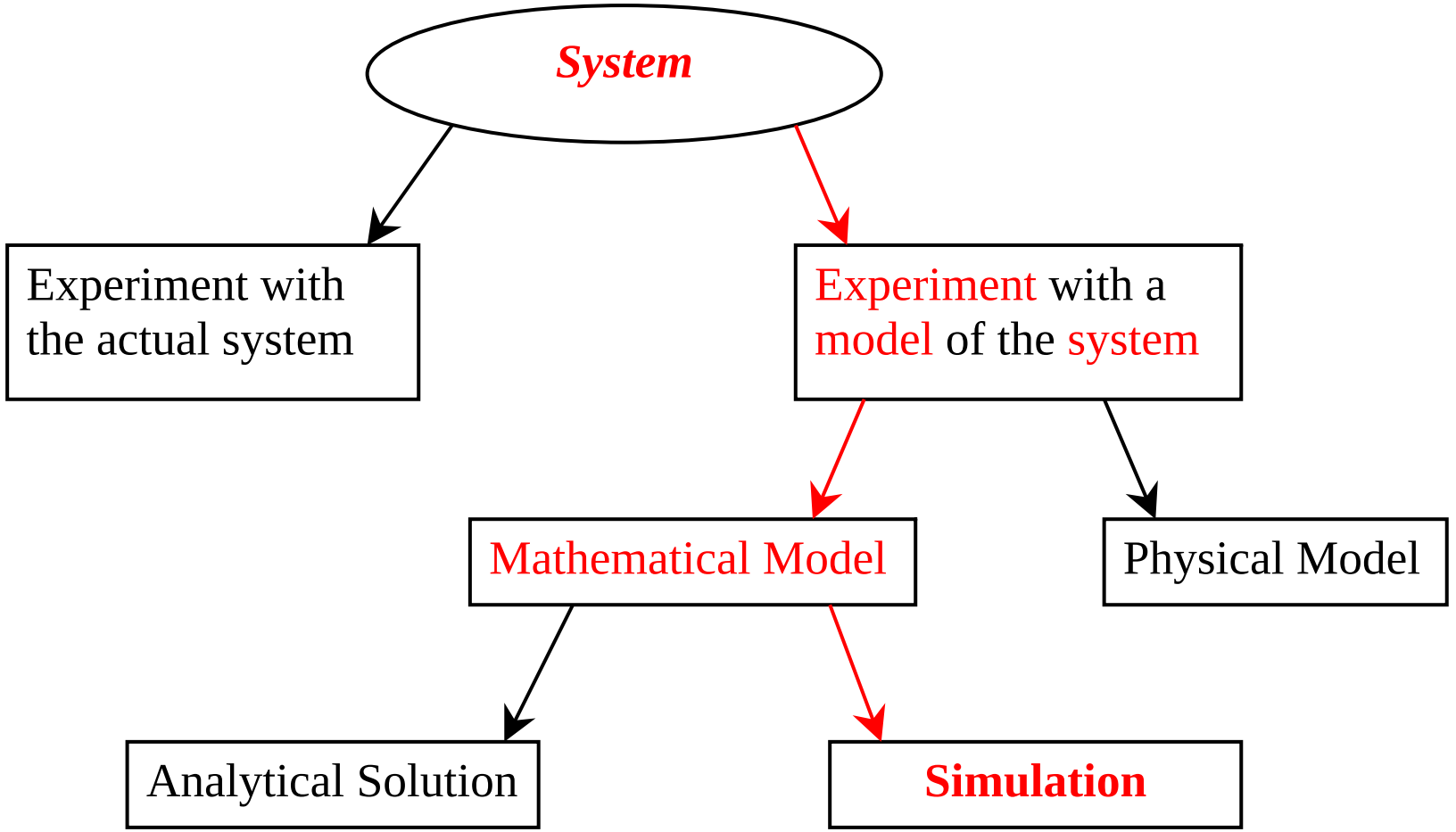
Modeling and Simulation / Distributed Computing

- A quick overview

Ways to Study a System



Definitions - 1

What is a **SYSTEM** **?**

"A system is a potential source of data"

Bernard Siegler, *Theory of Modeling and Simulation*, 1976.

