iCyPhy

Deterministic Actors

Marten Lohstroh, Edward A. Lee

Contributors: Jeronimo Castrillion, Patricia Derler, Christopher Gill, Andrés Goens, Chadlia Jerad, Christian Menard, Íñigo Íncer Romeo, Alberto Sangiovanni-Vincentelli, Martin Schoeberl, Marjan Sirjani, Armin Wasicek.

UC Berkeley

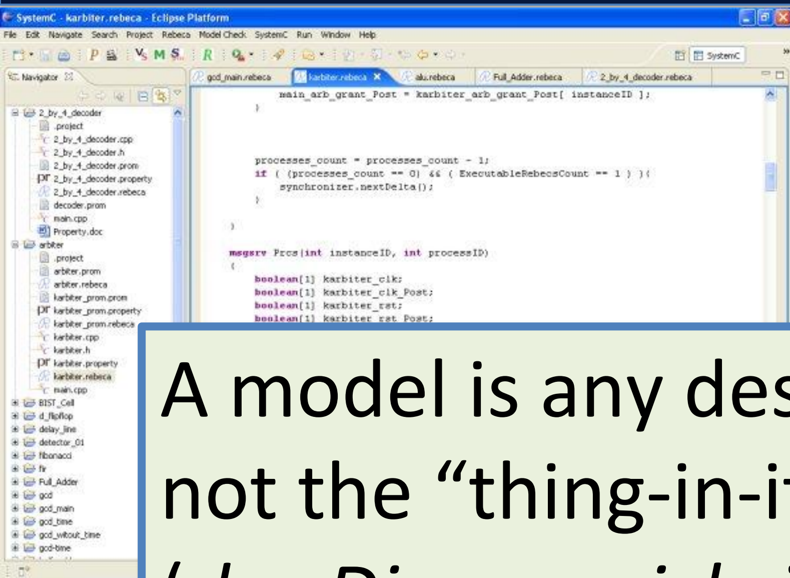
Southampton, September 3, 2019

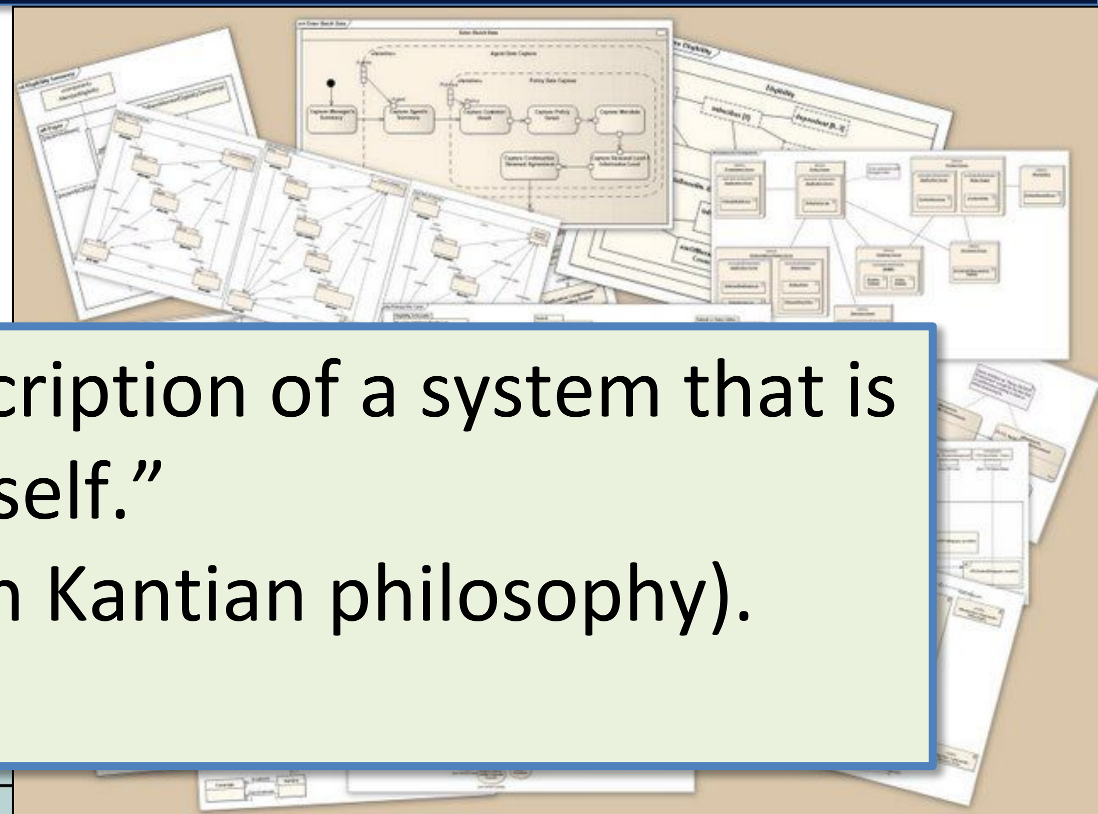
University of California, Berkeley

Determinism as a Property of Models

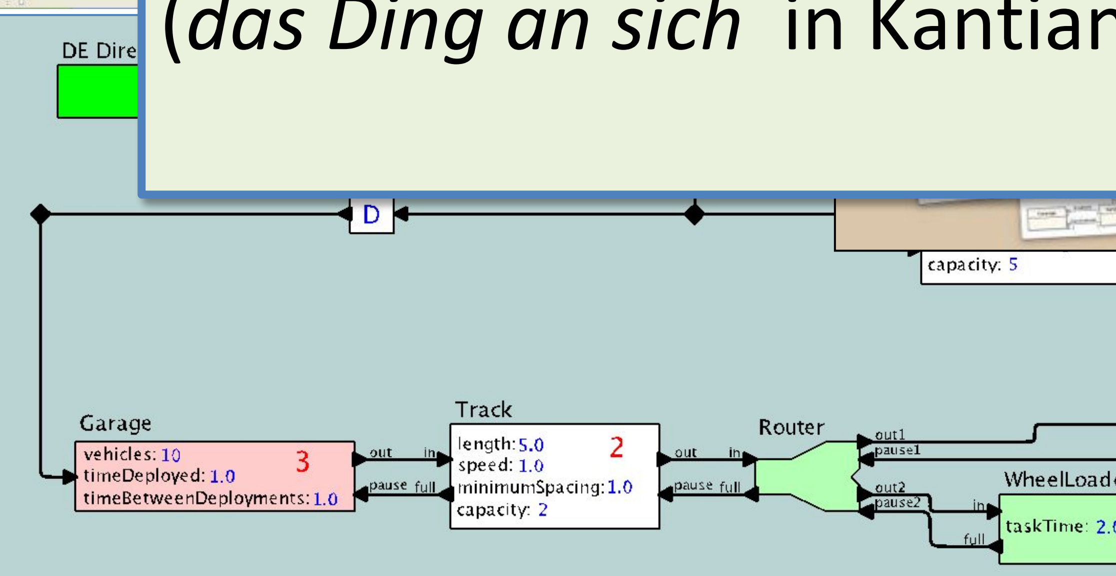
