Simulation

COS 323

Why Simulation?

- Make predictions or make decisions regarding complex phenomena or poorly-understood phenomena
- Test theories about how real systems work
- Explore consequences of changes to a system
- Train people to make better decisions or take correct actions
- …

Simulation

One program variable for each element in the system being simulated; numerical and/or finite-state transitions

… as opposed to

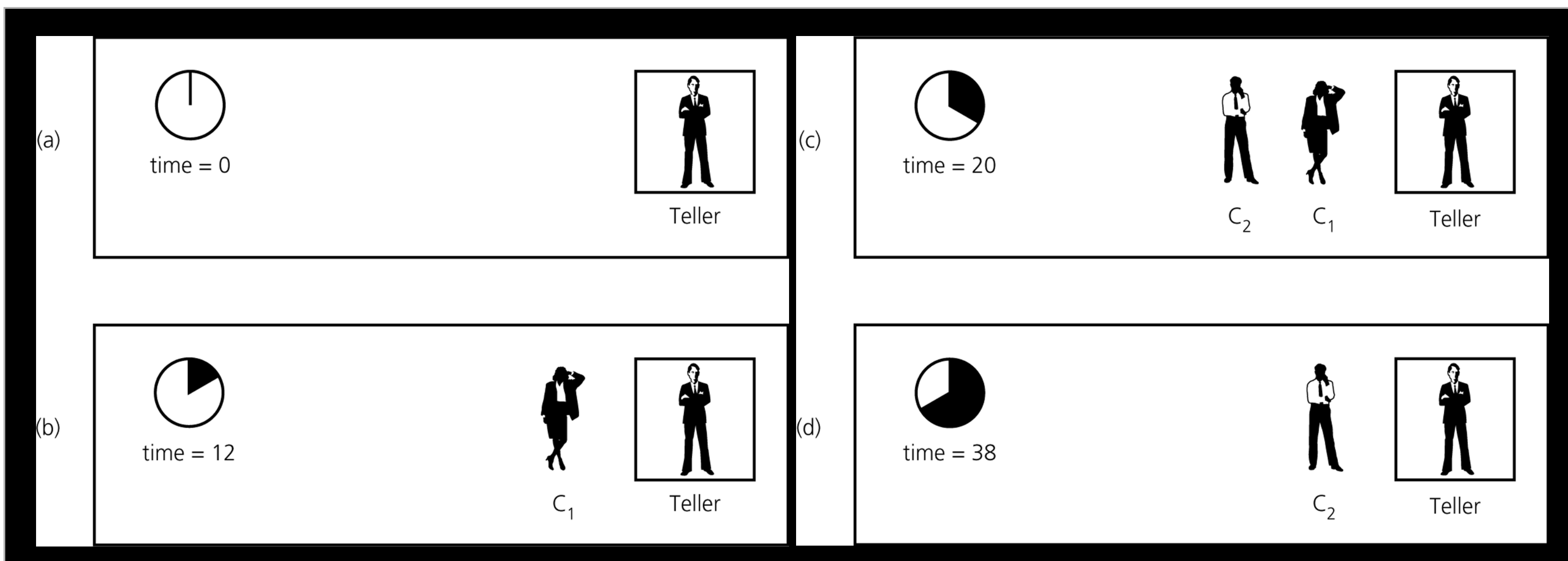
– analytical solution
– formulation of algebraic or differential equations

Approaches to Simulation

- Differential equation solvers can be thought of as conducting a *simulation* of a physical system
  - Advance through time
  - “Continuous” equations model change in state
- Some simulations are more “discrete”:
  - Decisions, actions, events happen at discrete points in time

Discrete Event Simulation: Bank Teller Example

- Simple example: lines at the bank
  - Customers arrive at random times
  - Wait in line(s) until teller available
  - Conduct transaction of random length



Bank Teller

- Simple example: lines at the bank
  - Customers arrive at random times
  - Wait in line(s) until teller available
  - Conduct transaction of random length
- Simulate arbitrary phenomena
  (e.g. spike in customer rate during lunch)
- Goal: mean and variance of waiting times
  - As a function of customer rate, # tellers, # queues

Bank Teller

- *Time-driven* simulation:
  - A master clock increments time in fixed-length steps
  - At each step, compute probabilities of customer(s) arriving, transactions finishing
    - e.g., probability of 2% that a new customer arrives at each time step
  - More accurate simulation with shorter time steps, but then have more steps when *nothing* happens

Bank Teller

- *Event-driven* simulation: All computation happens at changes to system state
  - New customer arrives
  - Teller finishes processing a customer

Bank Teller Event-Driven Simulation

- Compute times of events and put in a “future event list”:
  - When will the next customer arrive?
  - When new customer reaches teller, when will the transaction finish?
- Repeatedly process one event, then fast-forward until scheduled time of next event
- Good accuracy and efficiency: automatically use “time steps” appropriate for what is happening

Time-driven Example: Epidemics

The SIR Model

- W. O. Kermack and A. G. McKendrick, 1929
- Susceptible: susceptible, not yet infected
- Infected: infected and capable of spreading
- Recovered / removed: recovered and immune

$$\frac{dS}{dt} = -\beta S I$$

$$\frac{dI}{dt} = \beta S I - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Time-Driven Simulation: Epidemics

- [Dur95] R. Durrett, "Spatial Epidemic Models," in Epidemic Models: Their Structure and Relation to Data, D. Mollison (ed.), Cambridge University Press, Cambridge, U.K., 1995.

- Discrete-time, discrete-space, discrete-state

Durrett's Spatial SIR model

- Time, $t = 0, 1, 2, \ldots$
- Space: orthogonal (square) grid
- State: {susceptible, infected, removed}

