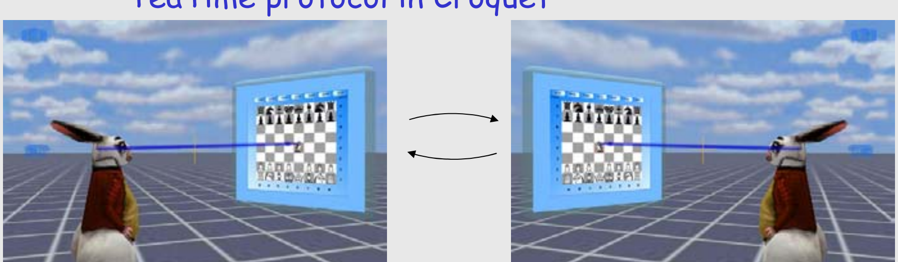
Discrete-Event Systems:
Generalizing Metric Spaces and Fixed-Point Semantics

Adam Cataldo
Edward Lee
Xiaojun Liu
Eleftherios Matsikoudis
Haiyang Zheng

Chess Review
May 11, 2005
Berkeley, CA

Discrete-Event (DE) Systems

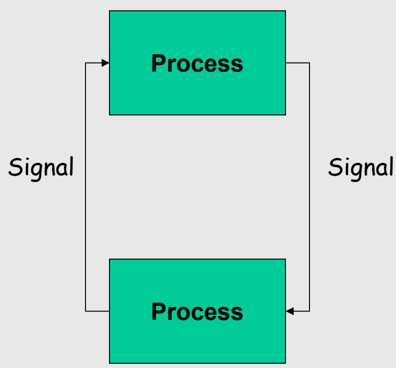
- Traditional Examples
  - VHDL
  - OPNET Modeler
  - NS-2
- Distributed systems
  - TeaTime protocol in Croquet

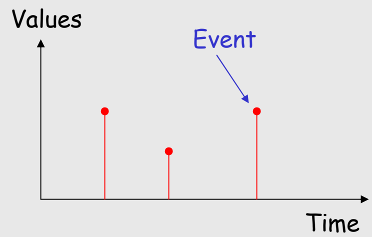


(two players vs. the computer)

Introduction to DE Systems

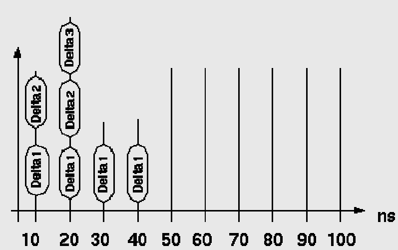
- In DE systems, concurrent objects (processes) interact via signals





What is the semantics of DE?

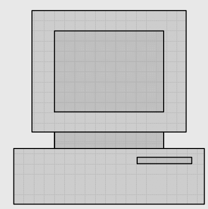
- Simultaneous events may occur in a model
  - VHDL Delta Time



- Simultaneity absent in traditional formalisms
  - Yates
  - Chandy/Misra
  - Zeigler

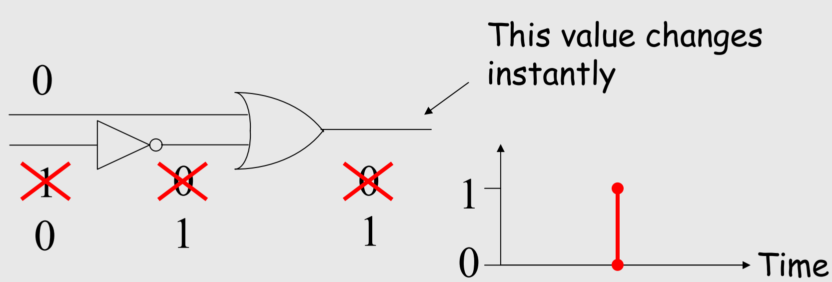
Time in Software

- Traditional programming language semantics lack time
- When a physical system interacts with software, how should we model time?
- One possiblity is to assume some computations take zero time, e.g.
  - Synchronous language semantics
  - GIOTTO logical execution time