What is an **EXPERIMENT** **?**

"An experiment is the process of extracting data from a system by exerting it through its inputs."

François Cellier, *Continuous System Modelling*, 1990.

Definitions - 2

What is a **MODEL**?

" A model (*M*) for a system (*S*) and an experiment (*E*) is anything to which E can be applied in order to answer questions about S."

Marvin Minski, *Models, Minds, Machines*, 1965.

Note that ...

- A model is not necessarily a computer program !
- Here we concentrate on models that can be expressed as computer programs, the so-called *Mathematical Models*.
- By definition, a model can be qualified as a system, which allows to cut out smaller pieces to generate a new model (*hierarchy of models*)
  - Jack Kleijnen, “Concept of Meta-Models,” in *Progress in Modelling and Simulation*, 1982.
- A model is always related to the tuple *System* and *Experiment*.
- Model validation *always* relates to an experiment to be performed on a system.

Definitions - 3

What is a **SIMULATION**?

"A simulation is an experiment performed on a model."

Granino Korn and John Wait, *digital continuous system simulation*, 1978.

Note that ...

- A mathematical simulation is a coded description of an experiment with a reference to the model to which this experiment is applied.
- Note the (important) separation between model description and experiment description.
- Also note the potential danger of this separation; it is very easy to apply an experiment to a model for which the model is not *valid*.

The Dangers of Simulation

- Both it's strength and weakness: generality and ease of its application…

"All too often, simulation is a love story with an unhappy ending. We create a model of a system, and then fall in love with it. Since love is usually blind, we immediately forget all about the experimental frame, we forget that this is *not* the real world, but that it represents the world only under a very limited set of experimental conditions (we become 'model addicts')."

F. Cellier, *Continuous System Modeling*, 1990

Why is Modeling Important ?

"Modeling means the process of organizing knowledge about a given system"

Bernard Ziegler, *Multifaceted Modeling and Discrete Event Simulation*, 1984

"... it can thus be said that modeling is the single most central activity that unites *all* scientific and engineering endeavors. While the scientist is happy to simply *observe* and *understand* the world, i.e., create a model of the world, the engineer wants to *modify* it to his advantage. While science is all *analysis*, the essence of engineering is *design*.."

"... *simulation* can be used not only for analysis (direct problems) but also for design (inverse problems)."