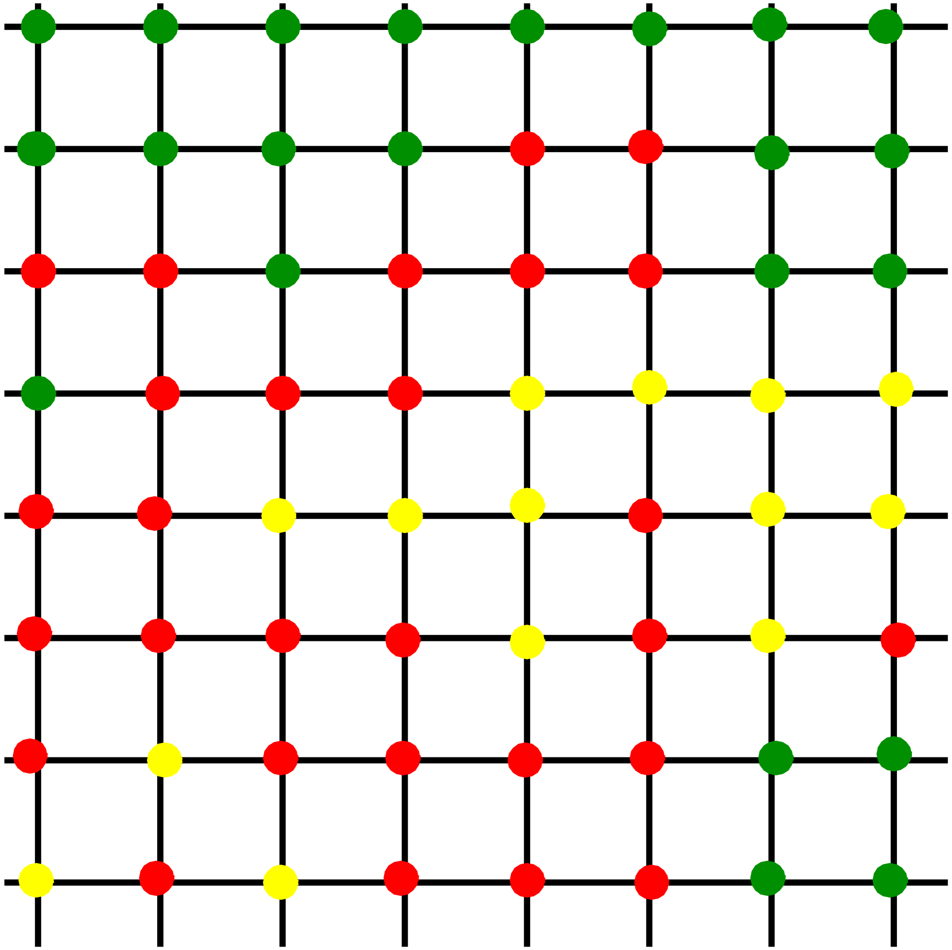
Rules tell us how to get from t to $t+1$ for each spatial location

Each site has 4 neighbors,

contains 0 or 1 individual

Durrett's Rules for Spatial SIR model

- Susceptible individuals become infected at rate proportional to the number of infected neighbors
- Infected individuals become healthy (removed) at a fixed rate δ
- Removed individuals become susceptible at a fixed rate α



Time, t = 0, 1, 2, …
Space: orthogonal (square) grid
State: {susceptible, infected, removed}

Simulation Results

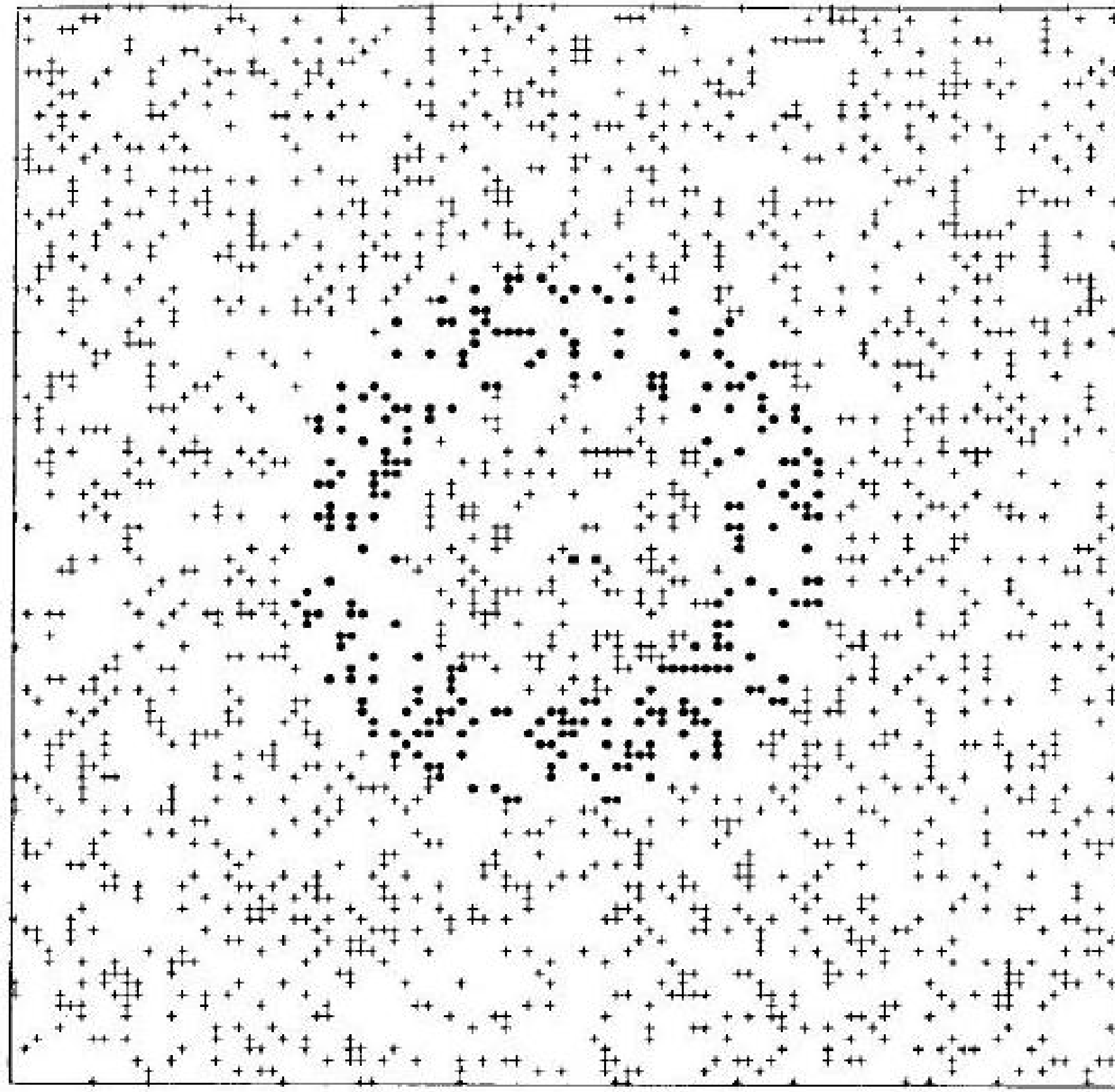
$\alpha = 0$: No return from removed; immunity is permanent. If δ, recovery rate, is large, epidemic dies out. If δ is less than some critical number, the epidemic spreads *linearly* and approaches a *fixed shape*.

