SWEN90004
Modelling Complex Software Systems

Lecture Cx.06
Cellular Automata II: Implementing a Model

Nic Geard
ngeard@unimelb.edu.au

Semester 1, 2019

Recap

Cellular Automata (CA) as a formalism for modelling complex systems.

- Compared to the ODE models, they are bottom up rather than top down.
- Complex system characteristics:
  - many interacting parts
  - parallel distributed processing
  - state updates are determined by local rules and interactions
  - complex global behaviour emerges
- Can be suited for modelling systems where "macro-rules" are hard to identify.
- But, not straight forward to analyse behaviour.
- Good for testing questions such as: Can local mechanism X generate phenomenon Y?

Objectives

- recognise some of the key decisions that need to be made when implementing a CA model
- understand the steps involved in specifying a CA model based on an ODE model
- appreciate the 'creative' aspects of model design

Outline

Key questions: Space

How is space represented?

How are agents located in space?

Consider:

- discrete vs continuous space
- single vs multiple occupancy of cells
- proximate vs long-range interactions

Key questions: Time

How is time represented?

How does the order of events affect behaviour?

Consider:

- the order in which the state of each cell is updated
  - will every component of the system be updated simultaneously? (*synchronous updating*)
  - or will they be updated one-at-a-time? (*asynchronous updating*)
  - if so, in what order will the components be updated?
- discrete vs continuous time
  - discrete: check what happens at each time step
  - continuous: what event happens next and when does it happen?

Key questions: Information

What information do components (cells/agents) use?

How do they obtain it?

Consider:

- the scope of variables: global, cell (patch), agent (turtle)
- how these variables are accessed and modified
- the (spatial) range of sensing: what happens as this is increased? (ie, what is the *neighbourhood*)

Key questions: State updating

Will updating be deterministic or stochastic?

How does this decision affect behaviour?

- will the future state of a component be uniquely specified by its current inputs, environment, etc? (*deterministic updating*)?
- or will their be some randomness involved? (*stochastic updating*)

Outline

The Lotka-Volterra model (from lecture Cx.03)

Prey (rabbits):

$$\frac{dR}{dt} = \alpha R - \beta RF$$

Predators (red foxes):

$$\frac{dF}{dt} = \delta RF - \gamma F$$

Parameters:

- α = growth rate of the rabbit population
- β = rate at which foxes predate upon (eat) rabbits
- δ = growth rate of the fox population
- γ = decay rate of the fox population due to death and migration

How could you build a CA model of the fox-rabbit population dynamics?

A CA version of the Lotka-Volterra model

- Rather than modelling the size of each population (the *macro-variable*) directly, we will represent each individual explicitly, and then *observe* (measure) the size of the population.
- Space: Rather than assuming our populations are "well mixed", we will represent the location of each individual on a 2D regular lattice (grid).
- Space: We will make a new assumption that only one animal can occupy a grid space at any point in time.
- Time: Rather than treating time continuously, we will use discrete time steps.
- Information: We will assume that each individual animal is only aware of its immediate neighbours.
- State updating: ... (next slides)

A CA version of the Lotka-Volterra model

Representing our system

Each cell in our 2D lattice will take one of three values:

$$x_{i,j} = \begin{cases} 0 & \text{if site } (i,j) \text{ is empty} \\ 1 & \text{if site } (i,j) \text{ contains a rabbit (prey)} \\ 2 & \text{if site } (i,j) \text{ contains a fox (predator)} \end{cases}$$