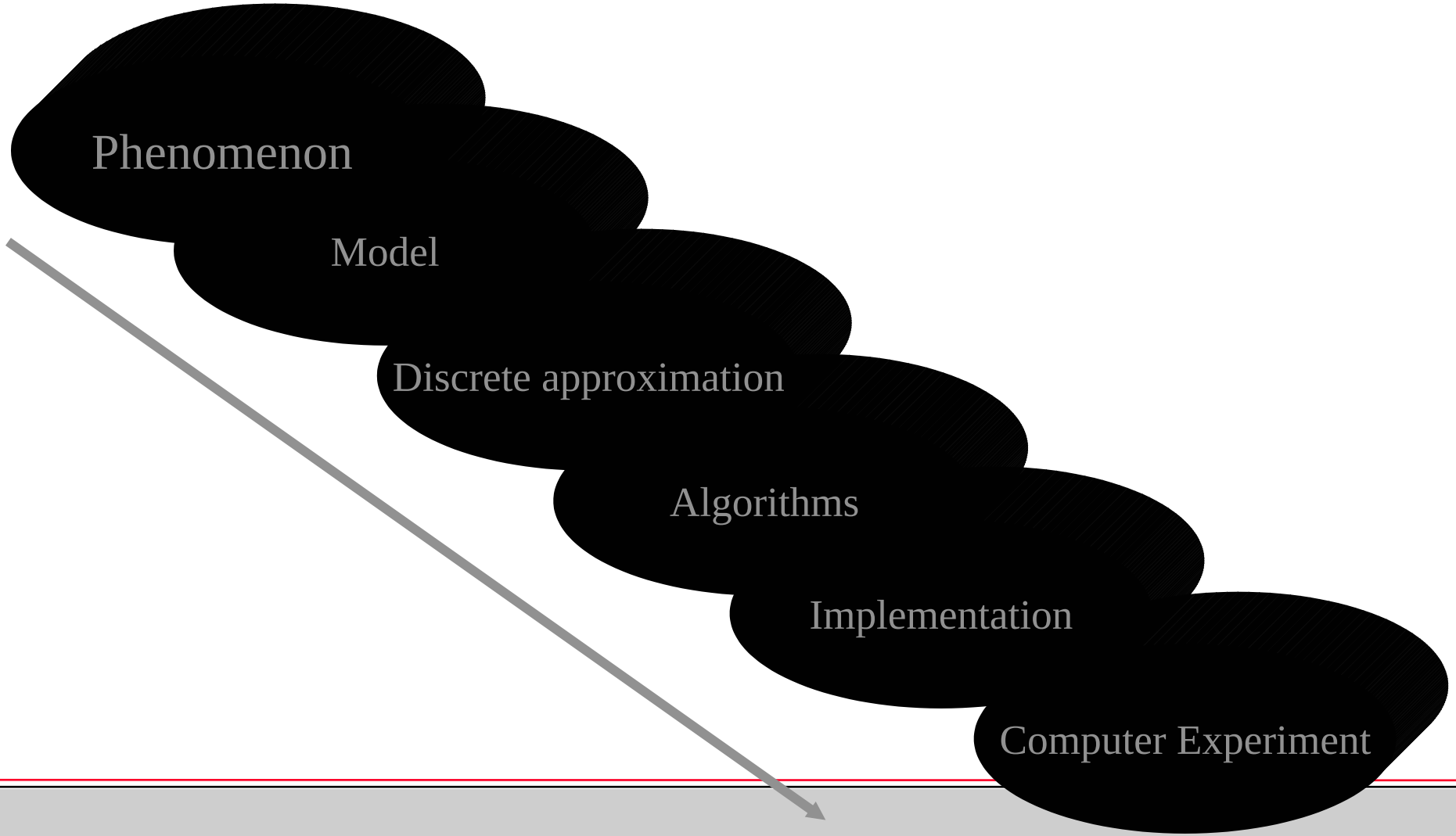
François Cellier, *Continuous System Modelling*, 1990

