

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

The screenshot displays the AnyLogic University interface for an agent named 'Person'. The main workspace shows a statechart with three states: 'Susceptible' (yellow), 'Infectious' (red), and 'Recovered' (grey). A transition labeled 'Infection' connects 'Susceptible' to 'Infectious', and a transition labeled 'Recovery' connects 'Infectious' to 'Recovered'. A variable 'color' is also present. The Properties panel for the 'Infection' transition is open, showing the following configuration:

- Name: Infection
- Show name:
- Ignore:
- Show name:
- Triggered by: Message
- Message type: boolean int double String Other
- Class name: Object
- Fire transition: Unconditionally If message equals If expression is true
- Action: (empty field)
- Guard: (empty field)

Sending a Message

- (Self-transition because remains in state)

The screenshot shows the AnyLogic University interface. The main workspace displays a statechart for a 'Person' agent. The statechart starts at a 'statechart' entry point leading to the 'Susceptible' state (yellow rounded rectangle). A transition labeled 'Infection' leads to the 'Infectious' state (orange rounded rectangle). From the 'Infectious' state, a transition labeled 'Recovery' leads to the 'Recovered' state (grey rounded rectangle). A self-transition labeled 'Contact' is shown on the 'Infectious' state, with a blue arrow indicating the transition. A variable 'color' is shown as a yellow circle with a 'V' next to it.

The 'Properties' panel at the bottom right is open to the 'Contact - Transition' section. The 'General' tab is selected. The 'Name' is 'Contact'. The 'Triggered by' is 'Rate'. The 'Rate' is set to `get_Main().ContactRate / get_Main().InfectionProb...`. The 'Action' is `send("Infection", RANDOM_NEIGHBOR);`. The 'Guard' is empty.

The 'Projects' panel on the left shows a tree view of the project structure. Under 'Statecharts', there is an 'infectionStateChart' folder containing 'susceptible', 'infection', 'infective', 'beyondInfection', 'branch', 'recovery', 'death', and 'finalState'. Under 'Presentation', there are 'BigPopulation: Main', 'Simulation: Main', and 'SIR Agent Based*' folders. Under 'SIR Agent Based*', there are 'Main' and 'Person' folders.

The 'Problems' panel at the bottom left shows 'No problems' and a table with columns 'Description' and 'Locat...'. The table is currently empty.

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

The screenshot displays a statechart editor with a statechart diagram and a properties panel for a transition.

Statechart Diagram:

```
statechart
    state Susceptible
    state Infectious
    state Recovered
    Susceptible --> Infectious : Infection
    Infectious --> Recovered : Recovery
```

The diagram shows a statechart with three states: Susceptible (yellow oval), Infectious (orange rounded rectangle), and Recovered (grey rounded rectangle). A transition labeled "Infection" connects Susceptible to Infectious, and a transition labeled "Recovery" connects Infectious to Recovered. A variable "color" is shown as a yellow circle with a "V" next to it.

Properties Panel (Contact - Transition):

- Name:** Contact
- Triggered by:** Rate
- Rate:** `get_Main().ContactRate / get_Main().InfectionProbability`
- Action:** `send("Infection", RANDOM_NEIGHBOR);`
- Guard:**

The "Action" field is highlighted with a red box, showing the code for sending a message with a string payload.

Sending a Message with Object Payload

The screenshot displays a statechart editor interface. On the left, a project tree shows a statechart for a 'Person' object with states 'Susceptible' and 'Infectious'. The main workspace shows a statechart with a transition labeled 'Infection' from 'Susceptible' to 'Infectious', and a transition labeled 'Recovery' from 'Infectious' to 'Recovered'. The 'Infectious' state has a 'Contact' transition. The 'Contact' transition is selected, and its properties are shown in the bottom panel.

Contact - Transition

General

Name: Contact Show Name Ignore Public Show

Description

Triggered by: Rate

Rate: `get_Main().ContactRate / get_Main().InfectionProbability`

Action: `send(this, RANDOM_NEIGHBOR);`

Guard:

A red arrow points to the `this` parameter in the `send` action, with the following text:

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 screenshot displays the AnyLogic Advanced interface. On the left, a project tree shows the hierarchy: environment > Embedded Objects > Analysis Data > Presentation > Person > Plain Variables > color > Statecharts > statechart > statechart > Susceptible. The main workspace shows a statechart with states: Susceptible (yellow), Infectious (orange), and Recovered (grey). Transitions are labeled: Infection (Susceptible to Infectious), InfectionTransmission (Infectious to Susceptible), and Recovery (Infectious to Recovered). The Properties window for the 'Person - Active Object Class' is open, showing the 'On Message Received' action set to `statechart.receiveMessage(msg);`. A blue arrow points from the statechart to the code, and a red arrow points from the code to the text below.

