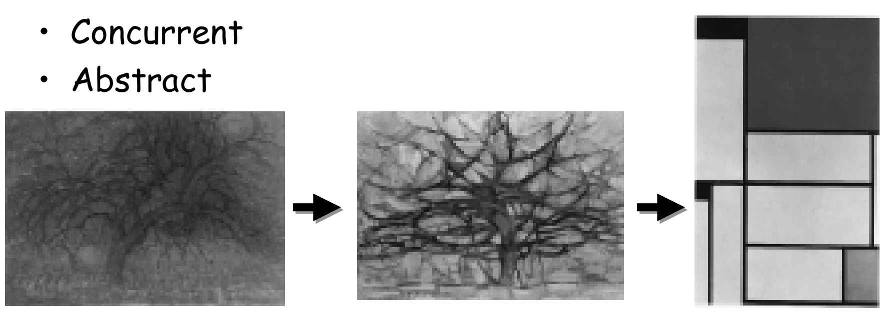
- Analog computers (ODEs)
- Spatial/temporal models (PDEs)
- Discrete time (difference equations)
- Discrete-event systems (DE)
- Synchronous-reactive systems (SR)
- Sequential processes with rendezvous (CSP)
- Process networks (Kahn)
- Dataflow (Dennis)



Block diagrams often provide a nice syntax for concurrent semantics

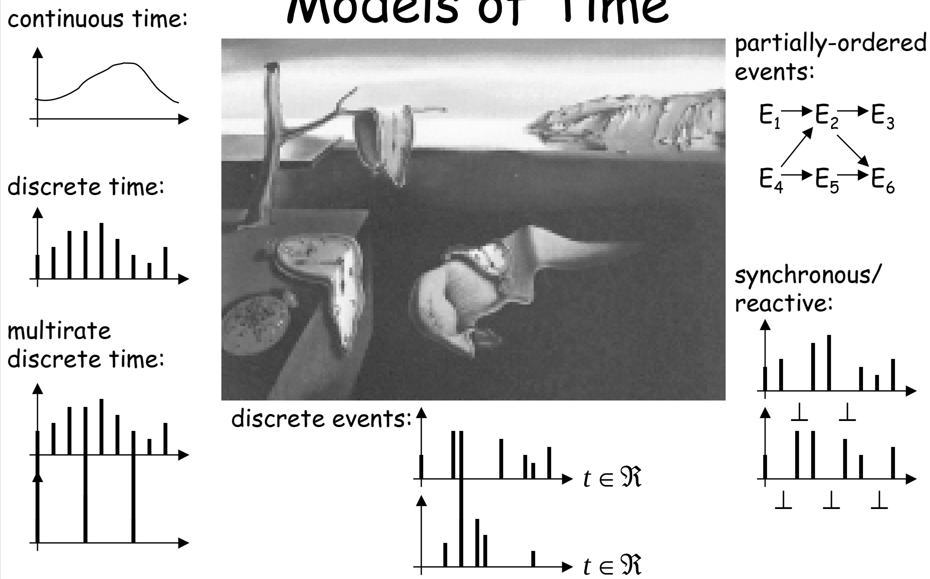
Shared Properties

- Strengths and weaknesses (no silver bullet)
- Many are domain-specific
- Modular (some are compositional)
- Amenable to visual syntaxes
- Hierarchical
- Concurrent
- Abstract

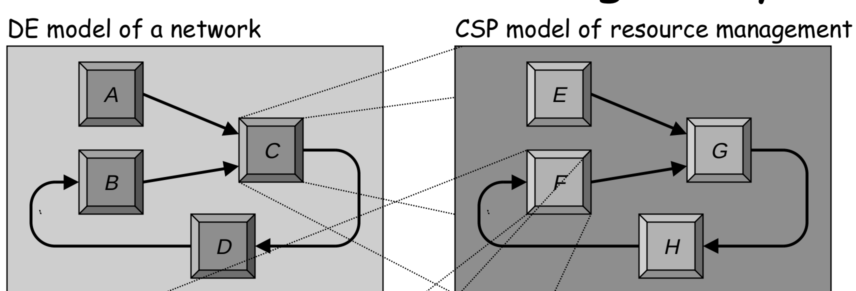


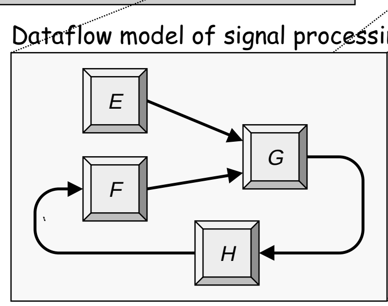
Piet Mondrian, Red Tree (1908), Gray Tree (1912), Tableau I (1921)