A model is *deterministic* if, given the initial *state* and the *inputs*, the model defines exactly one *behavior*.

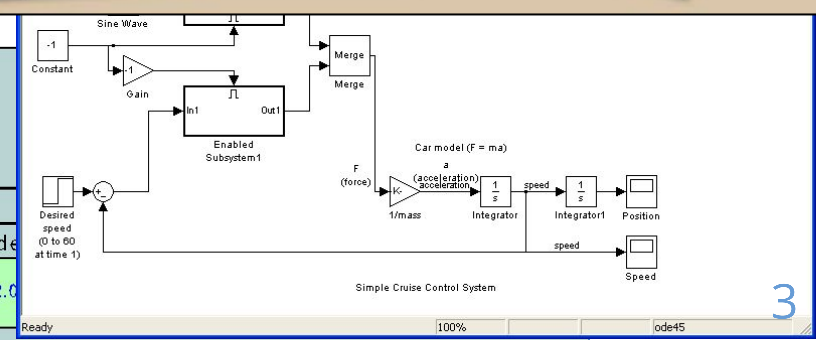
What is a Model?





A model is any description of a system that is not the “thing-in-itself.”
(*das Ding an sich* in Kantian philosophy).





Deterministic Models

Software | ISAs | SDL

Model

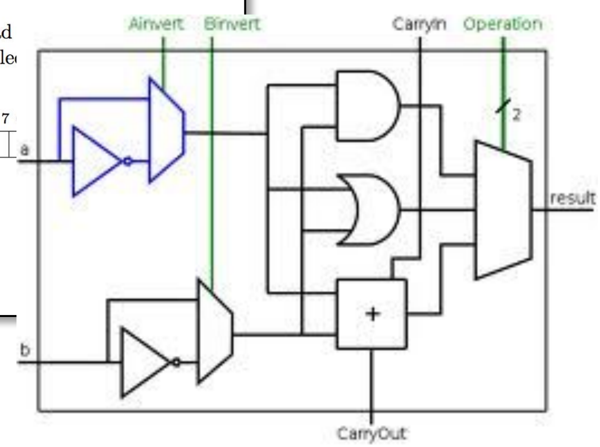
```
void foo(int32_t x) {
        if (x > 1000) {
                x = 1000;
        }
        if (x > 0) {
                x = x + 1000;
                if (x < 0) {
                        panic();
                }
        }
}
```

