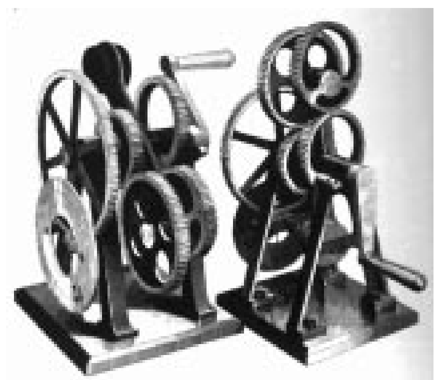
Advances in Concurrency in the Ptolemy Project

Edward A. Lee

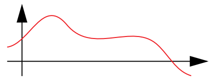
UC Berkeley
Dept. of EECS

Some Models of Computation

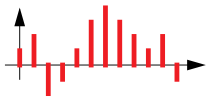


- Gears
- Imperative languages
- Petri nets
- Synchronous dataflow
- Dynamic dataflow
- Process networks
- Concrete data structures
- Discrete-events
- Synchronous/Reactive languages
- Communicating sequential processes
- Finite state machines
- Hierarchical communicating finite state machines

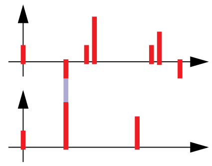
What is Time?



continuous time

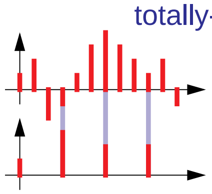


discrete time

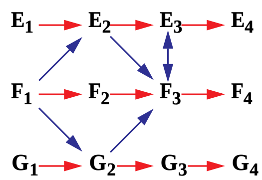


Salvador Dali, *The Persistence of Memory*, 1931

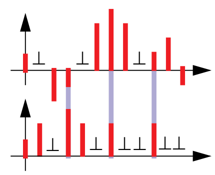
totally-ordered discrete events



multirate discrete time



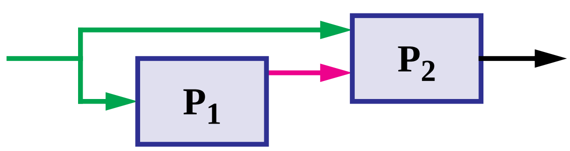
partially-ordered discrete events



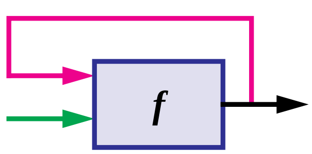
synchronous/reactive

Totally-Ordered Discrete-Event Models

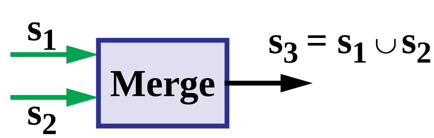
Examples of the sorts of problems that arise from a computerized model of physical time:



What if P_1 is causal but not strictly causal?



What does this mean?



What if s_1 and s_2 have synchronous events?

The Tagged Signal Model

A mathematical framework within which the essential properties of models of computation can be understood and compared.

A denotational framework.

Events and Signals

Abstractions of *time* give us tools to deal with these questions.

- **set of *values*** V
- **set of *tags*** T
- **an event** $e \in T \times V$
- **a *signal* is a set of events**
- **a *functional signal* is a (partial) function** $s: T \to V$
- **the set of all signals** $S = \wp(T \times V)$ **(the powerset)**
- N**-tuples of signals s** $\in S^N$

Possible Interpretations of Tags

- **Universal time ($T = \Re$)**
- **Discrete time (T is a *totally ordered* discrete set)**
- **Precedences (T is a *partially ordered* discrete set)**

Why not always use the "most physical" model: universal time?

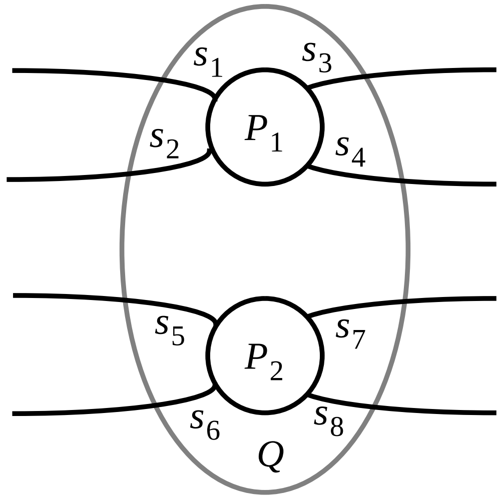
- **In specifying systems, avoid over-specifying.**
- **In modeling systems, recognize the inherent difficulty of maintaining a globally consistent notion of time.**

Processes and Connections

Processes

- **a *process* $P \subseteq S^N$ for some N**
- **a *behavior* s $\in P$ (s *satisfies* the process)**
- **a *process* is a set of possible *behaviors***