Essential Differences - Models of Time



Hierarchical Heterogeneity



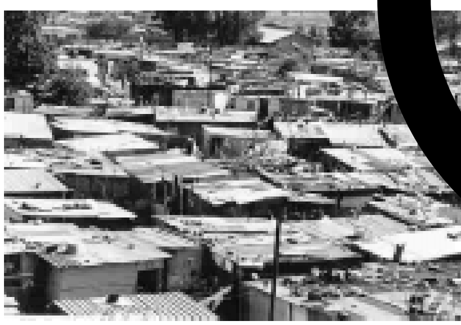


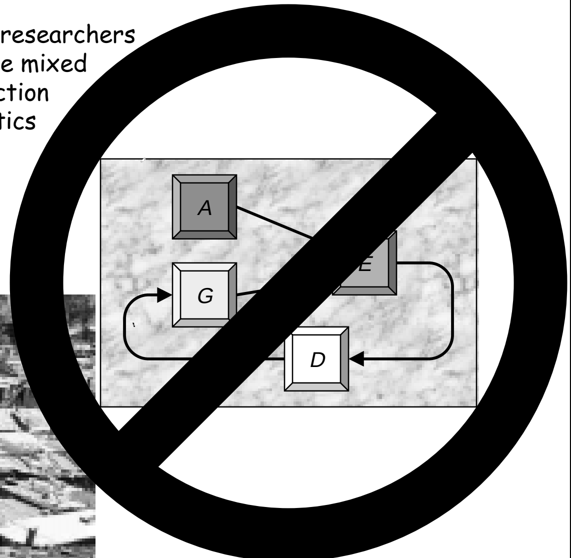
Methods exist for hierarchically composing some, but not all models of computation.

Amorphous Heterogeneity

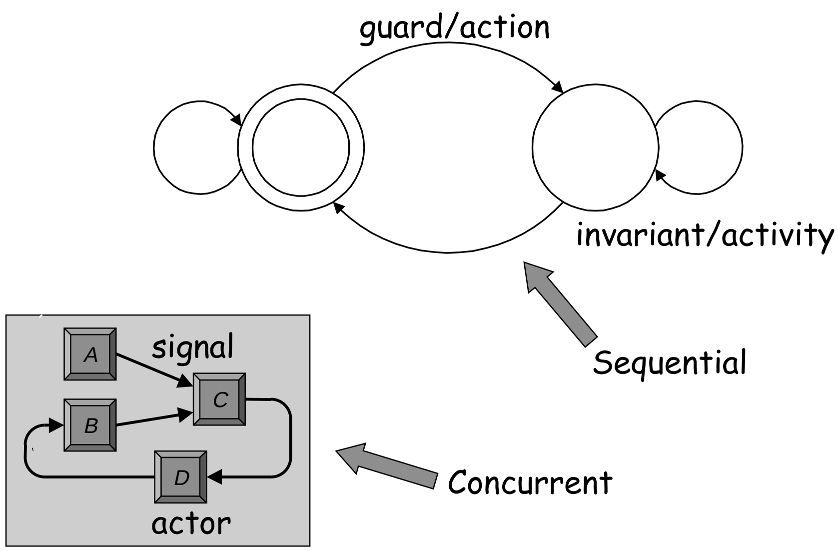
A few researchers propose mixed interaction semantics

But it becomes very hard to figure out what is going on.