Integer Register-Register Operations

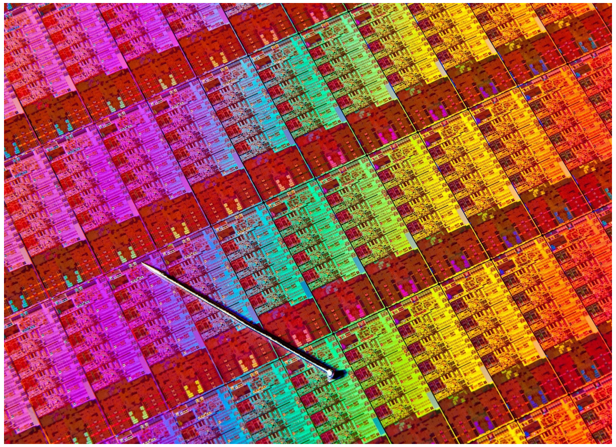
RISC-V defines several arithmetic R-type operations. All operations read as source operands and write the result into register *rd*. The *funct* field sele

31 27	26 22	21 17	16 7
rd	rs1	rs2	funct10
5	5	5	10
dest	src1	src2	ADD/SUB/SLT/SLTU
dest	src1	src2	AND/OR/XOR
dest	src1	src2	SLL/SRL/SRA
dest	src1	src2	ADDW/SUBW
dest	src1	src2	SLLW/SRLW/SRAW

Waterman, et al., The RISC-V Instruction Set Manual, UCB/EECS-2011-62, 2011



Physical System



Images: Wikimedia Commons

Why have deterministic models proven so valuable?

- They enable testing.
  - Known inputs => known outputs
- Analysis is more tractable.
  - Math: Boolean algebra, calculus, etc.
- Simulation is more useful.
  - One input yields one trace.
- Verification scales better.
  - Much smaller state space.
- More certifiable?

Actors: Asynchronous Message Passing

Distributed Systems

ERLANG

akka

Machine Learning

RAY

Cyber-physical Systems

AUTOSAR

Example

```
Actor Bar {
    handler init(){
        Foo x = new
Foo();
        Baz z = new
Baz();
        z.pass(x);
        x.increment(1);
    }
}
```

```
Actor Baz {
    handler pass(Foo x){
        x.double();
    }
}
```

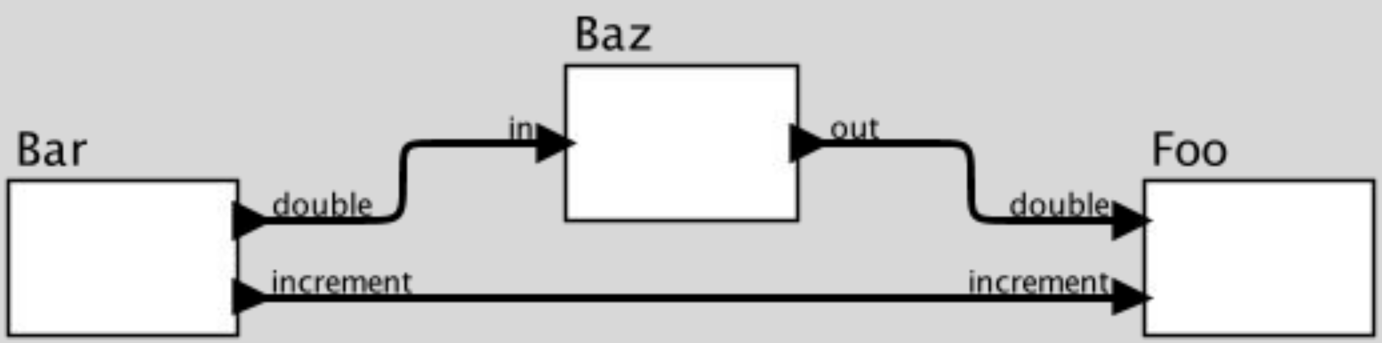
What is printed?

```
Actor Foo {
    int state = 1;
    handler double(){
        state *= 2;
    }
    handler increment(arg){
        state += arg;
        print state;
    }
}
```

