Building a Hierarchical Metapopulation Model

Nathaniel Osgood
CMPT 858

November 6, 2012
Capturing Multi-Level Hierarchies: A Frequent Modeling Need

• We frequently have hierarchies of environments and actors
  – Region/Municipalities/Neighborhoods/Individuals
  – Region/Municipalities/Schools/Children

• As is widely recognized in multi-level modeling (e.g. HLM), these hierarchies frequently exert successive levels of influence

• Frequently these hierarchies are also associated with their own *structural* and *dynamic complexities*
One Example of Structural and Dynamic Complexities

• Municipalities may be arranged in transportation networks (road, rail, air connections)
• People may exhibit primarily contact patterns within a given municipalities
• People may move between municipalities
AnyLogic Flexibility

• While certain AnyLogic elements are most commonly associated with part of the project hierarchy, we can use them in many places, e.g.
  – Statecharts can be placed within “Main”
  – Populations of other agents can be placed within “Agents”

• Partly because of this flexibility, we can create structural hierarchies in AnyLogic that parallel the hierarchies in the world
Hands on Model Use Ahead

Load Previously Built Model: MinimalistNetworkABMModel

Suggest Saving as “HierarchicalCityPopulationModel”
One Example

• Main contains a population of Cities
  – Arranged in one type of network

• City contains a population of Persons
  – Arranged in another type of network
Copy “Person” Class
(Right Click on “Person” in “Project” Window and Choose “Copy”)
Click Right on Project Name & Choose “Paste”
Result
Rename “Person1” as “City”
Open “Main”
Click on “population”
Delete “population” (Right click & Choose “Delete”, or Use “Edit” menu)
Click (once) on “City” and Drag into “Main” Canvas & Drop
Click on "city" in Main Canvas, set name to "cities", use "environment" and set replication to 10.
In “Advanced” tab in “Properties” window for “environment”, make sure that “Network Type” is “Distance Based”, and “Connection Range” is “250”
Double-click on “City” in “Project” Window
Adding a Municipal Population
Click (once) on “Person” and Drag into “City” Canvas & Drop
Click on “person” in “City” canvas, set name to “cityPopulation” & set replication to “uniform(10, 200)”. Result:
Add an Environment & Rename it “City Environment”
In “Advanced” properties of “Properties” window for “cityEnvironment”, set “Network Type” to “Scale free” and the “Width” and “Height” both to 75.
Click on “cityPopulation”, and set the “Environment” field to “cityEnvironment”. Result:
Run the Model
Change the Relative Size for Cities & People

Double-Click on “City”

Expand “Project” Window Hierarchy on left under “City” to get to “oval” (Under “presentation”)
Recenter on Origin
Model
Now Move the Origin for Placing People to Upper Left of City
Add a Statistic at the City Level
Add a Function the “Person” canvas. Name it “moveToRandomCity”. Set the “General” tab properties as follows
Set “Code” Tab for “moveToRandomCity” as follows

```java
City myCity = this.get_City();

int countConnectedCities = myCity.getConnectionsNumber();
int randomConnectedCityIndex = uniform_discr(0, countConnectedCities - 1);
trace("Moving to city " + randomConnectedCityIndex);
City cityDestination = (City) myCity.getConnectedAgent(randomConnectedCityIndex);

trace("this + " just moved from city " + myCity + " to " + cityDestination);
moveBetweenSpecifiedCities(myCity, cityDestination);
```
Add a Function to “Person” canvas, named “moveBetweenSpecifiedCities”. Set “General” tab properties as follows.
Code for New Function

cityFrom.remove_cityPopulation(this);
cityTo.add_cityPopulation();
cityTo.cityEnvironment.applyNetwork();
Add an Event to the “Person” canvas. Name it “desiringIntercityMove”
Run the Model. People should now be moving around between the Cities.
Extension 1: Adding Name Labels to Cities
Add “Name” Parameter to City
Establishing a Text Field (Label) for Cities
Set Dynamic “Name” Text Field Property
Establishing Collection (Array) of City Names

Imports section:

Setting the “Name” Parameters of Successive Cities Upon Creation
Running
Extension 2: Extending the Model with Infection Spread (Note that this does not build on Extension 1)
Hands on Model Use Ahead

Load Previously Built Model: “HierarchicalCityPopulationModel”
Suggest Saving as “HierarchicalCityPopulationModelWithInfectionSpread”
In “Person”, add a Statechart Entry Point
Add a “Susceptible” State
Add an “Infective” State
Add a “Recovered” State
Neaten Up (as Aesthetics Require)
Add a “Recovery” Flow
Rate is 0.1 (implying mean residence time of 10)
Add “InfectionIncidence” transition.
In “Agent” Properties for “Person”, Route the message to the “infectionStatechart”
Add an “exposureTransition” to Spread Infection
(This sends an “Infection” message every time unit)
Add a Variable called “color”
This is of type “Color”, and should have initial value “Black”
Set Color for “Infective” State to Green
Set Color for “Infective” State to Red
Set Color for “Recovered” State to Gray
Set the Formula for Person’s Oval “Fill Color” Property to be “color”
Run the Model

The Infection Spreads locally, and a bit between Cities
Add a "meanRecoveryTime" parameter to "Person"
Set the “Recovery” Rate to the Reciprocal of “meanRecoveryTime”.

Note that “meanRecoveryTime” lives in “Main”. To get a reference to “Main”, we first have to get a reference to our enclosing city, and from the city request a reference to “Main”.

Rate: \( \frac{1}{\text{this.get\_City()}.\text{get\_Main()}.\text{meanRecoveryTime}} \)
Note that Experiments now have a “meanRecoveryTime” parameter.
Request Creation of a New Experiment
Call the new Experiment “RecoveryTime100”
For this Experiment, Assume a "meanRecoveryTime" of 100
Click on the original Experiment (“Simulation”) Rename it to “RecoveryTime10”
Create a 3rd Experiment, called “Baseline”
Set “Baseline” to Assume a “meanRecoveryTime” of 200
Run the Baseline Experiment
Output from the Baseline Experiment
Set the “Stop Time” for the Final Experiment to “500” (in the “Model Time” tab for the Properties of the experiment)
Do the same for the “RecoveryTime10” Experiment
Do the same for the “RecoveryTime100” Experiment