Reasons to use Simulation

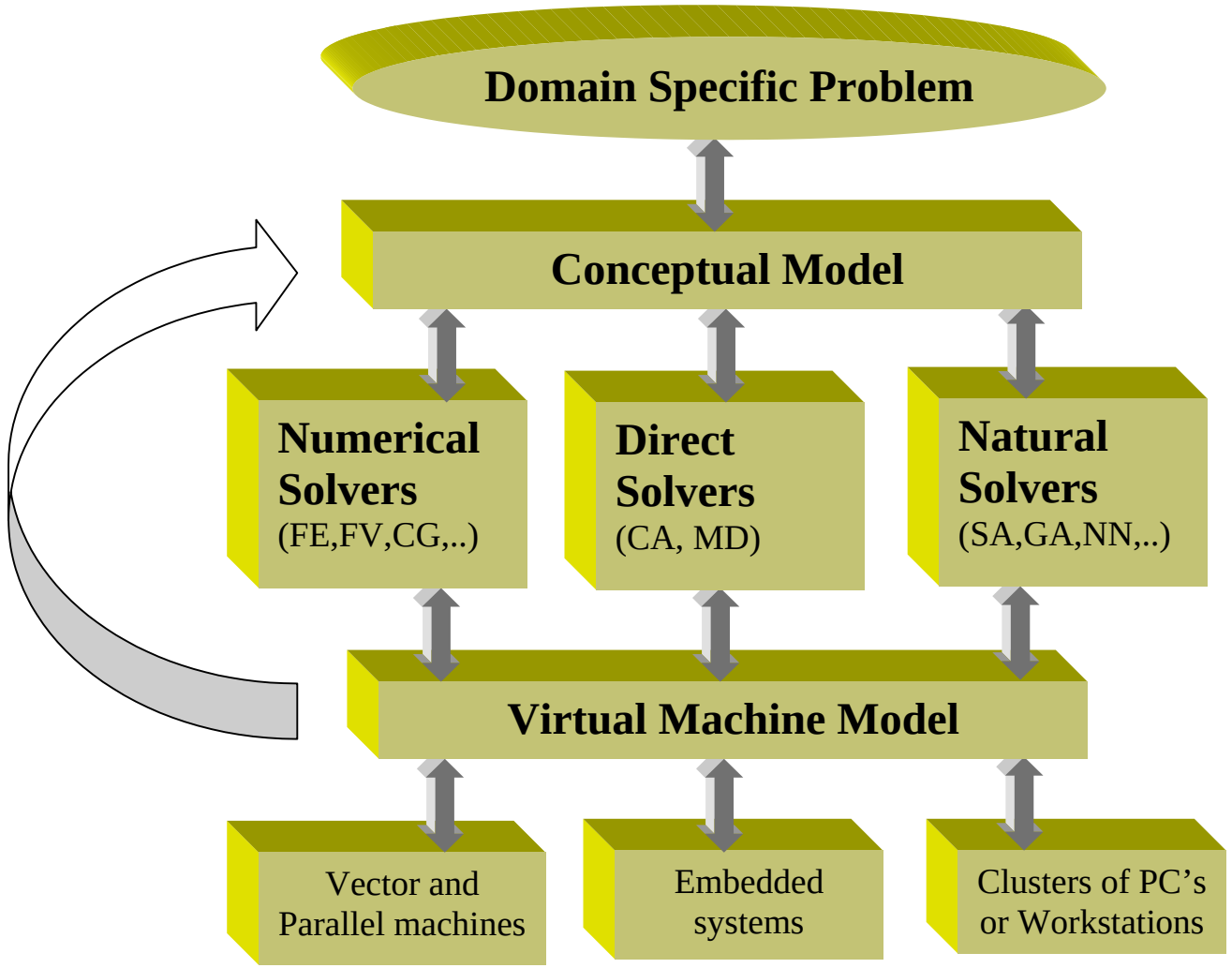
- The physical system in not available
- The experiment may be dangerous
- The cost of the experiment is too high
- The time constants of the system are not compatible with those of the experimenter
- Control variables may be inaccessible
- Suppression of disturbances
- Suppression of second-order effects


SAAB
Airbag
BMW
Molecule

The computer experiment I



Modeling and Simulation Cycle

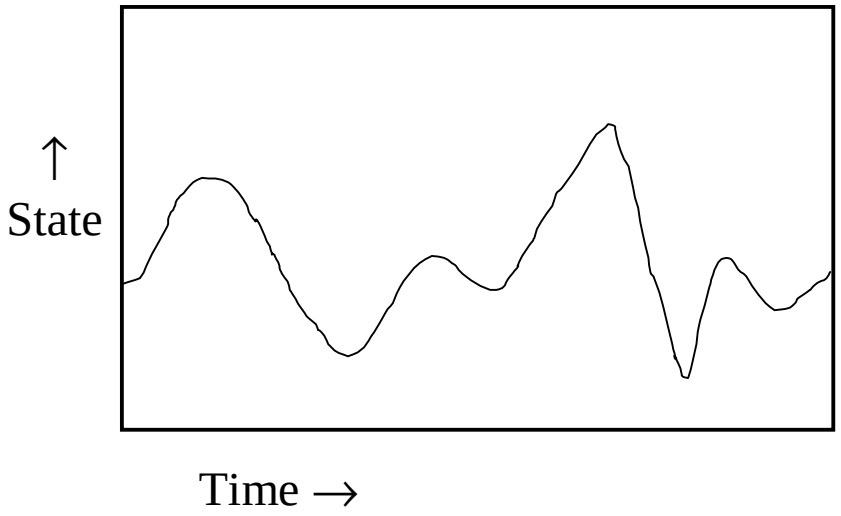


Validation and Verification

- Validation
  - Check the validity of the model.
- Verification
  - Check the correctness of the simulation.