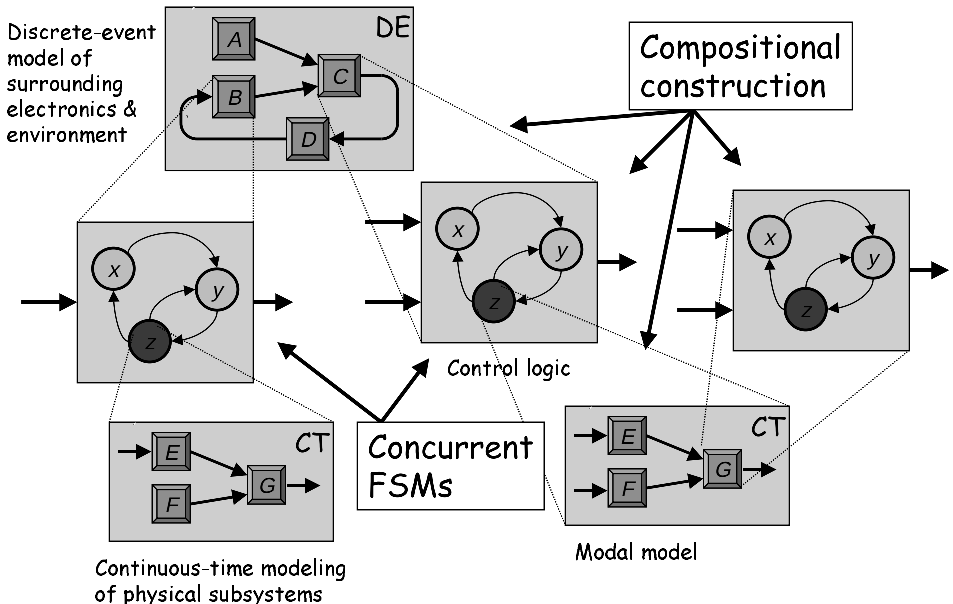




State Machines & Block Diagrams



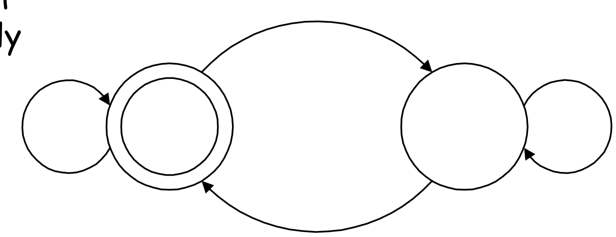
Concurrency + Control Logic



Useful State Machine Models

- Von-Neumann computers
- Imperative programming languages
- Finite state machines (FSMs)

All of these can be hierarchically combined with each of the concurrency models below.



Codesign?

- Software roots:
  - sequential
  - imperative
  - terminating
  - not real time
- Hardware roots:
  - parallel
  - declarative
  - non-terminating
  - real time
- Modern design:
  - concurrent
  - imperative & declarative
  - often non-terminating
  - real time & not real time

Key Semantic Issues

- Does a composition of actors have a behavior? More than one behavior?
  - Typical hard case: feedback systems
- Can a simulation or analysis strategy find a behavior? All behaviors? A subset of behaviors satisfying some property?
  - The "strategy" is an operational semantics.
- Does a composition have the same semantics as an actor?
  - "compositionality" property.

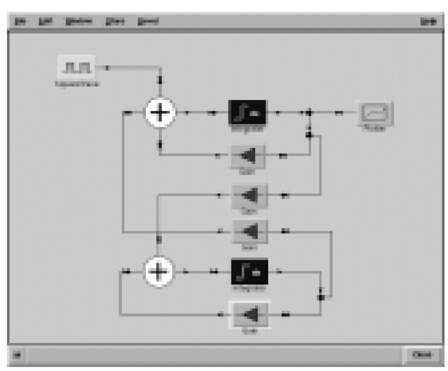
Key Practical Issues

- Can it be simulated?
  - Bounded memory
  - Bounded time for at least a partial solution
  - Simulation speed
- Can it be implemented?
  - Bounded memory
  - Bounded time for at least a partial solution
  - Synthesis algorithms
- How many ways can it be implemented?
  - Software vs. hardware
  - Parallel vs. sequential
  - Scheduling algorithms
  - Avoiding overspecification

1. Analog Computers (Coupled ODEs)

Semantics:

- actors define relations between functions of time (ODEs or algebraic equations)
- fixed point is a set of signals satisfying these relations



Ptolemy II model, designed by Jie Liu

Examples:
Spice, HP ADS, Simulink, Saber, Matrix X, ...