→ Can be formulated and proven as a theorem!

$\alpha > 0$: behavior is more complicated

More recent work:

"Epidemic Thresholds and Vaccination in a Lattice Model of Disease Spread“, C.J. Rhodes and R.M. Anderson, *Theoretical Population Biology* **52**, 101118 (1997) Article No. TP971323.



Note ring of vaccinated individuals.

The SZR model

- Susceptible
  - Can die naturally with parameter delta (become Removed)
  - Can become zombie-infected with parameter beta
- Zombie
  - Can be killed by human with parameter alpha (become removed)
- Removed
  - Removed humans can be resurrected into zombies with parameter zeta

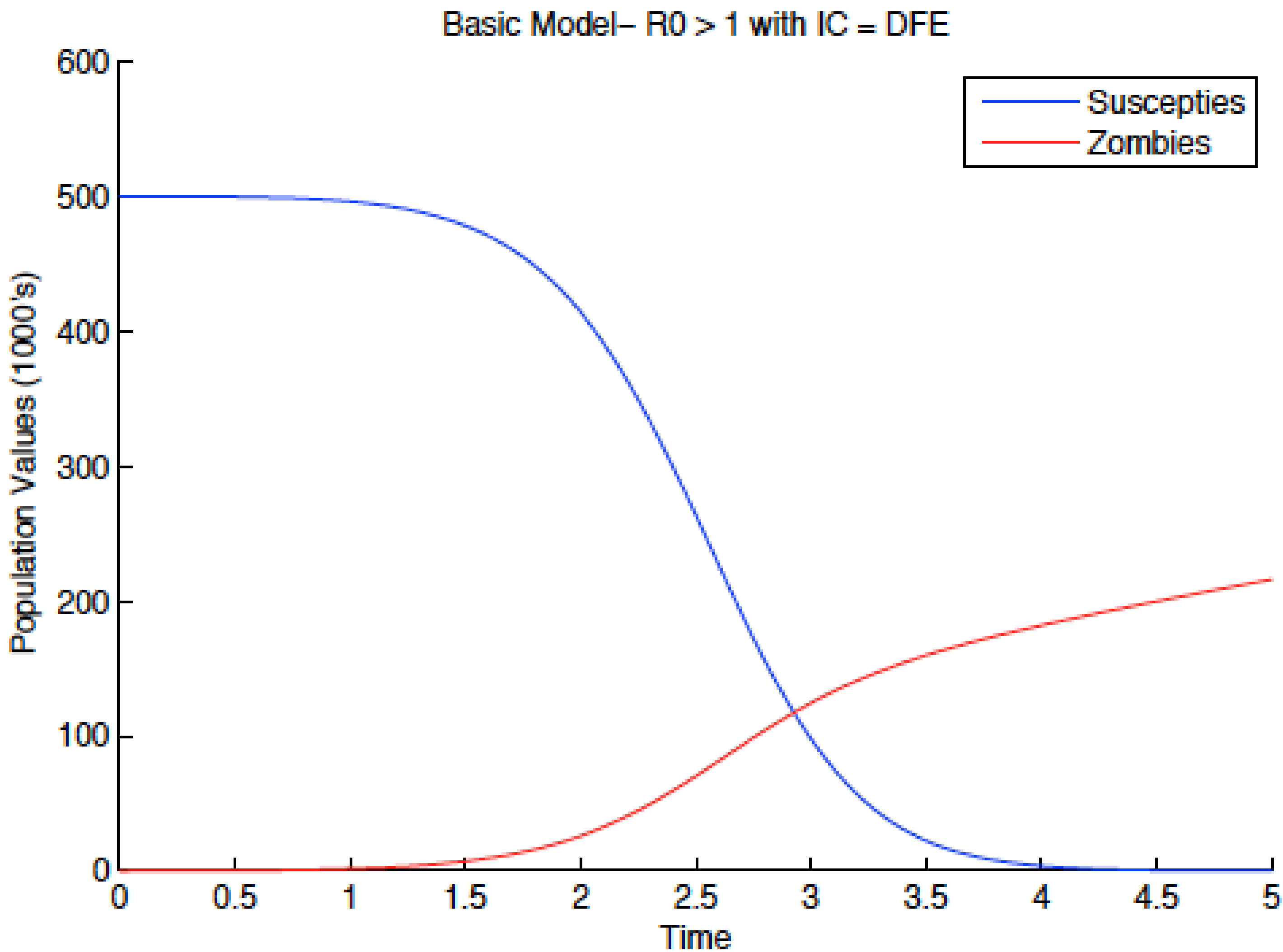
Computing with SZR

$$
\begin{aligned}
S' &= \Pi - \beta SZ - \delta S \\
Z' &= \beta SZ + \zeta R - \alpha SZ \\
R' &= \delta S + \alpha SZ - \zeta R \,.
\end{aligned}
$$