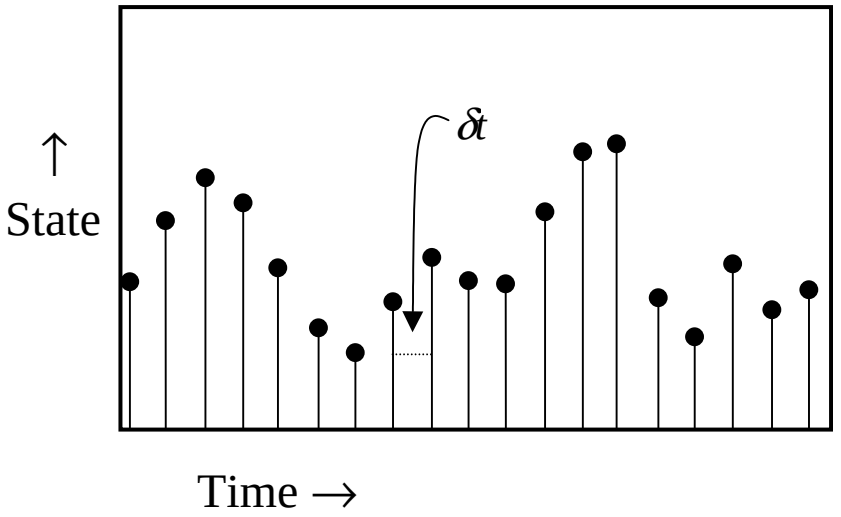
Types of Mathematical Models I

- Continuous Time model
  - State variables change continuously over time.
  - Sets of differential equations.



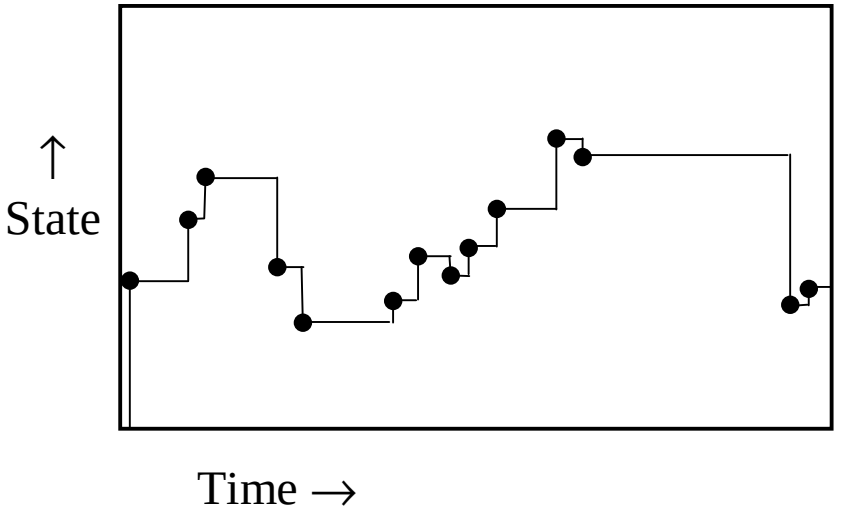
Types of Mathematical Models II

- Discrete Time model
  - State variables change at discrete time intervals.
  - Sets of difference equations.
  - Can occur naturally (e.g. population dynamics), or in engineering.
  - Discretized versions of continuous models.