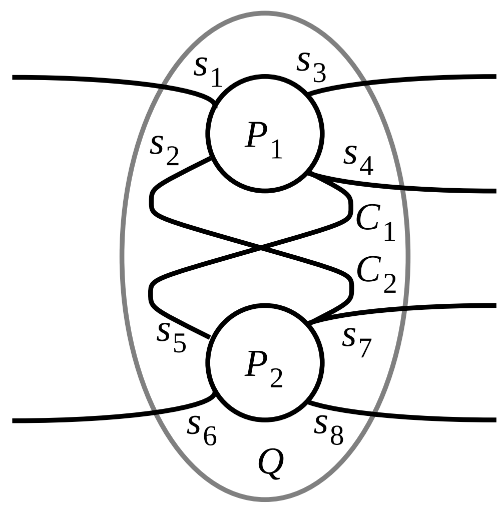
Composing Independent Processes



$$Q = P_1 \times P_2 \subseteq S^8$$

Composing Interacting Processes

A ***connection*** $C \subset S^N$**:** $\mathbf{s} = (s_1, \ldots, s_N) \in C \Leftrightarrow s_i = s_j$



$$Q = (P_1 \times P_2) \cap C_1 \cap C_2$$

Projections and Composition

Let $I = (i_1, \ldots, i_m)$ be an ordered set of indexes in the range $1 \le i \le N$, and define the projection $\pi_I(\mathbf{s})$ of $\mathbf{s} = (s_1, \ldots, s_N) \subseteq S^N$ onto S^m by

$$\pi_I(\mathbf{s}) = (s_{i_1}, \ldots, s_{i_m})$$

Using projection and tensor products, a composition of processes can always be given as an intersection of sets of behaviors:

$$Q = \left(\bigcap_{P_i \in \mathbf{P}} P_i \right)$$

Inputs

- An *input* to a process is an externally imposed constraint $A \subseteq S^N$ such that $A \cap P$ is the total set of acceptable behaviors.
- The *set of all possible inputs* $B \subseteq \wp(S^N)$ is a further characterization of a process.

Example: for a process $P \subseteq S^N$ with m input signals having indexes in the set I, each element $A \in B$ is a set of tuples of signals $\{\mathbf{s} : \pi_I(\mathbf{s}) = \mathbf{s}'\}$ for some $\mathbf{s}' \in S^m$

Determinacy

A process is *determinate* if for all inputs $A \in B$,

$$|A \cap P| = 1 \text{ or } |A \cap P| = 0.$$

Functional Processes

Inputs and Outputs

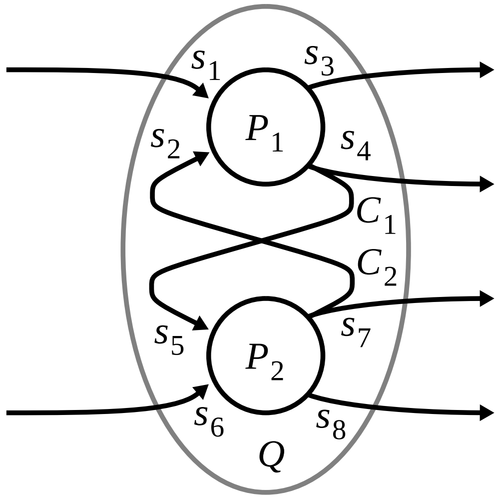
- **an index set I for m input signals and**
- **an index set O for n output signals.**

A process *P* is *functional* with respect to (I, O) if for every $s \in P$ and $s' \in P$ where $\pi_I(s) = \pi_I(s')$, it follows that $\pi_O(s) = \pi_O(s')$.

For such a process, there is a single-valued mapping $F: S^m \to S^n$ such that for all $s \in P$, $\pi_O(s) = F(\pi_I(s))$.

Functional processes are determinate

Example



Suppose:

P_1 **is functional with respect to** $(I, O) = (\{1, 2\}, \{3, 4\})$**.**

P_2 **is functional with respect to** $(I, O) = (\{5, 6\}, \{7, 8\})$**.**

Key question: is Q **functional w.r.t.** $(I, O) = (\{1, 6\}, \{3, 4, 7, 8\})$**? Answer: It depends!**

Partial Ordering of Tags and Events

- ***Partially ordered*****: there exists an irreflexive, antisymmetric, transitive relation "<" between tags.**
- **Version of this relation: "≤".**
- **Ordering of the tags ⇒ ordering of events. Given two events** $e_1 = (t_1, v_1)$ **and** $e_2 = (t_2, v_2)$, $e_1 < e_2 \Leftrightarrow t_1 < t_2$.

Timed Systems

- ***Timed system*****:** T **is totally ordered.**
- ***Metric time*****:** T **is a metric space.**

Discrete Event Systems

Given a process Q, and a tuple of signals $\mathbf{s} \in Q$ that satisfies the process, let $T(\mathbf{s})$ denote the set of tags (time stamps) appearing in any signal in the tuple $\mathbf{s}$.

