A brief introduction to Agent-Based Modeling

John Pepper
University of Arizona
and
Santa Fe Institute

You might ask...

- What is modeling good for?
- How is modeling different in the field of Complex Systems?
- What do you mean by "agent-based modeling"?
  - How does it work?

Saved for later:

- Some examples of agent-based models
- How to build an agent-based model

Models are central to all of science

- We use them to formalize our assumptions and our hypotheses about how systems work.
- They often help us move from a hunch to a clear hypothesis, or to an empirically testable prediction.

Two approaches to modeling:

- Top-down (more traditional)
- Describe system-level properties without explicit reference to mechanisms, including agent interactions
  - Usually mathematical or 'Equation-based',
    - may be solved analytically or numerically (e.g. by computer)
- Bottom-up (or 'generative')
- Model explicitly describes only agent properties and interactions, not system-level behaviors (these are *generated* by the model as emergent properties)
  - bottom-up = agent-based
    - Almost always implemented computationally

Modeling Complex Systems

- Complex systems tend to show patterns of organization that "emerge" out of the interactions of the units they are composed of.
- This property is often central to studies of complex systems.
- Often the behavior of the system cannot be easily inferred from the behavior of its components, or vice versa
- A key question: What are the causal links between micro-rules and macro-behaviors?

How to model emergence?

- Agent Based Modeling (ABM):
  - The basic unit of ABMs is **the agent**
  - Usually a model will contain many agents (sometimes many thousands)
  - Model outcomes are determined by agent interactions with environment and each other
  - Although the model only describes the agents and their behaviors, it is often the emergent system-level phenomena that we are interested in studying with ABM, not the individual agents we spend so much effort on describing.

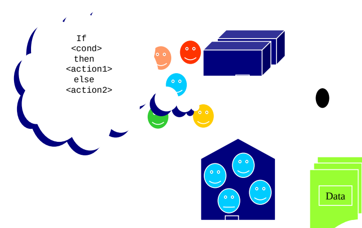
What is an agent?

- Generally speaking, an agent is an identifiable unit of computer program code which is :
  - ***autonomous:***
    - capable of independent action based on explicit rules that may be very simple, or more complicated
    -
  - ***a representation of the primary actor in the system***

What is an agent?

- Generally speaking, an agent is an identifiable unit of computer program code which is :
  - ***autonomous:***
    - capable of independent action based on explicit rules that may be very simple, or more complicated
    -
  - ***a representation of the primary actor in the system***

A cartoon view of an agent-based model:



What does an agent represent? (1)

- In many applications, an agent is an individual (person or organism)
  - In Population Biology and Ecology, ABM = IBM (‘individual-based model’)

- But this is not always the case…

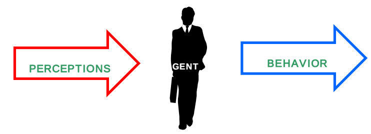
What does an agent represent? (2)

- In many situations in anthropology and social sciences, decisions are made and actions are taken by unified <u>families</u> or <u>households.</u>
  - Thus the *agents* in corresponding models represent households rather than individuals (examples coming from Dr. Kohler?…)

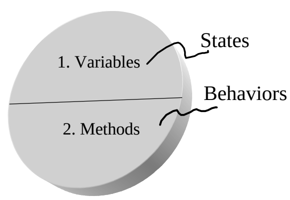
What does an agent represent? (3)

- To understand the origin and progression of cancer, we need to study the population dynamics and the evolution of <u>cells</u> within an individual
  - For this purpose, I use agent-based models in which the agents are *cells*.

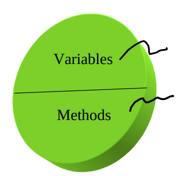
- An Agent is any component in an ABM that has:
  - Internal data representations (*states / instance variables*).
  - Means for sensing their states and their surroundings (*perceptions*).
  - Means for taking actions that may modify their internal state or external environment (*behaviors / methods*),. These actions are typically 'conditional' or 'contingent' – influenced by environment including other agents.



2 Components of an Agent



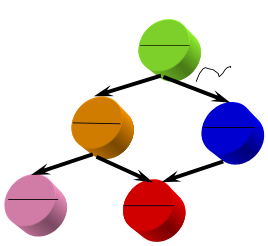
2 Components of an Agent



<u>Variables</u>
Age
Ability
Assets
Partnership
Debt

<u>Methods</u>
Work
Invest
Relocate
Pay expenses
Ask for business partner
Start another business
File Bankruptcy
Sell
Retire

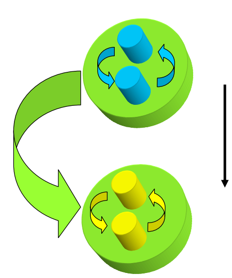
Interaction between agents takes place via messages



Messages from one agent may trigger methods in another, the results of which are often sent to other agents via additional messages.