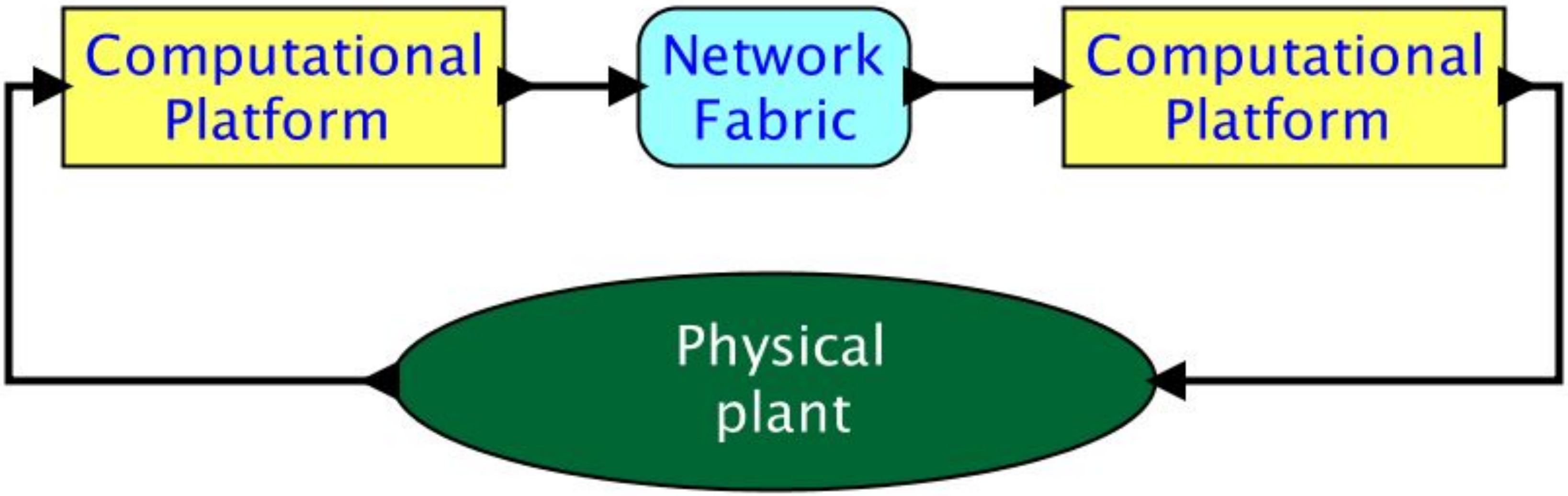
Coordination: Ports, Hierarchy, and Scheduling

```
composite Top {
   Foo x = new Foo();
   Bar y = new Bar();
   Baz z = new Baz();
   connect(y.double, z.in);
   connect(y.increment, x.increment);
   connect(z.out, x.double);
}
```

See our paper for an elaborate discussion on how coordination is done in KPN, DF, SR, and DE.



Cyber-Physical Systems



Predictability requires determinacy and depends on timing, including execution times and network delays.