1. Strengths and Weaknesses

Strengths:

- Accurate model for many physical systems
- Determinate under simple conditions
- Established and mature (approximate) simulation techniques

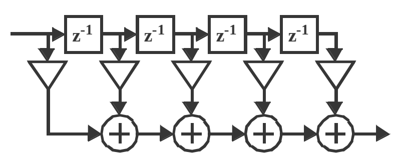
Weaknesses:

- Covers a narrow application domain
- Tightly bound to an implementation
- Relatively expensive to simulate
- Difficult to implement in software

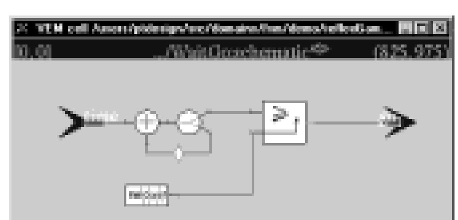
2. Discrete Time

Semantics:

- blocks are relations between functions of discrete time (difference equations)



- fixed point is a set of signals satisfying these relations



Ptolemy 0.7 subsystem, designed by Bilung Lee

Examples:
HP Ptolemy, SystemView, ...

2. Strengths and Weaknesses

Strengths:
- Useful model for embedded DSP
- Determinate under simple conditions
- Easy simulation (cycle-based)
- Easy implementation (circuits or software)

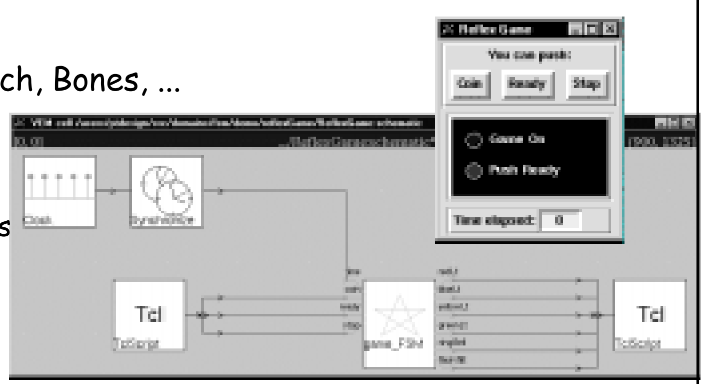
Weaknesses:
- Covers a narrow application domain
- Global synchrony may overspecify some systems

3. Discrete Events

- Events occur at discrete points on a time line that is usually a continuum. The entities react to events in chronological order.

Examples:
SES Workbench, Bones, ...

Application areas:
- networks
- Queueing systems
- Manufacturing
- Hardware design



3. Strengths and Weaknesses

Strengths:
- Natural for asynchronous digital hardware
- Global synchronization
- Determinate under simple conditions
- Simulatable under simple conditions

Weaknesses:
- Expensive to implement in software
- May over-specify and/or over-model systems

Machinery for Studying 1,2, and 3

- The Cantor metric:

$$d(s_1, s_2) = \frac{1}{2^\tau}$$

where τ is the GLB of the times where s_1 and s_2 differ.

- Metric space theorems provide conditions for the existence and uniqueness of fixed points.

Example result: VHDL (a DE language) permits programs where a fixed point exists but no simulator can find it.

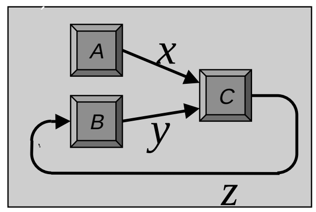
4. Synchronous/Reactive Models

- A discrete model of time progresses as a sequence of “ticks.” At a tick, the signals are defined by a fixed point equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_{A,t}(1) \\ f_{B,t}(z) \\ f_{C,t}(x,y) \end{bmatrix}$$

Examples:
Esterel, Lustre, Signal, Argos, ...