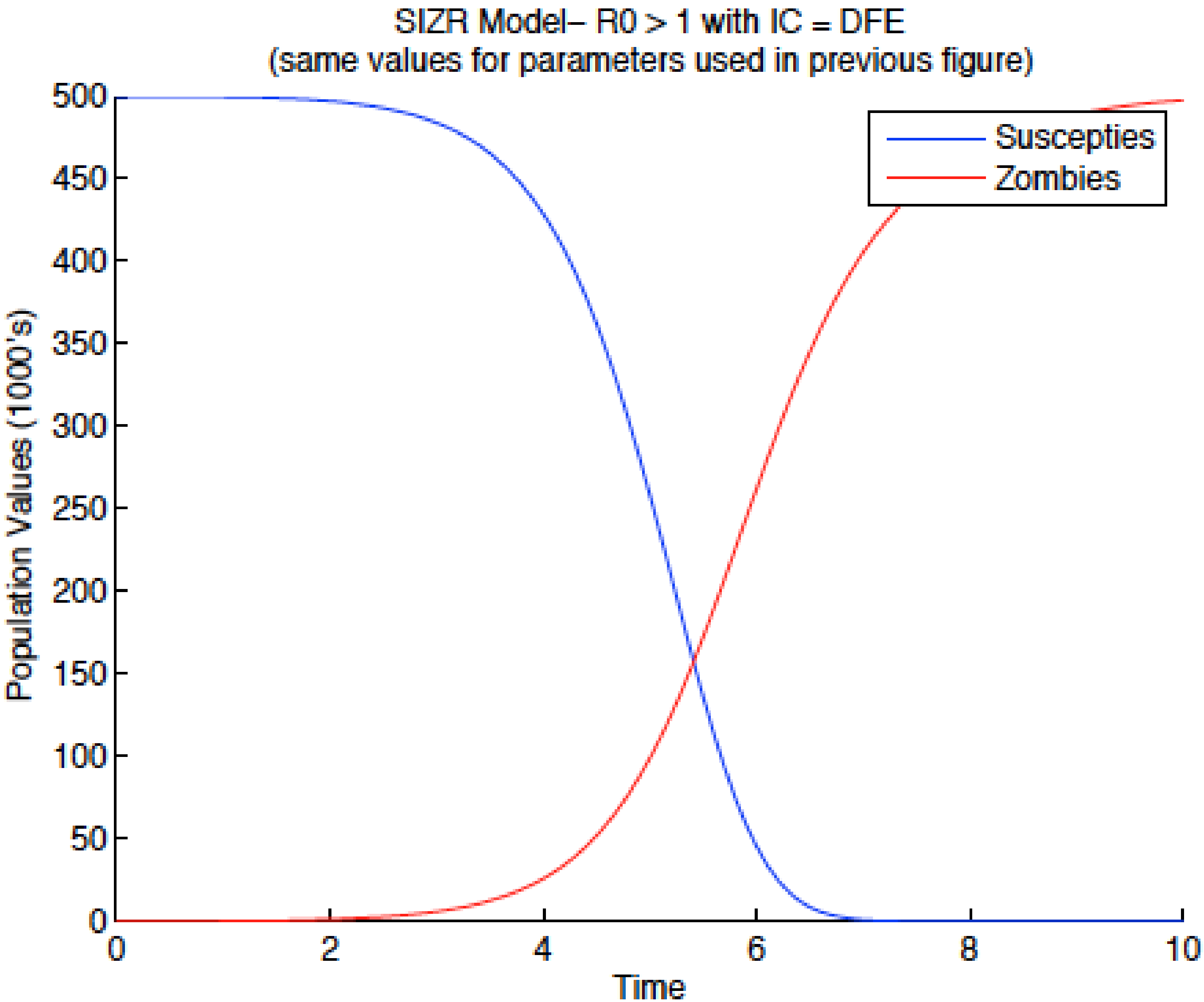
- Short timescale: (no births / natural deaths):

$$
\begin{aligned}
-\beta SZ &= 0 \\
\beta SZ + \zeta R - \alpha SZ &= 0 \\
\alpha SZ - \zeta R &= 0 \,.
\end{aligned}
$$

Using Euler's Method



Model with Latent Infection



Alternative Zombie Sim

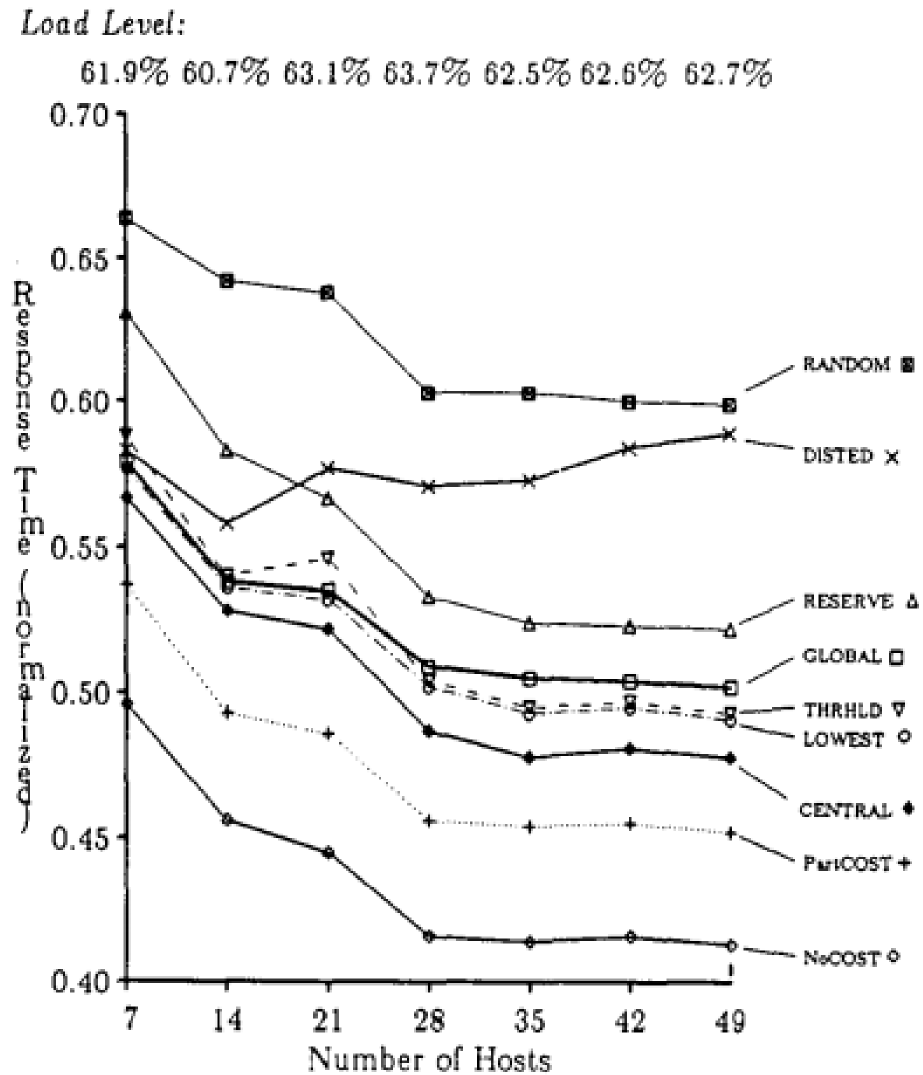
- http://kevan.org/proce55ing/zombies/

Event-Driven Examples

Event-Driven Simulation

- Applications:
  - Traffic during rush hour: effect of different algorithms for controlling traffic lights
  - Load on web server: effect of more machines, scheduling algorithms, etc.