	0	1	2	3	4
0	0	2	0	1	0
1	0	1	0	0	0
2	1	0	1	1	0
3	0	0	2	0	1
4	0	1	0	0	1

A CA version of the Lotka-Volterra model

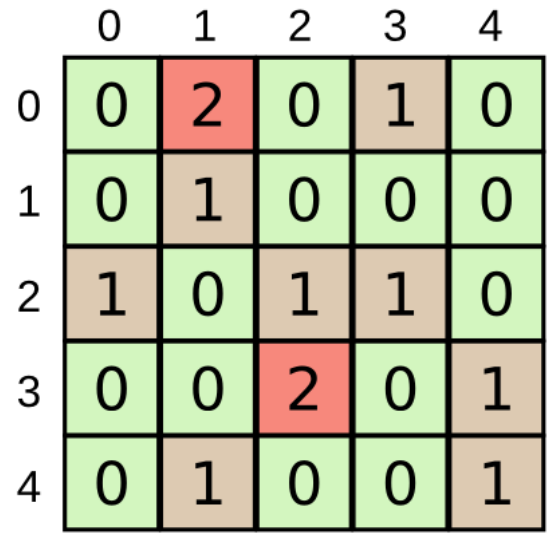
Update rules

1. Pick a cell (A) from our lattice at random
2. Choose one of A's neighbours (B) at random
3. Update as follows:
   - if A contains a rabbit and B is empty, then, with probability α, the rabbit reproduces, B now contains a new rabbit—prey reproduction
   - if A contains a fox and B contains a rabbit, or if A contains a rabbit and B contains a fox, then with probability β the rabbit is eaten; and, if this happens, with probability δ, the cell formerly containing a rabbit now contains a new fox—prey death and predator reproduction
   - if A contains a fox and B is empty, then, with probability γ, the fox dies—predator death
   - if A is empty and B contains a fox or a rabbit, then the adjacent animal moves from B to A—animal movement

A CA version of the Lotka-Volterra model

Update rules—prey reproduction

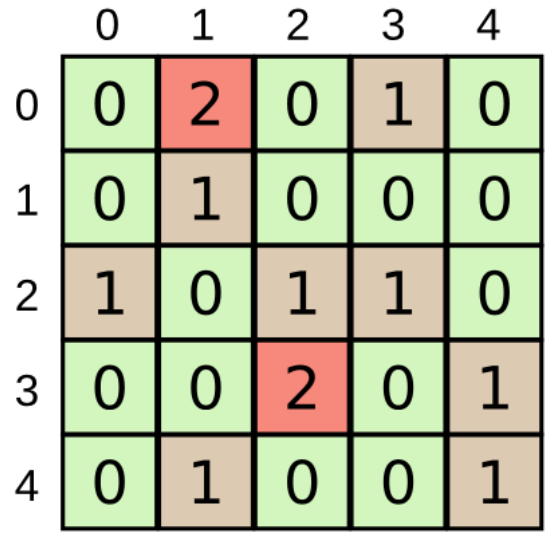
If A contains a rabbit and B is empty, then, with probability α, the rabbit reproduces, B now contains a new rabbit.



A CA version of the Lotka-Volterra model

Update rules—predation (prey death & predator reproduction)

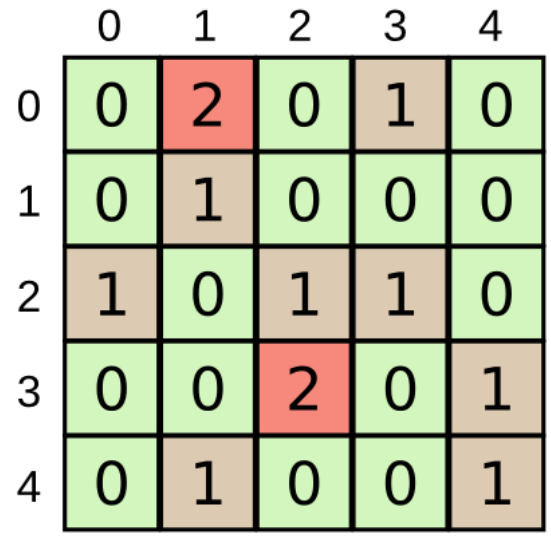
If A contains a fox and B contains a rabbit, or if A contains a rabbit and B contains a fox, then, with probability β, the rabbit is eaten; and, if this happens, then with probability δ, the cell formerly containing a rabbit now contains a new fox.