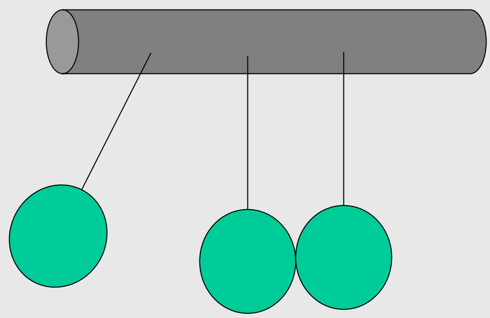


Simultaneneity in Hardware

- Simultaneity is common in synchronous circuits
- Example:



Simultaneity in Physical Systems

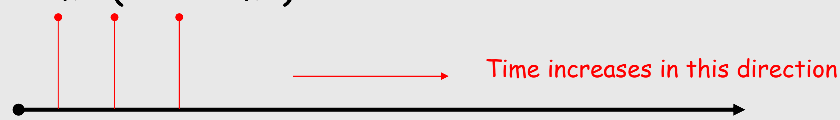


Our Contributions

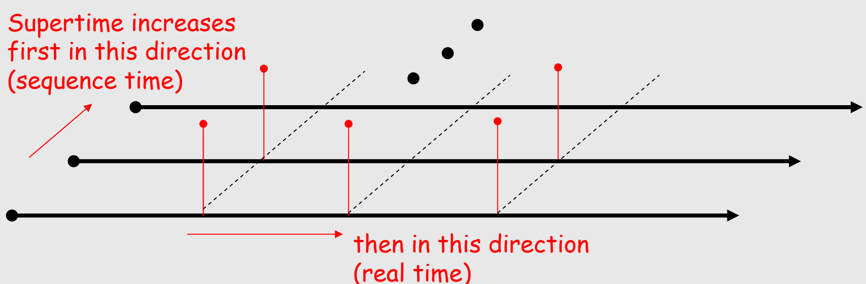
- We generalize DE semantics to handle simultaneous events
- We generalize metric space concepts to handle our model of time
- We give uniqueness conditions and conditions for avoidance of Zeno behavior

Models of Time

- Time (real time)

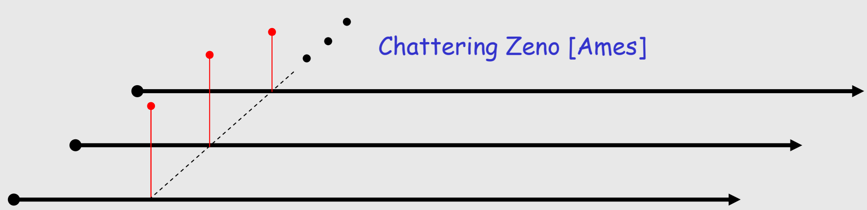


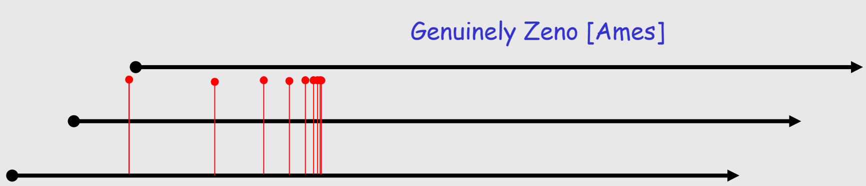
- Superdense time [Maler, Manna, Pnueli]



Zeno Signals

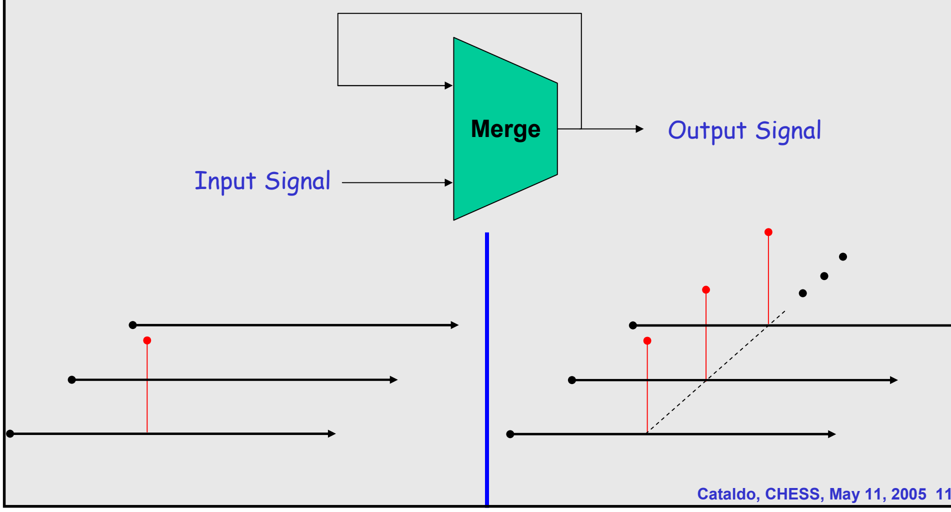
- Definition: *Zeno Signal*
  infinite events in finite real time