- **A *discrete-event tag system* is where T is totally ordered, and for every process Q and every behavior $\mathbf{s} \in Q$, there exists an order-preserving bijection from some subset of the integers to $T(\mathbf{s})$.**

Intuitively

Any pair of events in a signal have a finite number of intervening events.

Causality in DE Systems (Intuitively)

- **A *causal* process has a non-negative (but possibly zero) time delay from inputs to outputs.**
- **A *strictly causal* process has a positive time delay from inputs to outputs.**
- **A *delta causal* process has a time delay from inputs to outputs of at least Δ for some constant $\Delta > 0$.**

A Metric Space for DE Signals

In a one-sided DE system, where WOLG $T \subseteq [0, \infty)$, define the *Cantor metric* to be

$$d(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{2^t}$$

where t is the smallest time where the two signals differ, or if $\mathbf{s}_1 = \mathbf{s}_2$, then $d(\mathbf{s}_1, \mathbf{s}_2) = 0$.

With this metric, behaviors of a discrete-event system become points in a metric space!

Causality in the Cantor Metric Space

***Causality*: $d(F(\mathbf{s}), F(\mathbf{s}')) \le d(\mathbf{s}, \mathbf{s}')$.**

***Strict causality*: $d(F(\mathbf{s}), F(\mathbf{s}')) < d(\mathbf{s}, \mathbf{s}')$.**

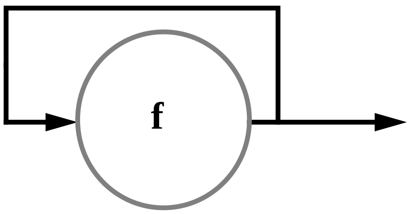
***Delta causality*: there exists a $k < 1$ such that**

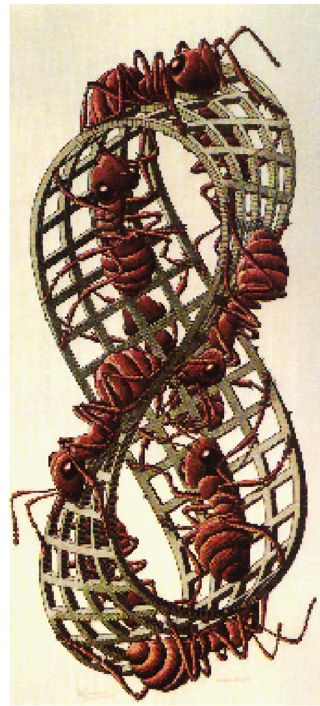
$$d(F(\mathbf{s}), F(\mathbf{s}')) \le k d(\mathbf{s}, \mathbf{s}')$$

F is a *contraction mapping*.

Note: $k = \frac{1}{2^{\Delta}}$.

The Semantics of Feedback





M. C. Escher, *Moebius Strip* II, 1963

For $f{:}S \to S$, define the *semantics* to be a *fixed point* of f

i.e. s such that $f(s) = s$.

Fixed Point Theorems Applied to Discrete-Event Systems

- **If f is strictly causal, then it has at most one fixed point. Hence the feedback composition is determinate.**
- **(*Banach fixed point theorem*) If the metric space is complete (it is) and f is delta causal, then it has exactly one fixed point, and that fixed point can be found by starting with any signal tuple s_0 and finding the limit of:**

$$s_1 = f(s_0),\ s_2 = f(s_1),\ s_1 = f(s_0),\ \ldots$$

- **If the metric space is compact (it is if V is a finite set and all signals are discrete-event), then f only needs to be strictly causal to apply the Banach fixed point theorem.**

Lessons

- **If subsystems are delta causal, then the Banach fixed point theorem gives us a *constructive* way to find their *one unique* behavior.**
- **Specification languages often only insist on *strict causality* (VHDL, for example, has a so-called "delta time" model that, despite the similar name, only ensures strict causality).**
- **The set of VHDL signals is not compact.**
- **The lack of a constructive solution manifests itself in practice (VHDL simulators, for example, can get stuck, where time fails to advance).**

Related Models

- **Fidge, 1991 (processes that can fork and join increment a counter on each event)**
- **Lamport, 1978 (gives a mechanism in which messages in an asynchronous system carry time stamps and processes manipulate these time stamps)**
- **Mattern, 1989 (vector time)**
- **Mazurkiewicz, 1984 (uses partial orders in developing an algebra of concurrent "objects" associated with "events")**
- **Pratt, 1986 (generalizes the notion of formal string languages to allow partial ordering).**
- **Winskel 1993 (describes "event structures," a closely related framework for concurrent systems).**
- **Yates, 1993 (works with Δ-causal functional processes in a timed model with metric time).**

Conclusions

Presented:

- **The beginnings of a framework for understanding and comparing models of computation.**
- **A suite of mathematical techniques for analyzing intrinsic properties of these models of computation.**

This is an evolving model. Can be used to analyze

- **Dataflow**
- **Process networks**
- **Petri nets**
- **Rendezvous-based concurrency models**
- **...**