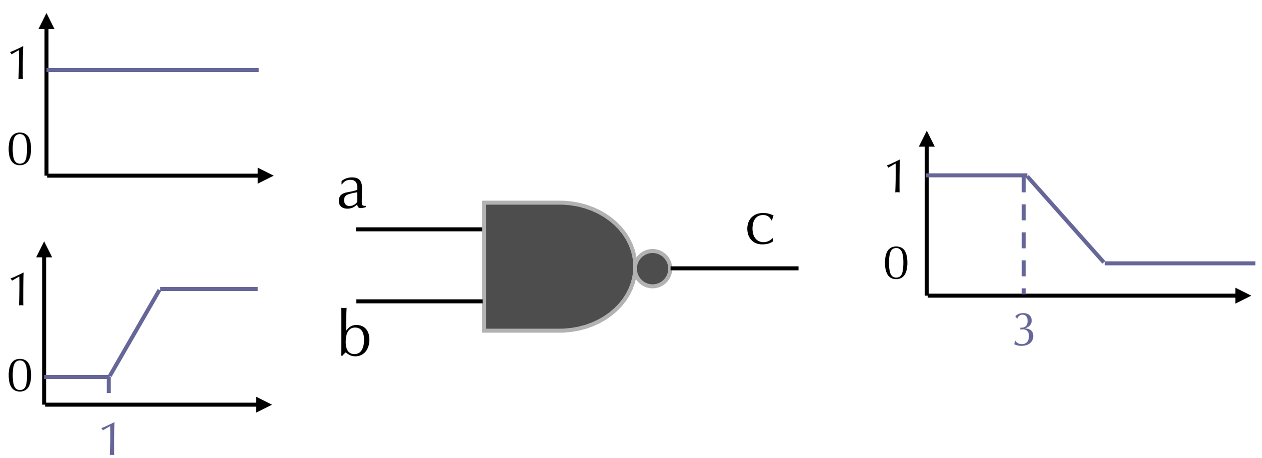
Example: Load Balancing Across Hosts



From Zhou, S. 1988. "A trace-driven simulation study of dynamic load balancing." *IEEE Trans. Software Eng.* 14(9).

Event-Driven Simulation

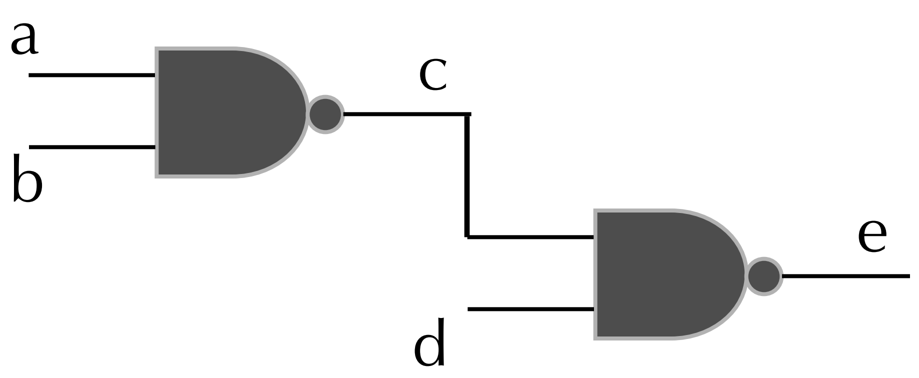
- Applications:
  - Circuit/chip simulation: clock rate needed for reliable operation



Events:
- Input: b(1)=1
- Output: c(3)=0

Event-Driven Simulation

- Applications:
  - Circuit/chip simulation: clock rate needed for reliable operation



Ingredients of Event-Driven Simulations

- Event queue
  - Holds (time, event) tuples
  - *Priority queue* data structure: supports fast query of event with lowest time
  - Possible implementation: linked list
    $O(n)$ insertion, $O(1)$ query, $O(1)$ deletion
  - Possible implementation: heap, binary tree
    $O(\log n)$ insertion, $O(1)$ query, $O(\log n)$ deletion
    (see COS226 for details)

Ingredients of Event-Driven Simulations

- Event loop
  - Pull lowest-time event off event queue
  - Process event
    - Decode what type of event
    - Run appropriate code
    - (Compile statistics)
    - Insert any new events onto queue
  - Repeat.

Ingredients of Event-Driven Simulations

- How are new events scheduled?
  - Some are a direct result of current event.
    Example: teller takes new customer
  - Some are background events.
    Example: new customer arrives
  - Some are generated via real-time user input

Stochastic Simulation

- Events have different likelihoods of occurrence
  - New customer arrives
  - Person contracts disease
- Properties of simulation components may vary
  - Bank customers may have more or less difficult problems
  - Drivers may be more or less polite
  - Individuals may be more or less susceptible to disease

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

--- John von Neumann (1951)

What Did von Neumann Mean?

- Distinguish between “random” and “pseudorandom”
- Big advantage of pseudorandom: repeatability
- Big disadvantage: not really random

Using RNGs

How would you…

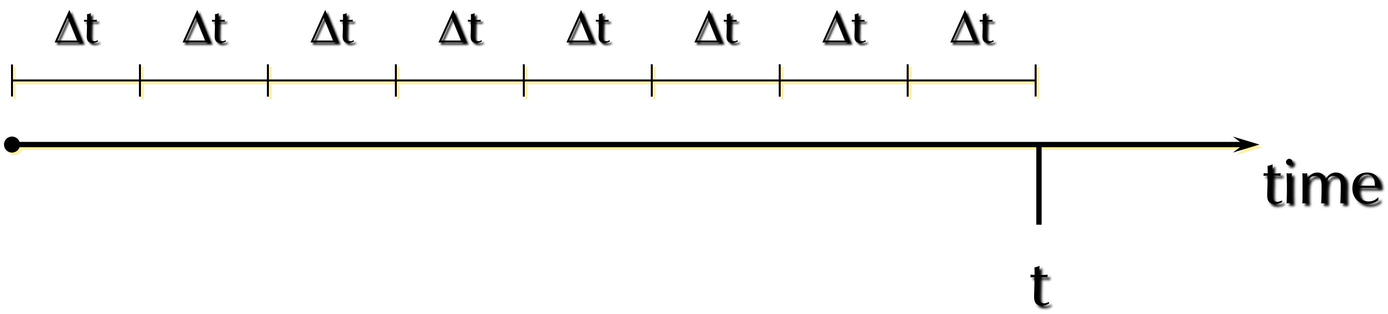
- Choose an integer i between 1 and N randomly
- Choose from a discrete probability distribution; example: $p(heads) = 0.4$, $p(tails) = 0.6$
- Pick a random point in 2-D: square, circle
- Shuffle a deck of cards

Bank Simulation: Scheduling Arrival Events

- Given time of last customer arrival, how to generate time of next arrival?
- Assume arrival rate is uniform over time: k customers per hour
- Then in any interval of length Δt, expected number of arrivals is $k\,\Delta t$