Lingua Franca Hello World

Polyglot

Time

Periodic reactions

Reaction interfaces

Mutually atomic reactions, written in target language

Deterministic, time-stamped communication

Hierarchical composition

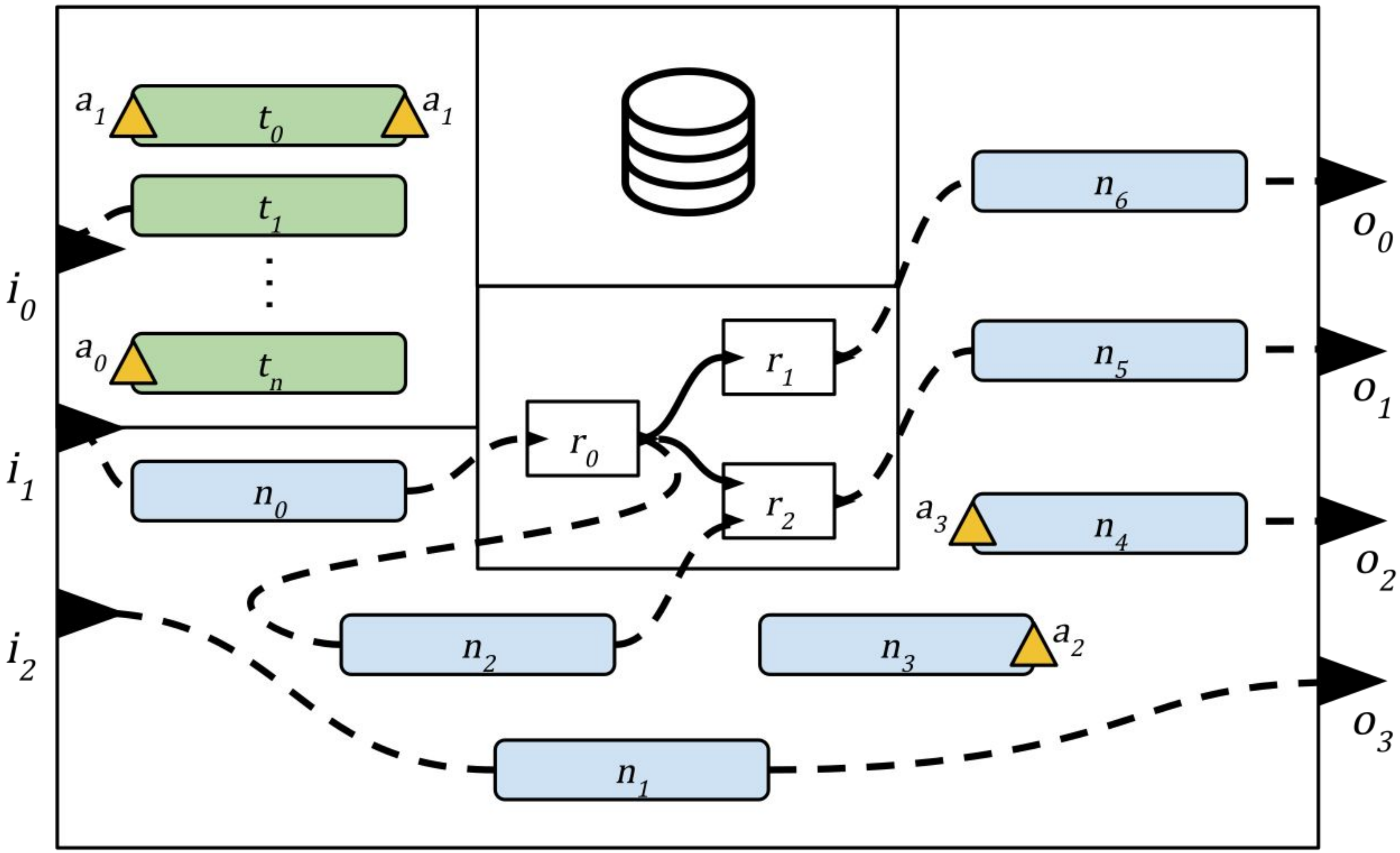
Not shown:

- Inheritance
- Actions
- Deadlines

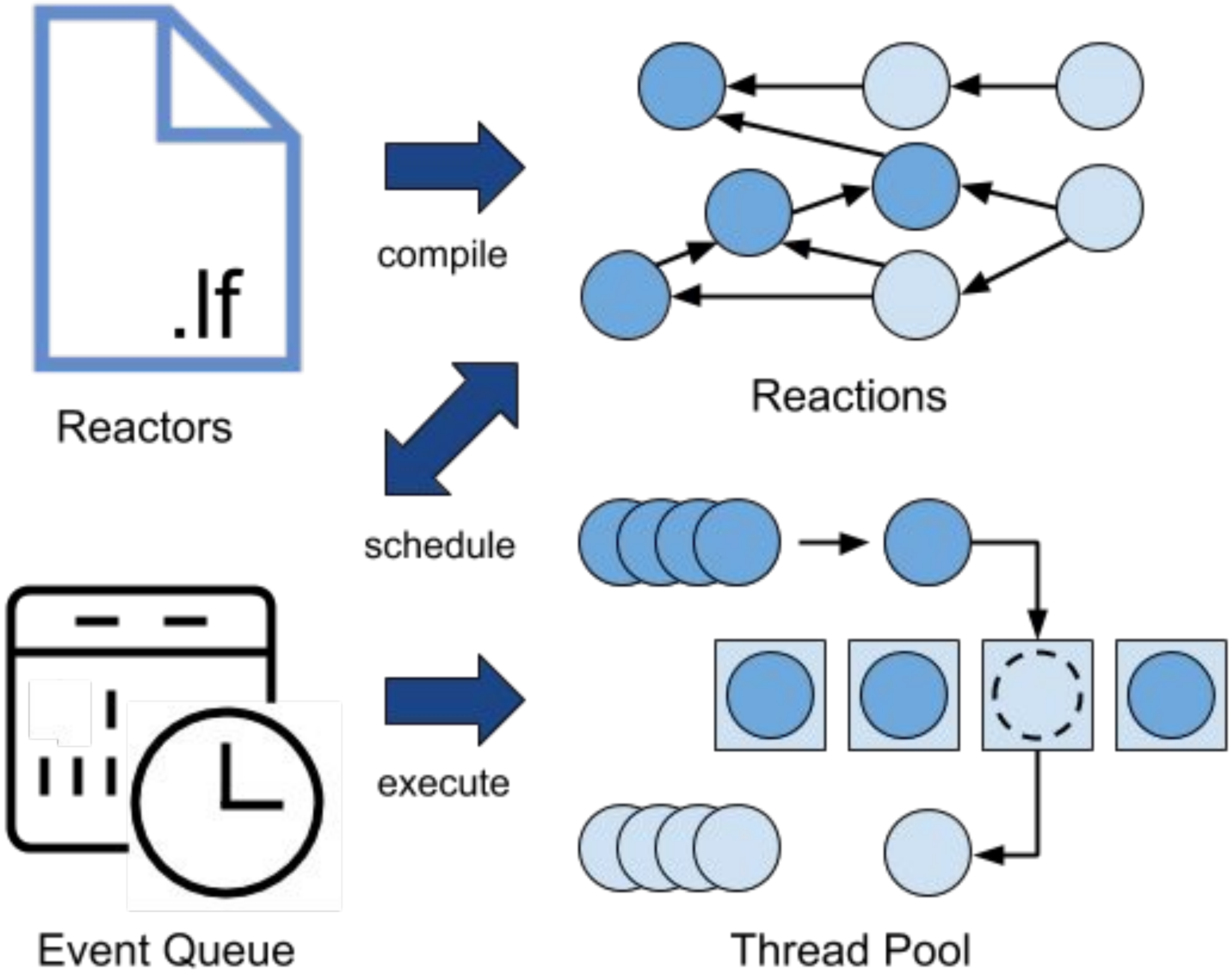
```
target C;
reactor Ramp(period:time(1 sec)) {
    input set:int;
    output out:int;
    timer clock(period);
    state count:int(0);
    reaction(clock) -> out {=
        (this->count)++;
        set(out, (this->count));
    =}
    reaction(set) {=
        this->count = set;
    =}
}
reactor Print {
    input in:int;
    reaction(in) {=
        printf("%d\n", in);
    =}
}

composite Main {
    a = new Ramp(period = 2 secs);
    b = new Print();
    a.out -> b.in;
}
```