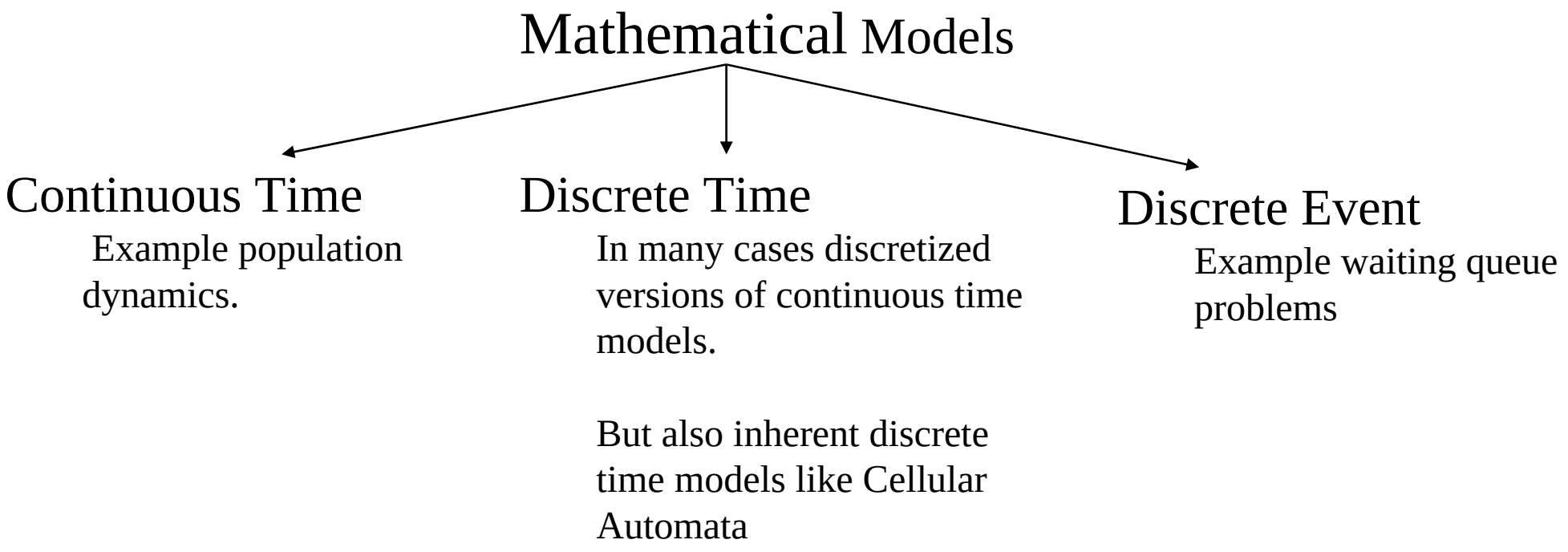


Types of Mathematical Models III

- Discrete Event model
  - State variables change at discrete events.
  - Paradoxically, both the time axis and the state axis in Discrete Event Models are continuous.
  - In a finite time span only a finite number of changes may occur.



Overview Mathematical Models



- Within a *finite* time span, the state variables change their values *infinitely* often

- Represented by: Set of differential Equations
  - Lumped parameter models: ODE
  - Distributed parameter models: PDE