Scheduling Arrival Events

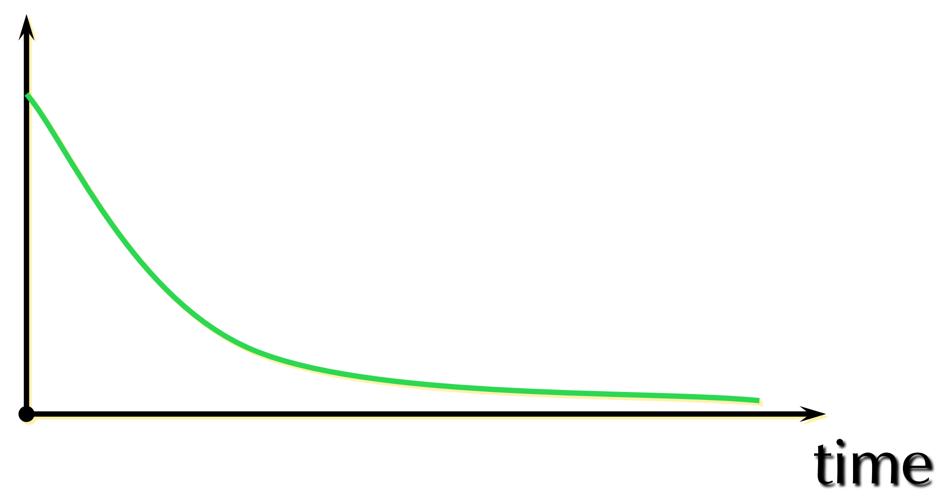
- Probability distribution for next arrival?
  - Equal to probability that there are no arrivals before time t
  - Subdivide into intervals of length Δt



p(no arrivals before t) = p(no arrival between 0 and Δt) * p(no arrival between Δt and $2\Delta t$) * …

Scheduling Arrival Events

- p(no arrival in interval) = $1 - k\,\Delta t$
- So, p(no arrivals before t) = $\lim_{\Delta t \to 0} \left(1 - k\,\Delta t\right)^{\frac{t}{\Delta t}} = e^{-kt}$

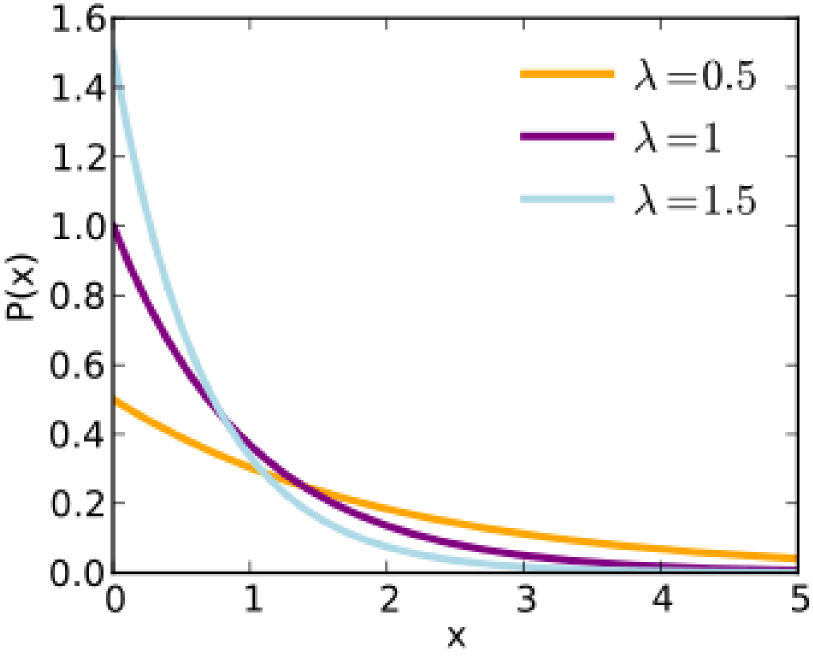


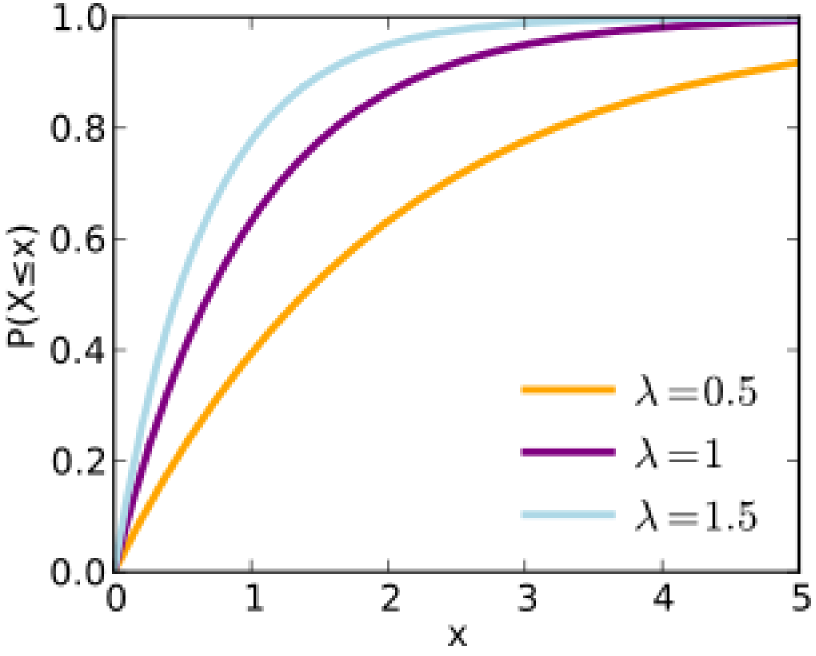
Exponential Distribution

- The exponential distribution describes the time in between events in a *Poisson process*

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0, \\ 0, & x < 0. \end{cases}$$