Application areas:
•user interfaces
•anything with elaborate control logic



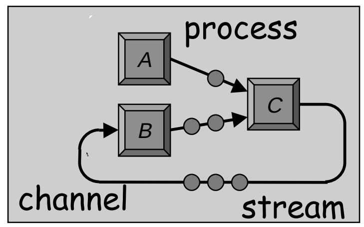
4. Strengths and Weaknesses

Strengths:
- Good match for control-intensive systems
- Tightly synchronized
- Determinate in most cases
- Maps well to hardware and software

Weaknesses:
- Computation-intensive systems are overspecified
- Modularity is compromised
- Causality loops are possible
- Causality loops are hard to detect

5. Process Networks

- Processes are prefix-monotonic functions mapping sequences into sequences.
- One implementation uses blocking reads, non-blocking writes, and unbounded FIFO channels.



Examples:
SDL, Unix pipes, ...

Possible application areas:
•telecommunications systems modeling
•asynchronous, multitasking, reactive systems

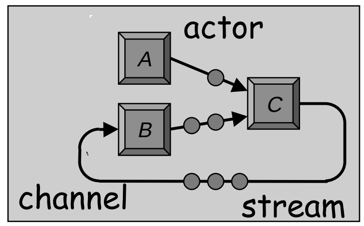
5. Strengths and Weaknesses

Strengths:
- Loose synchronization (distributable)
- Determinate under simple conditions
- Implementable under simple conditions
- Maps easily to threads, but much easier to use
- Turing complete (expressive)

Weaknesses:
- Control-intensive systems are hard to specify
- Bounded resources are undecidable

6. Dataflow

- A special case of process networks where a process is made up of a sequence of firings (finite, atomic computations).



Examples:
SPW, HP Ptolemy, Cossap, ...

Possible application areas:
•signal processing
•computer architecture (dynamic instruction scheduling)
•compilers (an analysis technique)

6. Strengths and Weaknesses

Strengths:

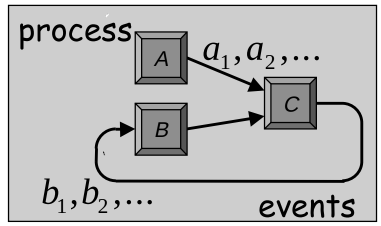
– Good match for signal processing
– Loose synchronization (distributable)
– Determinate under simple conditions
– Special cases map well to hardware and embedded software

Weakness:

– Control-intensive systems are hard to specify

7. Rendezvous Models

- Events represent rendezvous of a sender and a receiver. Communication is unbuffered and instantaneous.



Examples:
CSP, CCS, Occam, Lotos, ...

Possible application areas:
- Client/server systems
- Object-request brokers
- Resource sharing

7. Strengths and Weaknesses

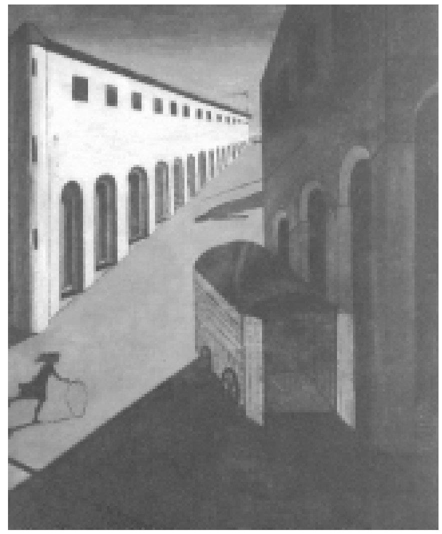
Strengths:
- Models resource sharing well
- Partial-order synchronization (distributable)
- Supports naturally nondeterminate interactions

Weaknesses:
- Oversynchronizes some systems
- Difficult to make determinate (and useful)

Conclusion

Modeling in an
artificial universe

When choosing semantics, we need not be constrained by laws of physics. Designers should reflect on the possibilities, and choose based on usefulness and appropriateness.



Giorgio de Chirico, Melancholy and Mystery of a Street, 1914