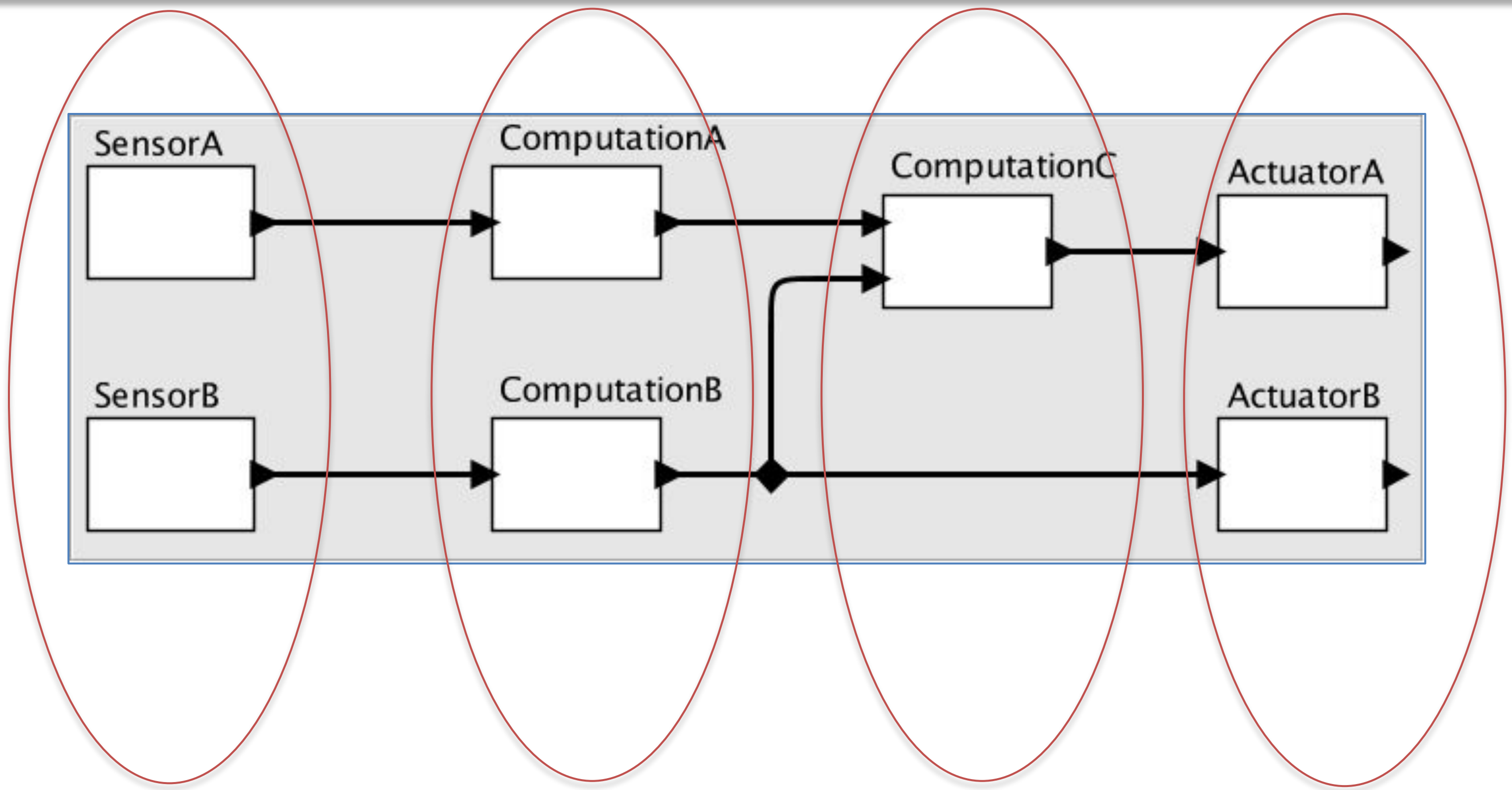
Reactors: Deterministic Actors



Lingua Franca Compiler & Runtime



Questions Addressed by Reactors



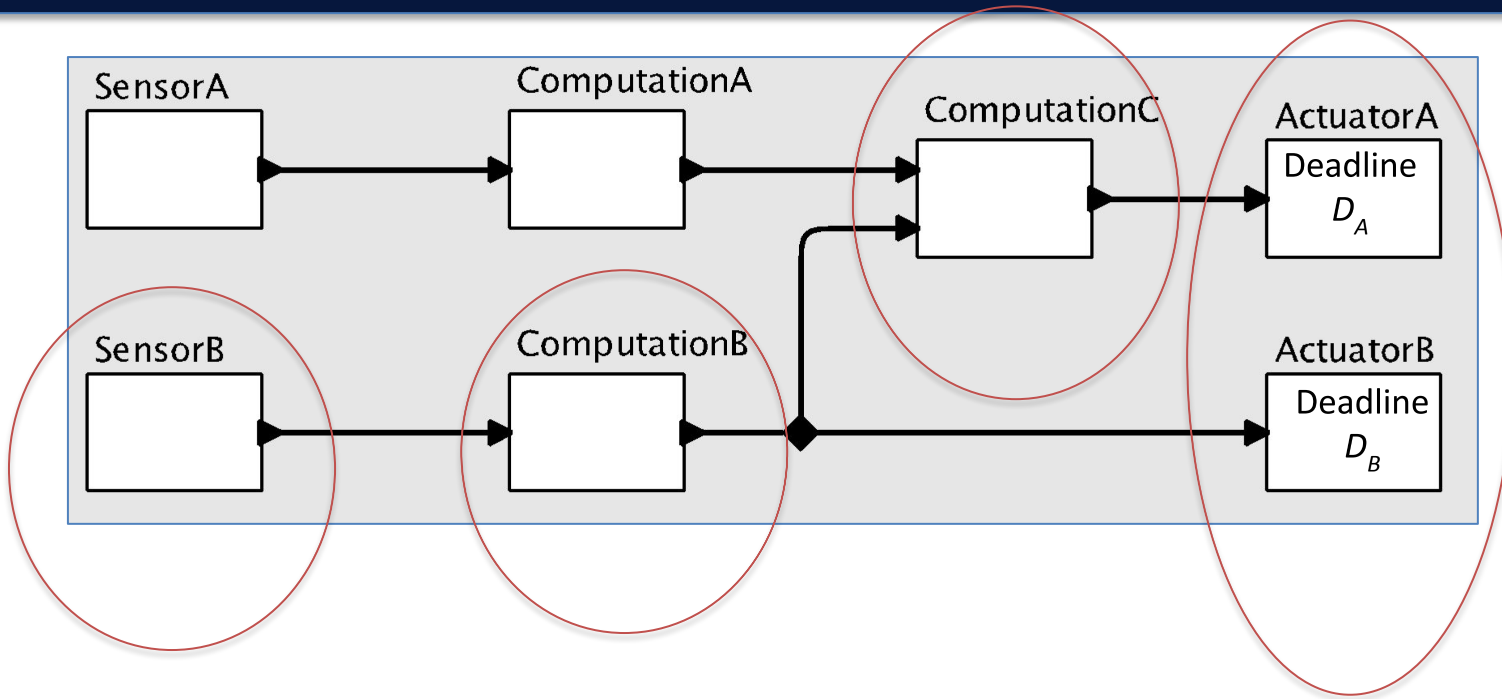
What combinations of periodic, sporadic, behaviors are manageable?