Also, each agent modifies its own states via its own internal methods.

ABMs often use 'discrete event simulation'



Simulation proceeds in discrete time steps

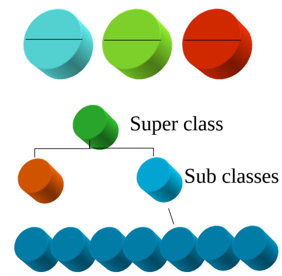
- But interactions between agents or procedures during a time step may have their own event schedules or particular order

Some technical terms regarding implementation

- Class
  - The definition of an object type
- Superclass
  - The root or basal class from which others inherit behaviors and variables
- Subclass
  - A class that inherits behaviors and variables from a superclass, but adds others
- Instance
  - An individual agent that has been created and exists in memory
- Method
  - A function (action/ behavior) that can be "called" by sending an appropriate message to an object.

The three principles of Object-oriented programming

- **Encapsulation**
  - Allows objects to hide their functions (**methods**) and data (**variables**) from each other
- **Inheritance**
  - Each **subclass** inherits all variables of its **superclass**
- **Polymorphism**
  - Multiple instances of same class, sharing behavior but not state or memory; this results in a population of heterogeneous agents



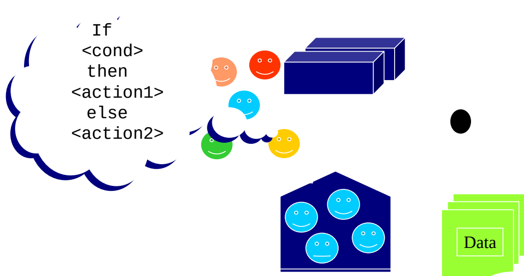
Agent Based Modeling

- Agent based models allow us to study
  - Spatial interaction
  - Adaptive, heterogeneous agents
  - Multilevel systems → economy, markets, firms, plants, employees
  - All of these aspects are often difficult to address with equation-based models

Agent Based Modeling…

- Bottom-up approach
- Generative vs. deterministic
- Can avoid some assumptions built into equation-based (ODE) models.
  - continuous quantities
  - Collection of homogenous entities
  - linear responses

Bottom up modeling



Three kinds of ABMs with different purposes:

- Minimal models for ideas

  Intended to explore a concept without reference to a particular species, place, etc.
- Minimal models for systems

  A simplified view of a particular kind of system.
- Synthetic models of systems

  A synthesis of detailed descriptions of all parts and processes in some system (an attempt to accurately emulate a system) – more common in management than basic research.
- \- Roughgarden 1996

Criteria for a good model:

- Simplicity
- Clarity
- Objectivity (freedom from bias)
- Tractability

Less is more…

- A simple model provides more clarity, tractability, more complete analysis, and easier communication.
- In most cases the real art of modeling lies in *leaving things out*!

Building a model: Overview

- An agent-based simulation typically proceeds in two stages.
  - The first stage is setup that prepares the simulation for running
  - The second stage is the actual running of the simulation. In most simulations the running of the simulation is divided into time steps or "ticks." With each tick some action occurs using the results of previous actions as its basis.
  - So, for example, in a prisoner's dilemma type simulation with two player types, setup would create instances of the two player classes. Such specialization is typically done via inheritance such that 'Player' becomes the super-class of your 'cooperator', 'defector' and 'tit-for-tat' classes.
  - Each tick or time step, each player would play the game (cooperate or defect) where their current play is dependent on their strategy and perhaps on the results of previous play.

Building a model: Classes

- Most ABM simulations typically have at least two classes. An agent class that describes the behavior of your agents (e.g. play a game by cooperating or defecting) and a model class that coordinates the setup and running of the model. (E.g., tracking time, arranging players into pairs, calculating the payoffs, etc.)
- Additional, more specialized classes can be created via inheritance such that 'Player' becomes the super-class of your 'cooperator', 'defector' and 'tit-for-tat' classes.

Acknowledgements

Thanks for some slides:

Luke Premo,

Max Planck Institute for Evolutionary Anthropology