Time Driven

- the simulation clock is advanced in increments of exactly 👎t time units

- the time step 👎t is small enough to capture every event in the system

Event Driven

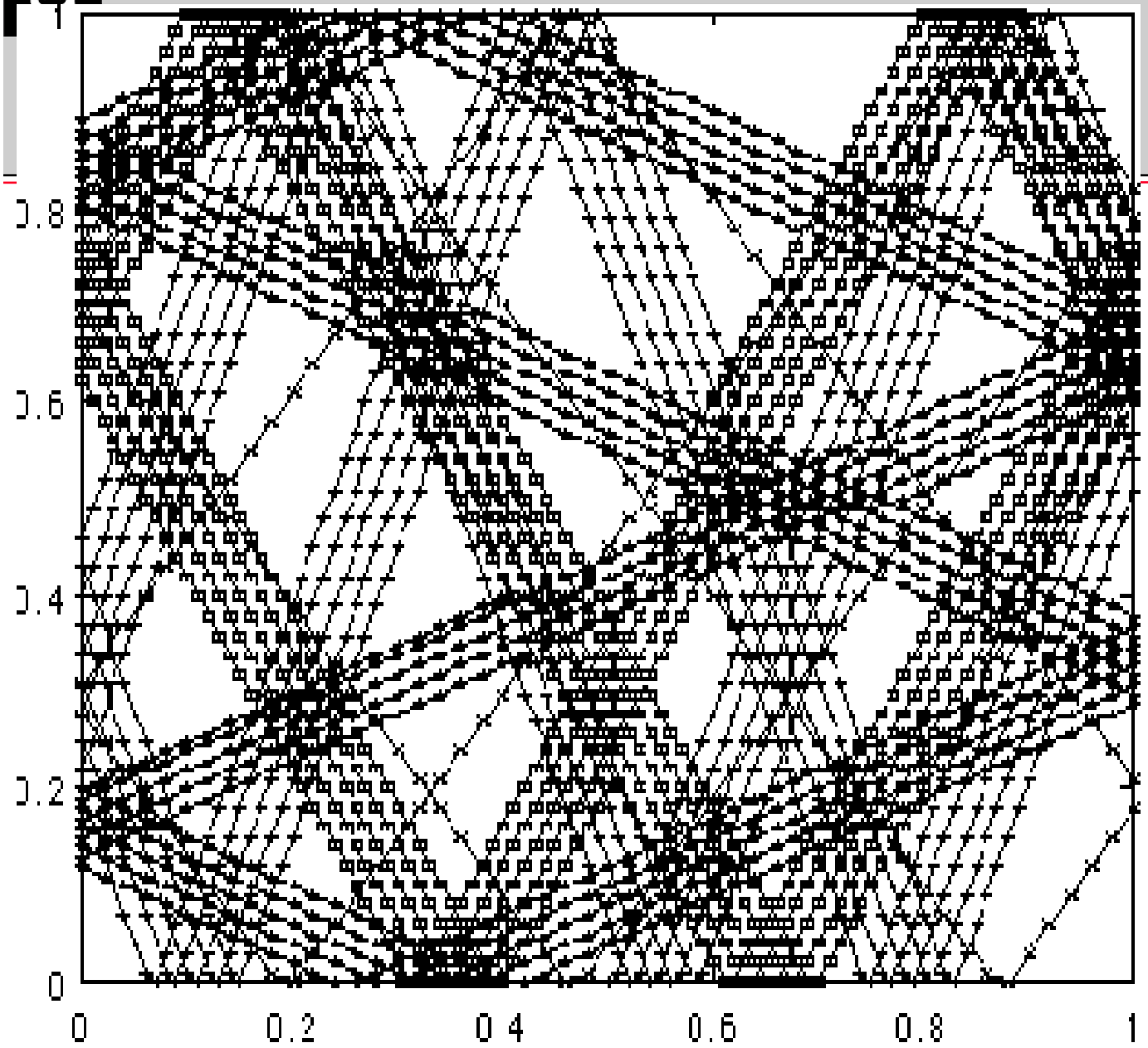
- **Step 1: The simulation clock is initialized to zero and the times of occurrence of future events are determined.**
- **Step 2: The simulation clock is advanced to the time of the occurrence of the most imminent (i.e. first) of the future events.**
- **Step 3: The state of the system is updated to account for the fact that an event has occurred.**
- **Step 4: Knowledge of the times of occurrence of future events is updated and the first step is repeated.**

Billiard-ball example

- Gas dynamics of 4 gas molecules, v=const.
- 2D kinetic energy conservation
- Consider Time and Event driven:

Billiard-ball Time Driven Simulation

```
input stop, simulation time, 👎t;
initialize balls with x,y position and velocity vx,vy
in x,y direction;
time = 0.0;
while (time < stop simulation)
{
for each ball do {
            x += 👎t * vx;
            y += 👎t * vy;
            if (x == 0) vx = -vx;
            if (y == 0) vy = -vy;
      }
time += 👎t;
}
```