How do execution times affect feasibility? How can we know execution times?

How do we get repeatable and testable behavior when communication is across networks?

How do we specify, ensure, and enforce deadlines?

Logical vs. Physical Time



Sporadic events are assigned a time stamp based on the local physical-time clock

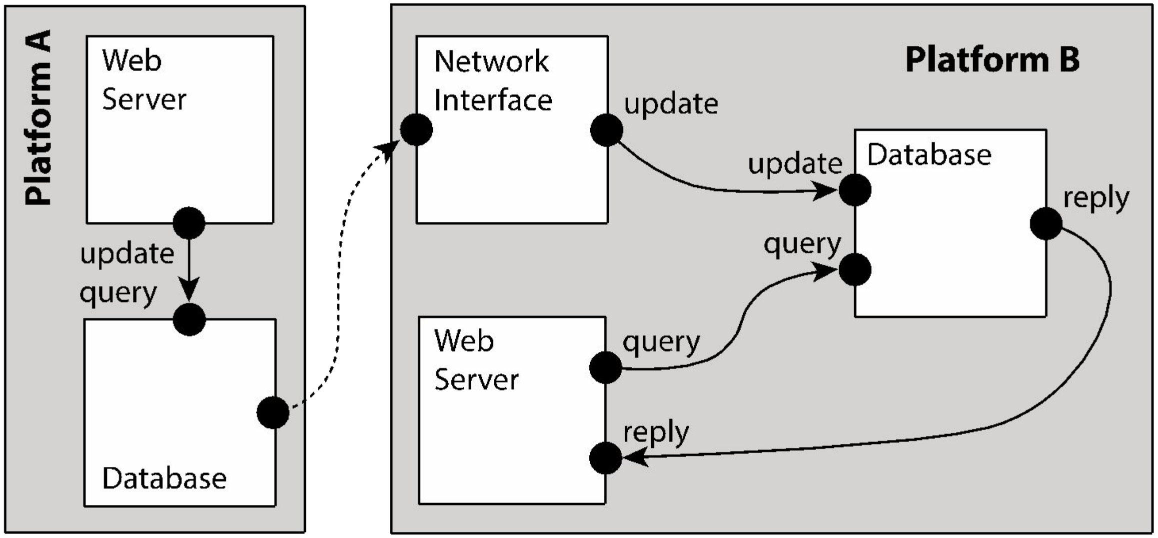
Computations have logically zero delay.

Every reactor handles events in time-stamp order. If time-stamps are equal, events are "simultaneous"

Actuators can have a deadline *D*. An input with time stamp *t* is required to be delivered to the actuator before the local clock hits *t* + *D*.

Distributed Execution w/PTIDES

- Bounded clock synchronization error
- Bounded network latency
- Bounded execution times



in Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 07) , Bellevue, WA, United States.

Fundamental tradeoff between latency and determinism.

A Programming Model for Time-Synchronized Distributed Real-Time Systems

http://ptolemy.org/projects/chess/ptides

Yang Zhao
EECS Department
UC Berkeley

Jie Liu
Microsoft Research
One Microsoft Way

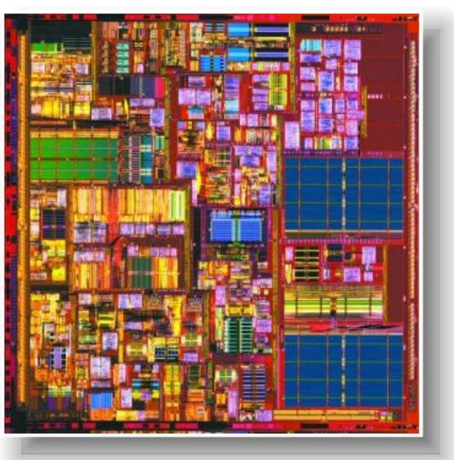
Edward A. Lee
EECS Department
UC Berkeley

PRET Machines

Our Vision: PRET Machines

http://ptolemy.org/projects/chess/pret

PREcision-Timed processors: Performance & Predicability



+

= PRET

Slide from Stephen Edwards and Edward Lee at DAC'07

Conclusions

- Reactors provide a programming model that fits PRET and PTIDES
- The coordination of reactors does not require any code analysis, enabling ***Lingua Franca***:
  - Time is a first-class citizen in the language
  - Nondeterminism must be introduced explicitly using *actions*
  - A small runtime API allows for the reading of inputs, writing of outputs, and scheduling of actions
  - The LF compiler is relatively simple because code optimization is left to the target language compile entirely
  - Adding support for a new target language only requires writing a runtime and code generator

https://github.com/icyphy/lingua-franca