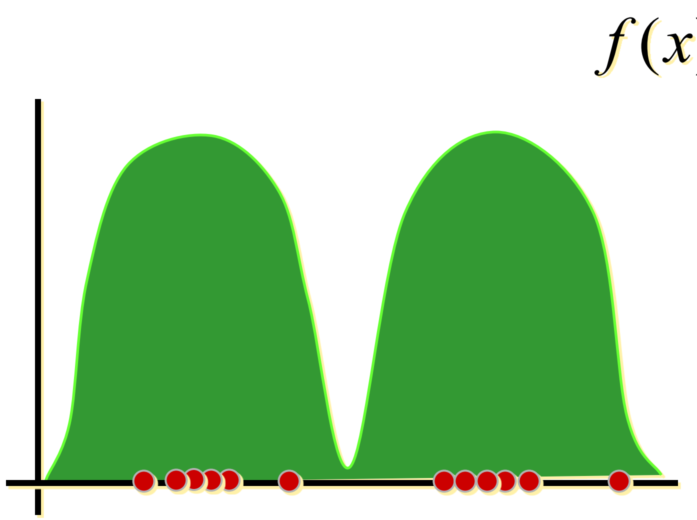
$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x}, & x \ge 0, \\ 0, & x < 0. \end{cases}$$

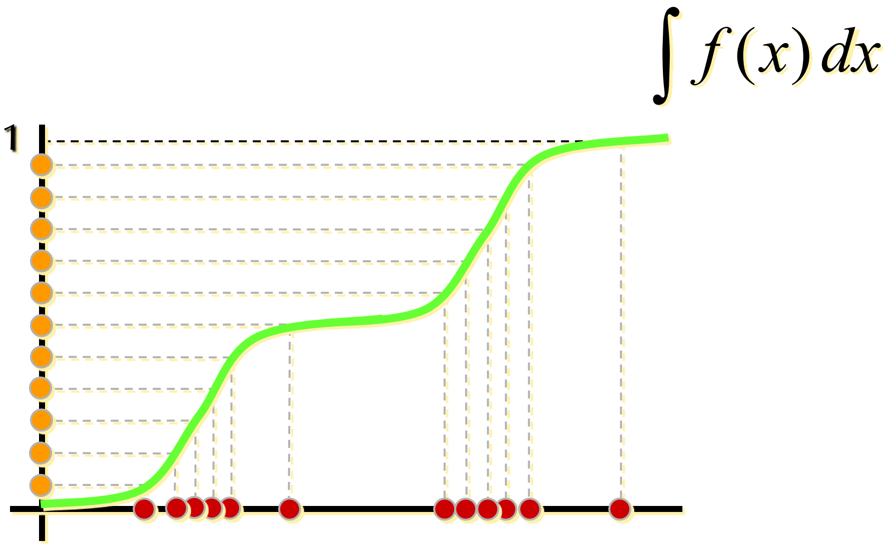




Sampling From a Non-Uniform Distribution

- "Inversion method"
  - Integrate $f(x)$: Cumulative Distribution Function
  - Invert CDF, apply to uniform random variable





Scheduling Arrival Events

- So: Normalize, integrate, invert

→ time to next arrival event can be found from uniform random variable $\xi \in [0..1]$ via

$$t_{next} = t_{last} - \frac{\ln \xi}{k}$$

Ingredients of Event-Driven Simulations

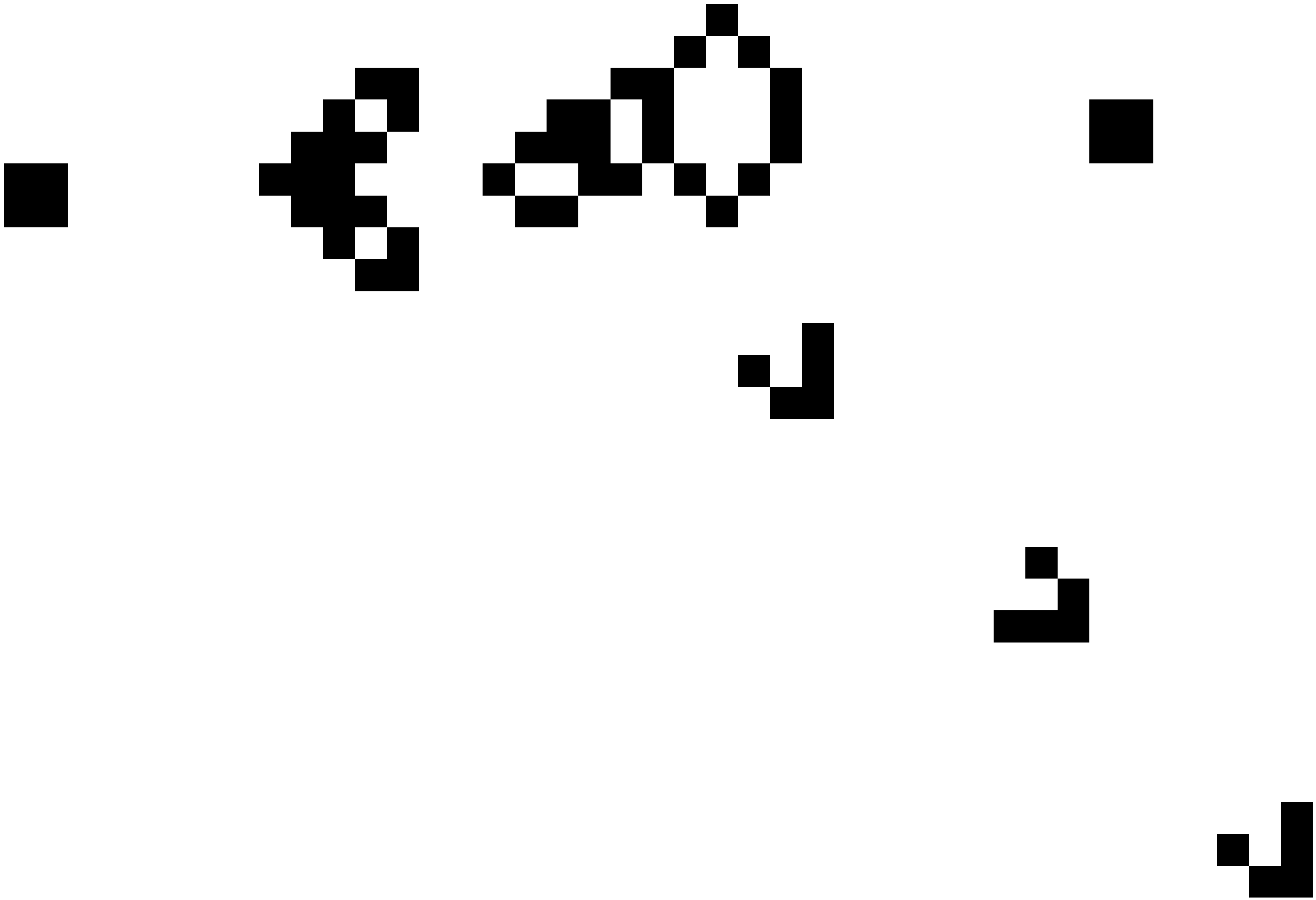
- How are events handled?
  - Need to run different piece of code depending on type of event
  - Code needs access to data: which teller? which customer?
  - Original motivation for Object-Oriented Programming languages: encapsulate data and code having a particular interface
  - First OO language: Simula 67