A CA version of the Lotka-Volterra model

Update rules—predator death

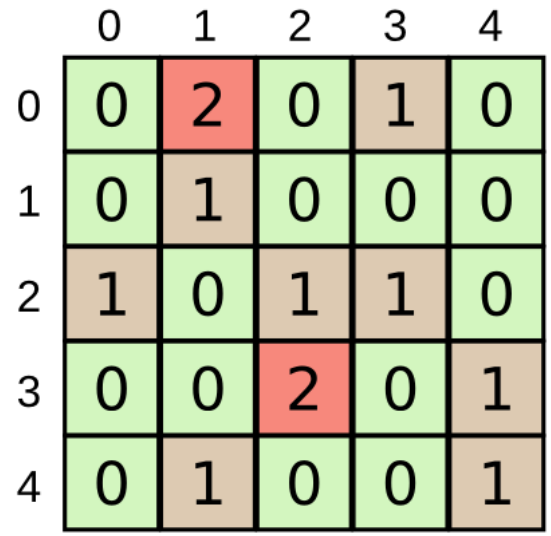
If A contains a fox and B is empty, then, with probability γ, the fox dies.



A CA version of the Lotka-Volterra model

Update rules—movement

If A is empty and B contains a fox or a rabbit, then the adjacent animal moves from B to A.



Outline

The SIR disease model (from lecture Cx.04)

$$\frac{dS}{dt} = -\beta SI$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

State variables:

- S = the fraction of the population who are currently susceptible
- I = the fraction of the population who are currently infectious
- R = the fraction of the population who are currently recovered
- Note: therefore $S + I + R = 1$

Parameters:

- β = the rate of *effective* contact between susceptible and infectious people
- γ = the rate of recovery of infectious people

A CA version of the SIR disease model

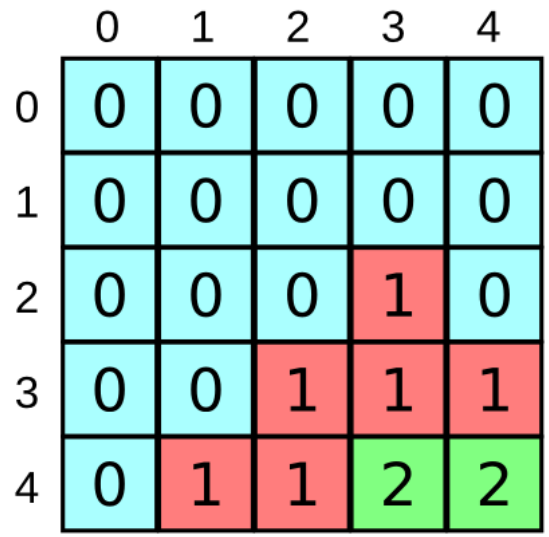
- Rather than measuring the fractions of the population in each disease compartment (the *macro-variable*) directly, we will represent each individual explicitly, tagged with their current disease status, and then *observe* (measure) population fractions in each state.
- Again, rather than assuming our populations are "well mixed", we will represent the location of each individual on a 2D regular lattice (grid).
- Again, rather than treating time continuously, we will use discrete time steps.
- We will make the new assumption that only one person occupies a grid cell at once, and that people's location is fixed over time (ie, they don't move!)
- Initially, a fraction p of the population is infectious, and the remainder of the population is susceptible.