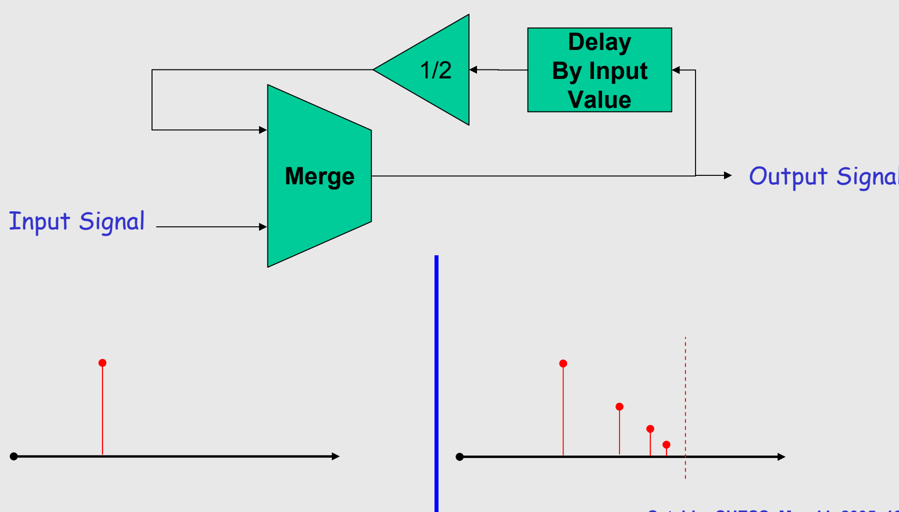
Source of Zeno Signals

- Feedback can cause Zeno



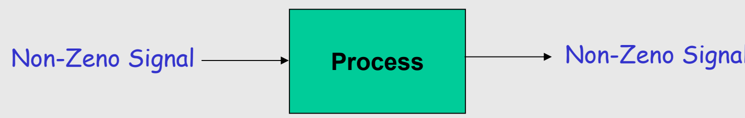
Genuinely Zeno

- A source of genuinely Zeno signals

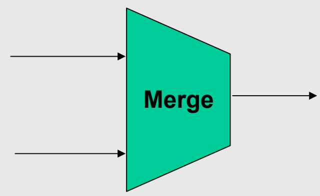


Simple Processes

- Definition: *Simple Process*

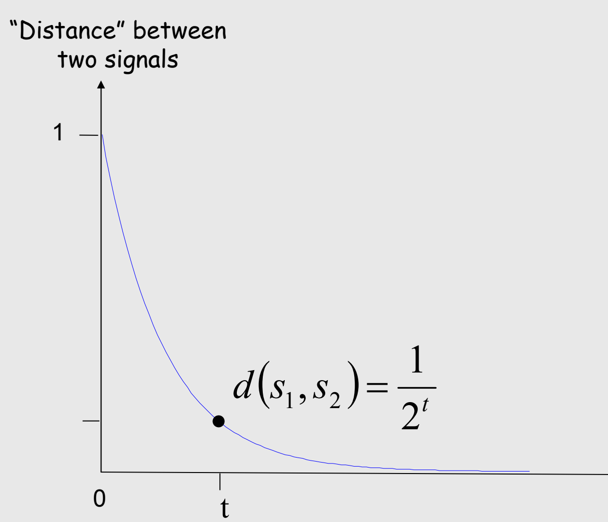


- Merge is simple, but it has Zeno feedback solutions

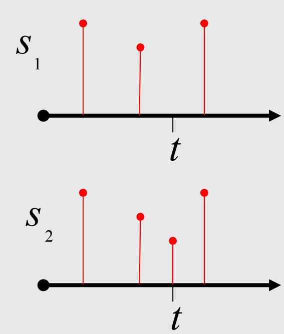


- When are compositions of simple processes simple?

