Discrete Inter-agent Dynamics, Sending & Receiving Messages

Nathaniel Osgood
MIT 15.879

March 14, 2012
Discrete Agent Coupling via Messages

• Within AnyLogic, agents can be coupled by either discrete (instantaneous and individuated) or continuous (ongoing and gradual) coupling

• The preferred mechanism for discrete coupling is messages sent between agents
  – Many types of messages payloads are possible
  – An agent can send a given message to one or more agents
  – Frequently messages are sent locally to neighbors within the environment
    • Neighboring nodes on the network
    • Nearby neighbors in space
Messages & Statecharts

• Messages may be handled in many ways
• One of the most common ways in which messages are handled is by statecharts
  – A transition can be triggered (“guarded” or gated) by a message
  – A transition may be associated with an action that fires off a message to other agents (or to other statecharts within the agent)
Hands on Model Use Ahead

Load Sample model:
SIR Agent Based.alp
Open Up “Person” class
Receiving a Message

- In this case, only 1 message type exists, so the simple fact that a message has been received is sufficient; there is no need to inspect message contents.
Sending a Message

• (Self-transition because remains in state)
Message Sending

• Messages may be sent to either
  – A particular, explicitly specified agent
  – An implicitly specified class of agents
    • Neighboring agents in the environment topology
    • Random agents
    • All agents
    • Any connected agents
    • All connected agents

• Mechanism:
  – `send(Message, DestinationObject)`
  – `send(Message, AgentClassId)`
Synchronous vs. Asynchronous Delivery

• Messages may be sent in two ways
  – Via `send`: Asynchronous (scheduled)
    • Delivery occurs sometime after call to `send`
    • This is like sending a text message – it can be read later
  – Via `deliver`: Synchronous (immediately called)
    • Risks infinite loops in delivery (if destination also calls `deliver` in the reverse direction)
    • This is like calling the other person’s phone – you demand their attention immediately
Message Payloads

• Sometimes just the fact that a message has been sent provides us with all of the information we need

• Sometimes just distinguishing different message types is sufficient

• We will sometimes send messages with payloads of data that provide extra information, e.g.
  – The agent that sent the message (eg that is infecting us)
  – Particular parameters

• Can send messages different payload types
  – Boolean/int.double/String/Other (can specify class type)
Sending a Message with a String Payload

```plaintext
send("Infection", RANDOM_NEIGHBOR);
```
Sending a Message with Object Payload

A reference to the Person sending the message!
This would allow the receiver to know who sent the message (Info also available in other ways)
Receiving a Message: Forwarding Messages on to the Statechart

The action for handling received messages delegates to the Statechart object.
Receiving a Message
Hands on Model Use Ahead

Load Previous Built [& Provided] Model: MinimalistNetworkABMMModel
Sending Messages
Using a “Contact” Event to Spread Infection

Add this transition

```java
send("Infection", RANDOM_CONNECTED);
```
Transition Type: Message Triggered
Making Infection Depend on a Message

Make sure you have selected the transition by clicking on it!
Setting “Person” so forwards Infection Message to Statechart

Make sure the “Agent” Tab is selected!
Setting Startup Code So Initially Infects a Random Person (so start with 1 infective)
Infection Percolation over the Network