A CA version of the SIR disease model

Representing our system—Version 1

Each cell in our 2D lattice will take one of three values:

$$x_{i,j} = \begin{cases} 0 & \text{if the person at site } (i,j) \text{ is susceptible} \\ 1 & \text{if the person at site } (i,j) \text{ is infectious} \\ 2 & \text{if the person at site } (i,j) \text{ is recovered} \end{cases}$$



A CA version of the SIR model

Update rules—Version 1

1. A susceptible person can be infected by an infectious neighbour with probability β
2. An infectious person recovers with probability γ

A *different* CA version of the SIR model

Note that for any real world system, we may have a *choice* of multiple possible models

A variation—Version 2

- As before, each site can be susceptible (S), infectious (I) or recovered (R) to a disease.
- Now, immunity is temporary, and people become susceptible again at a rate ω.
- Thus, on average, people are:
  - *infectious* for $q = \frac{1}{\gamma}$ days
  - *recovered* (immune) for $r = \frac{1}{\omega}$ days

A *different* CA version of the SIR model

Representing our system—Version 2

Each cell can now take a value in the range $[0, q + r]$ as follows:

$$x_{i,j} = \begin{cases} 0 & \text{if the person at site } (i,j) \text{ is susceptible} \\ 1, \ldots, q & \text{if the person at site } (i,j) \text{ is infectious} \\ q+1, \ldots, q+r & \text{if the person at site } (i,j) \text{ is recovered} \end{cases}$$

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	2	3	3
3	0	2	4	4	5
4	1	3	5	0	0

A *different* CA version of the SIR model

Update rules—Version 2

1. A susceptible person becomes infected if *at least one* of their neighbours is infectious
2. An infectious person recovers after q days
3. A recovered person loses their immunity and becomes susceptible after r days (waning immunity)

Exploring our model behaviour

Parameter sweeps

A common approach to characterising the behaviour of a model is to conduct a parameter sweep: systematically varying the value of each parameter to understand its effect on system behaviour.

For example (Version 2), run the model for each combination of:

- $p = 0.01, 0.05, 0.1$ (initial proportion of infectious people)
- $q = 2, 4, 8$ (number of days a person is infectious)
- $r = 2, 4, 8$ (number of days a person is recovered/immune)

ie, $3 \times 3 \times 3 = 27$ possible parameter combinations

Exploring our model behaviour

Parameter sweeps

Because our model behaviour is *stochastic*, we may want to run each parameter combination several times and consider the "average" behaviour of the system.

For trajectories (ie, sequences of state changes over time), this may be difficult. Typically, we identify some global measurement, record the value for each simulation run, and report the mean and standard deviation.

Outline

Extensions to the basic CA model

Asynchronous CA

The basic CA updates *all* cells synchronously (at the same time) at *every* time step. We could also update cells at different times.

Probabilistic CA

The basic CA update rules are deterministic (Cx.05 examples). We could also use update rules that are probabilistic/stochastic (Cx.06 examples).

Non-homogeneous CA

The basic CA applies the same update rule to every cell. We could also use context-sensitive rules.

Network-structured CA

The basic CA defines neighbourhood in terms of grid adjacency. We could also use a more complex *network* topology of neighbours (we'll look at this later)

Why use CA?

Advantages

- they are (relatively) simple and easy to implement
- they can represent *interactions* and *behaviours* that are difficult to model using ODEs
- they reflect the intrinsic *individuality* of system components

Disadvantages

- they are relatively constrained (topology, interactions, individual behaviour)
- the global behaviour may be difficult to interpret

Next week, we will relax some of these constraints further, and explore *agent based models* where the *actors* in a system are the primary focus.

Reading

- Daniel Shiffman, The Nature of Code, chapter 7 (available at http://natureofcode.com/book/chapter-7-cellular-automata/)
- Gary W Flake, The Computational Beauty of Nature, chapter 15
- Stephen A Wolfram, A New Kind of Science, Wolfram Media, 2002
- Langton, C. Studying artificial life with cellular automata. Physica D 22:120-149, 1986
- Hoekstra, Alfons G.; Kroc, Jiri; Sloot, Peter M.A. (Eds.) Simulating Complex Systems by Cellular Automata. Springer. 2010
- Joel L Schiff. Cellular automata : a discrete view of the world. Wiley-Interscience. 2008