Cantor Metric for Signals



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



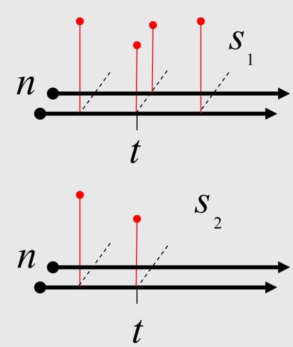
Tetrics: Extending Metric Spaces

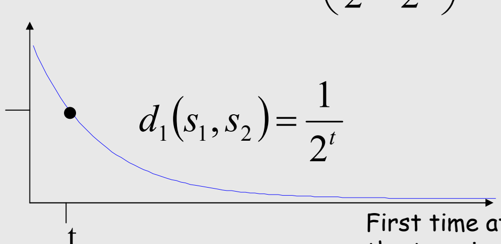
- Cantor metric doesn't capture simultaneity
- Tetrics are generalized metrics
- We generalized metric spaces with "tetric spaces"
- Our tetric allows us to deal with simultaneity

Our Tetric for signals

"Distance" between two signals:

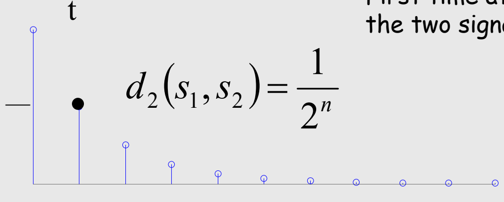
$$d(s_1, s_2) = \left(\frac{1}{2^t}, \frac{1}{2^n} \right)$$





$$d_1(s_1, s_2) = \frac{1}{2^t}$$

First time at which the two signals differ

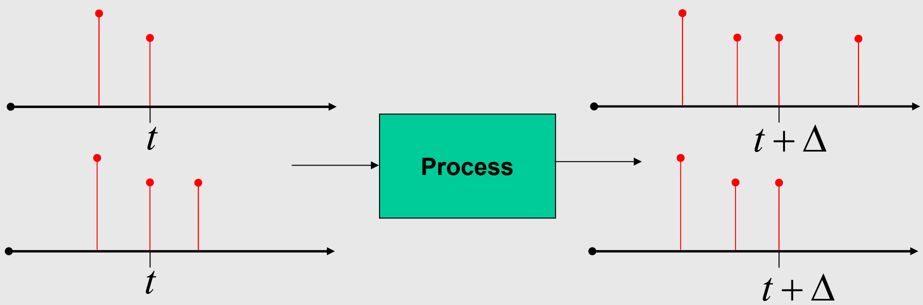


$$d_2(s_1, s_2) = \frac{1}{2^n}$$

First sequence number at which the two signals differ

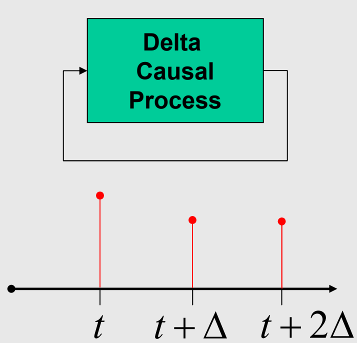
Delta Causal

Definition: *Delta Causal*
Input signals agree up to time t implies
output signals agree up to time $t + \Delta$



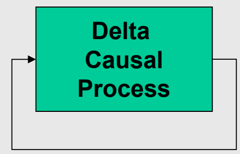
What Delta Causal Means

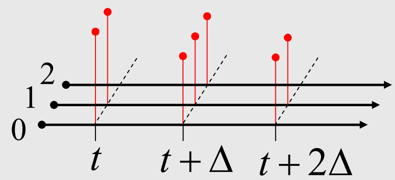
- Signals which delay their response to input events by delta will have non-Zeno fixed points