Result Time Driven

- $\Delta t = 0.001$

1000 state evaluations

Billiard Balls event driven I

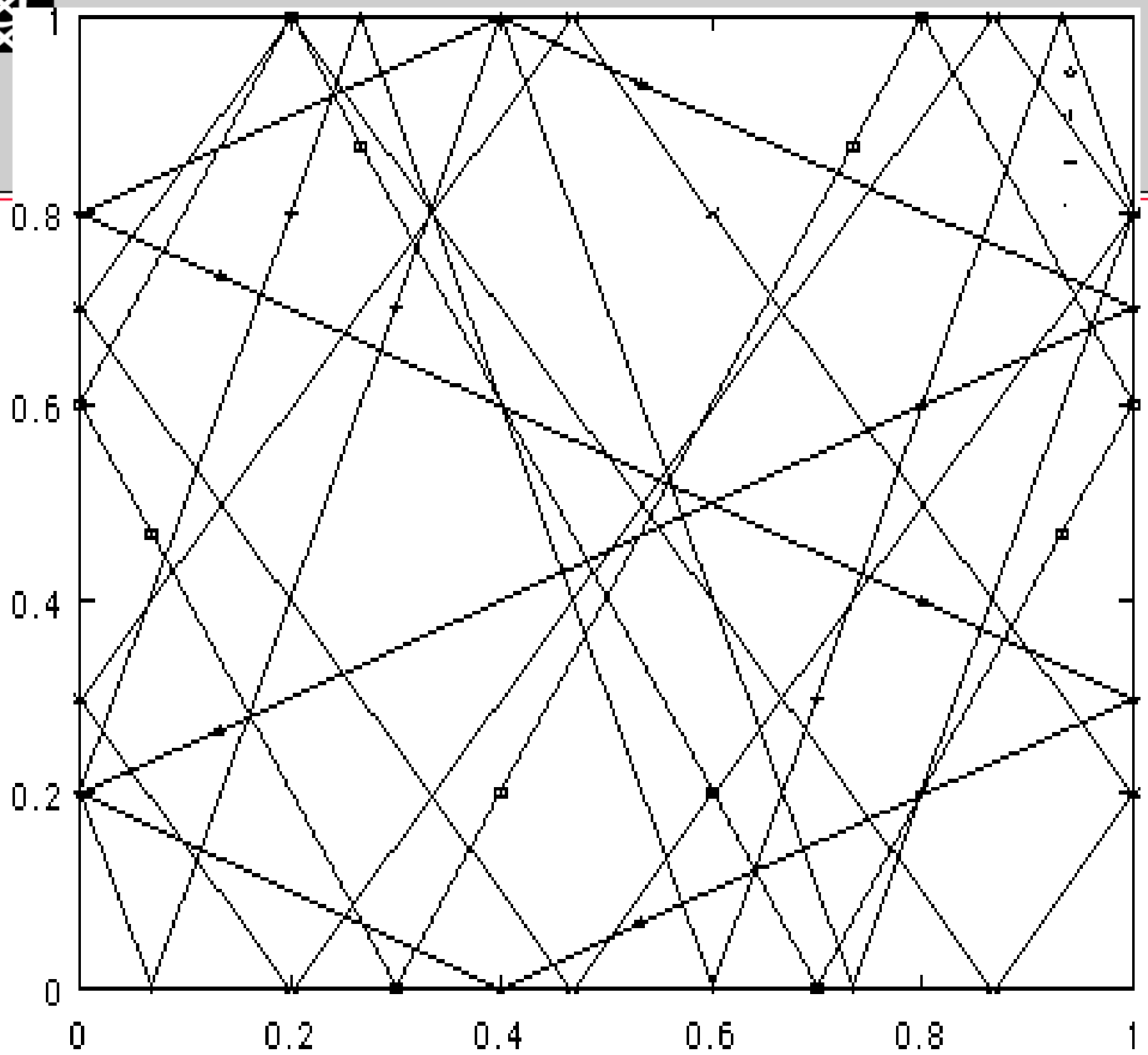
```
input stop_simulation_time;
initialize balls with x,y position and velocity vx,vy in
x,y direction;
for each ball do {
      impact_time = collision_time(ball);
      schedule(MOVE, impact_time, ball);
}
prev_time = 0.0;
while (time() < stop_simulation_time) {
      next_event(&event, &ball);
      switch(event) {
      case MOVE:
            update_positions(time() - prev_time);
            impact_time = collision_time(ball);
            schedule(MOVE, impact_time, ball);
            prev_time = time();
            break;
      }
```

Billiard Balls event driven II

```
collision_time(ball)
{
      if (vx >= 0) {
            t0 = (1 - x) / xv;
      }
            else {
            t0 = -x / xv;
            }
      if (vy >= 0) {
            t1 = (1 - y) / yv;
      }
            else {
            t1 = -y / yv;
            }
      return min(t0, t1);}
```

Billiard Balls event driven III

```
update_positions(👎t)
{
      for each ball do {
            x += 👎t * vx;
            y += 👎t * vy;
            if (x == 0 || x == 1)
                  vx = -vx;
            if (y == 0 || x == 1)
                  vy = -vy;
      }
```



Result Event Driven

60 state evaluations

Some Observations

- If the time between the succeeding events becomes small, time driven is preferred. This is the *frequency* parameter.

- The overhead per event (i.e., the state update) is larger with event-driven simulation. This is referred to as the *event overhead*.

- The amount of work between any two events plays an important role. This we call the *granularity* of the system.