Summary

- Insert events onto queue
- Repeatedly pull them off head of queue
  - Decode
  - Process
  - Add new events

CAs, Microsimulation and Agent-based Simulation

(Micro-level behaviors leading to emergent macro-level phenomena)



http://upload.wikimedia.org/wikipedia/commons/e/e
5/Gospers_glider_gun.gif

Cellular Automaton

- Discrete-time, discrete-space, finite-state model
- Each cell's new state is a function of its previous state and the previous states of its neighbors
  - Typically instantaneous updates, same rules for all cells
  - e.g., for Conway's Game of Life, death if 0, 1, or >3 living neighbors, survival if 2, birth if 3
- Can lead to complex phenomena: ordered or chaotic, self-replicating, Turing machines (!), etc.

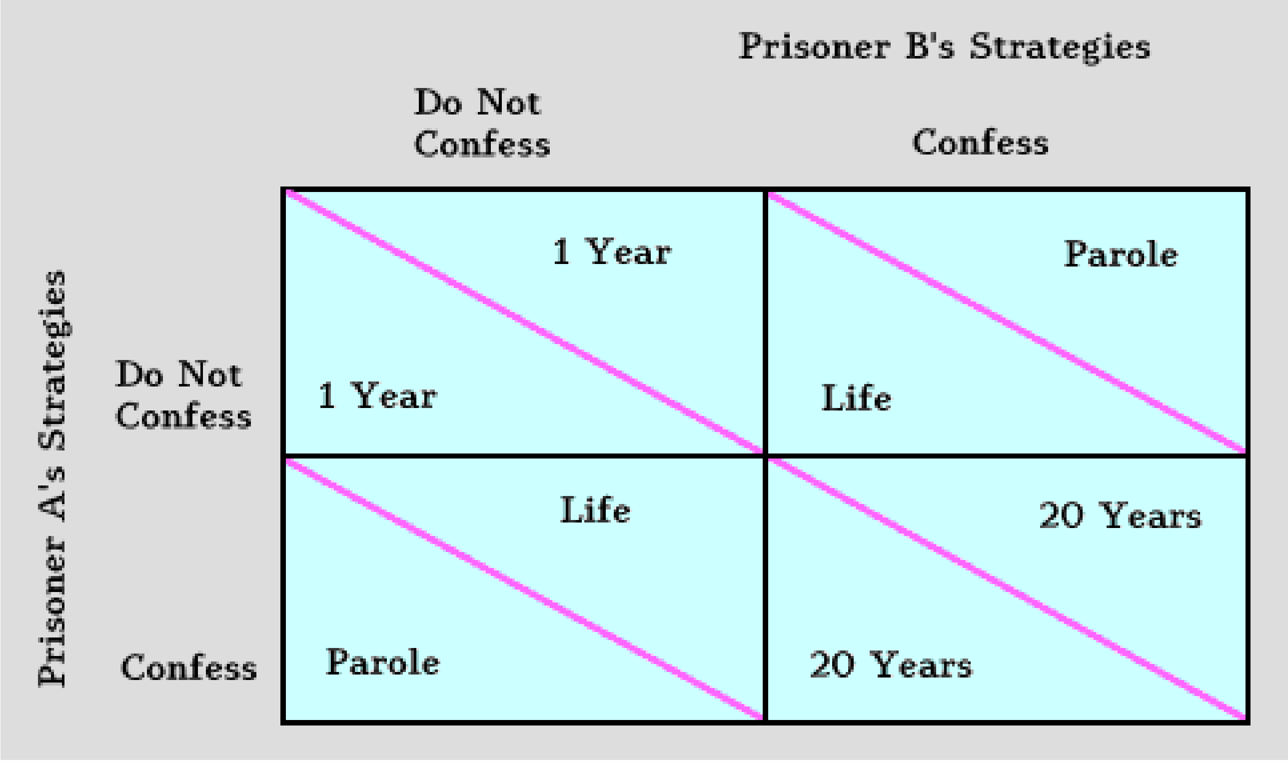
Microsimulation

- Model components of system as independent entities with differing characteristics
  - e.g., different susceptibility to disease
- Behavior is governed by particular rules
- Useful in traffic, health, econometrics (e.g., taxation)
- Demo:
  - http://www.traffic-simulation.de/

Agent-Based Modeling

- Accommodates interdependencies, adaptive behaviors
- E.g., “The evolution of cooperation”

The Prisoner's Dilemma



- Globally optimal: Neither confesses
- Game-theoretically optimal strategy: Always confess

The Evolution of Cooperation

- Robert Axelrod: A tournament for simulations to play with each other in repeated rounds
- Winner: cooperate first, then tit for tat (TFT)
- All top strategies are "nice"
- Necessary conditions for success:
  - Be nice
  - Be provocable
  - Don't be envious
  - Don't be too clever

A Better Strategy

- Jennings et al. in 2004 tournament:
  - Submit multiple prisoners and collude

Simulation: Pros and Cons

- Pros:
  - Building model can be easy (easier) than other approaches
  - Outcomes can be easy to understand
  - Cheap, safe
  - Good for comparisons
- Cons:
  - Hard to debug
  - No guarantee of optimality
  - Hard to establish validity
  - Can't produce absolute numbers

Simulation: Important Considerations

- Are outcomes statistically significant? (Need many simulation runs to assess this)
- What should initial state be?
- How long should the simulation run?
- Is the model realistic?
- How sensitive is the model to parameters, initial conditions?

Finally…

ARE YOU LIVING IN A COMPUTER SIMULATION?

BY NICK BOSTROM

[Published in *Philosophical Quarterly* (2003) Vol. 53, No. 211, pp. 243-255. (First version: 2001)]

This paper argues that *at least one* of the following propositions is true: (1) the human species is very likely to go extinct before reaching a "posthuman" stage; (2) any posthuman civilization is extremely unlikely to run a significant number of simulations of their evolutionary history (or variations thereof); (3) we are almost certainly living in a computer simulation. It follows that the belief that there is a significant chance that we will one day become posthumans who run ancestor-simulations is false, unless we are currently living in a simulation. A number of other consequences of this result are also discussed.

Implications

- Who designed it all?
- How should we behave?
- What if we start running too many of our own simulations?