Extending Delta Causal

- The system should be allowed to chatter

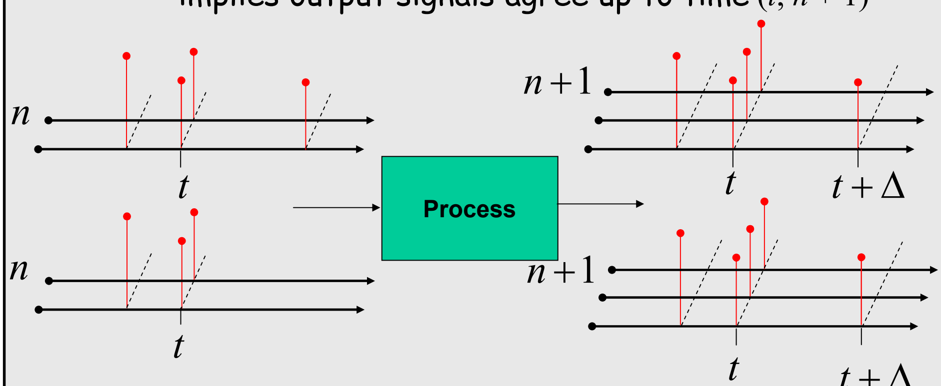




- As long as time eventually advances by delta

Tetric Delta Causal

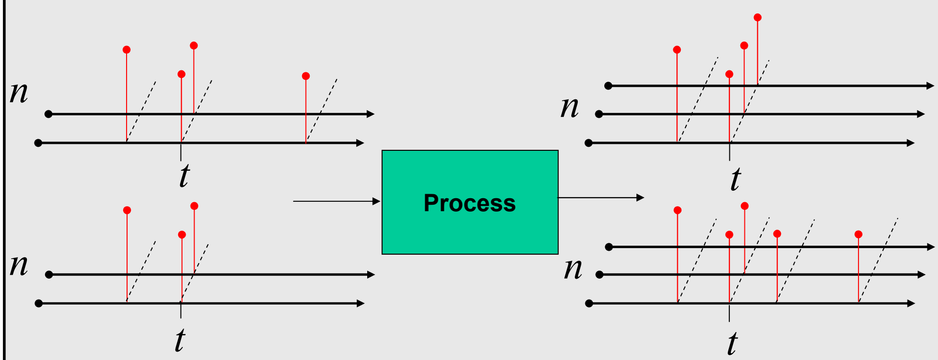
Definition: *Tetric Delta Causal*
1) Input signals agree up to time (t,n)
implies output signals agree up to time $(t, n+1)$



2) If n is large enough, this also
implies output signals agree up to time $(t+\Delta,0)$

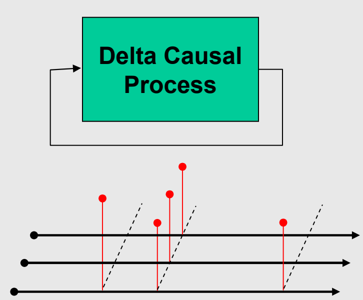
Causal

Definition: *Causal*
If input signals agree up to supertime (t, n) then the output signals agree up to supertime (t, n)



Result 1

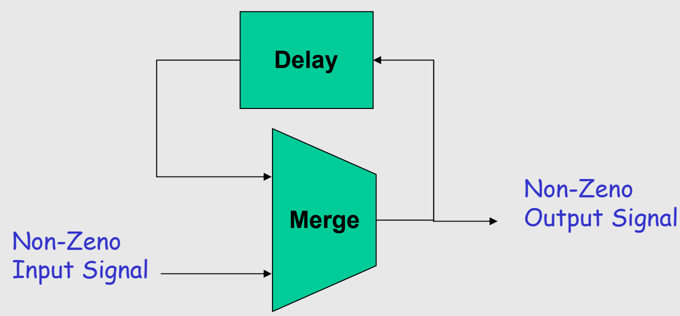
- Every (extended) delta causal process has a unique feedback solution



Unique feedback solution

Result 2

- Every network of simple, causal processes is a simple causal process, provided in each cycle there is a delta causal process
- Example



Conclusions

- We broadened DE semantics to handle superdense time
- We invented tetric spaces to measure the distance between DE signals
- We gave conditions under which systems will have unique fixed-point solutions
- We provided sufficient conditions under which this solution is non-Zeno
- http://ptolemy.eecs.berkeley.edu/papers/05/DE_Systems

Acknowledgements

- Edward Lee
- Xiaojun Liu
- Eleftherios Matsikoudis
- Haiyang Zheng
- Aaron Ames
- Oded Maler
- Marc Rieffel