Person Main Person

statechart

color

Susceptible

Infection

Infectious

InfectionTransmission

Recovery

Recovered

Console Properties

Person - Active Object Class

General

Advanced

Agent

Parameters

Description

Environment defines initial location

Initial coordinates:

X:

Y:

Movement parameters:

Velocity:

Rotation:

On Arrival:

On Message Received: `statechart.receiveMessage(msg);`

On Before Step:

On Step:

Model

- Parameter
- Flow Aux ...
- Stock Vari...
- Event
- Dynamic ...
- Plain Vari...
- Collectio...
- Function
- Table Fun...
- Port
- Connector
- Entry Point
- State
- Transition
- Initial Stat...
- Branch
- History St...
- Final State
- Environm...

Problems

Description	Location

Selection

The **action** for Handling received messages delegates to the Statechart object

Receiving a Message

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a statechart diagram with three states: 'Susceptible' (yellow oval), 'Infectious Contact' (red rounded rectangle), and 'Recovered' (grey rounded rectangle). Transitions are labeled 'Infection' (from Susceptible to Infectious Contact) and 'Recovery' (from Infectious Contact to Recovered). A variable 'color' is shown in the top left.

The 'Infection - Transition' configuration panel is open, showing the following settings:

- Name: Infection
- Show Name:
- Ignore:
- Public:
- Show:
- Triggered by: Message
- Message type: boolean int double String Other
- Class Name: Object
- Fire transition: Unconditionally If message equals If expression is true (use msg for message)
- Action: (empty field)
- Guard: (empty field)

The left sidebar shows a project tree with folders for 'Functions', 'Presentation', and 'Simulation: Main'. The 'Problems' window at the bottom left lists several errors: 'Engine.log cannot be resolved'.



Hands on Model Use Ahead



Load Previous Built [& Provided] Model:
MinimalistNetworkABMModel

Sending Messages

Using a “Contact” Event to Spread Infection

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart diagram with two states: 'Susceptible' (yellow rounded rectangle) and 'Infected' (yellow rounded rectangle). A transition arrow points from 'Susceptible' to 'Infected'. A blue arrow points to this transition with the text 'Add this transition'. The transition is triggered by a 'Rate' event with a rate of 2. The action for this transition is `send("Infection", RANDOM_CONNECTED);`, which is highlighted with a red oval. The 'Properties' window for the transition is titled 'Contact - Transition' and shows the following configuration:

- Name: Contact
- Triggered by: Rate
- Rate: 2
- Action: `send("Infection", RANDOM_CONNECTED);`
- Guard: (empty)

The left sidebar shows a project tree with folders for 'Simulation: Main', 'MalariaV2', 'TestModel2*', and 'Spatial SIR with Waning Immunity'. The bottom status bar shows 'Problems'.

Transition Type: Message Triggered

Making Infection Depend on a Message

The screenshot displays the AnyLogic Advanced interface. The main workspace shows a statechart with two states: 'Susceptible' and 'Infected'. A transition between them is labeled 'InfectionS'. A blue arrow points to this transition, and a large blue text overlay reads: "Make sure you have selected the transition by clicking on it!". The 'Properties' window for the 'Infection - Transition' is open, with a red oval highlighting the 'Triggered by' dropdown set to 'Message' and the 'Other' radio button selected under 'Message type'. Other options include 'boolean', 'int', 'double', and 'String'. The 'Fire transition' section has 'Unconditionally' selected. The 'Action' and 'Guard' fields are empty.

Project: MalariaV2

- Main
 - Parameters
 - Functions
 - CountInfectiveHum...
 - CountInfectiveMosc...
 - PersistSimulationDa...
 - SelectRandomPerso...
 - SetParameters
 - SetSimulationOutpu...
 - getHumanPopulatio...
 - getMosquitoPopulat...
 - Events
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Mosquito
 - Person
 - Simulation: Main
- TestModel2*
 - Main
 - Person
 - Plain Variables
 - Statecharts
 - InfectionStatechart
 - Presentation
 - Simulation: Main
- Spatial SIR with Waning Immunity
 - Main
 - Person
 - Simulation: Main
- Network Modification of SIR AB
 - Main
 - Person

Setting "Person" so forwards Infection Message to Statechart

Message to Statechart

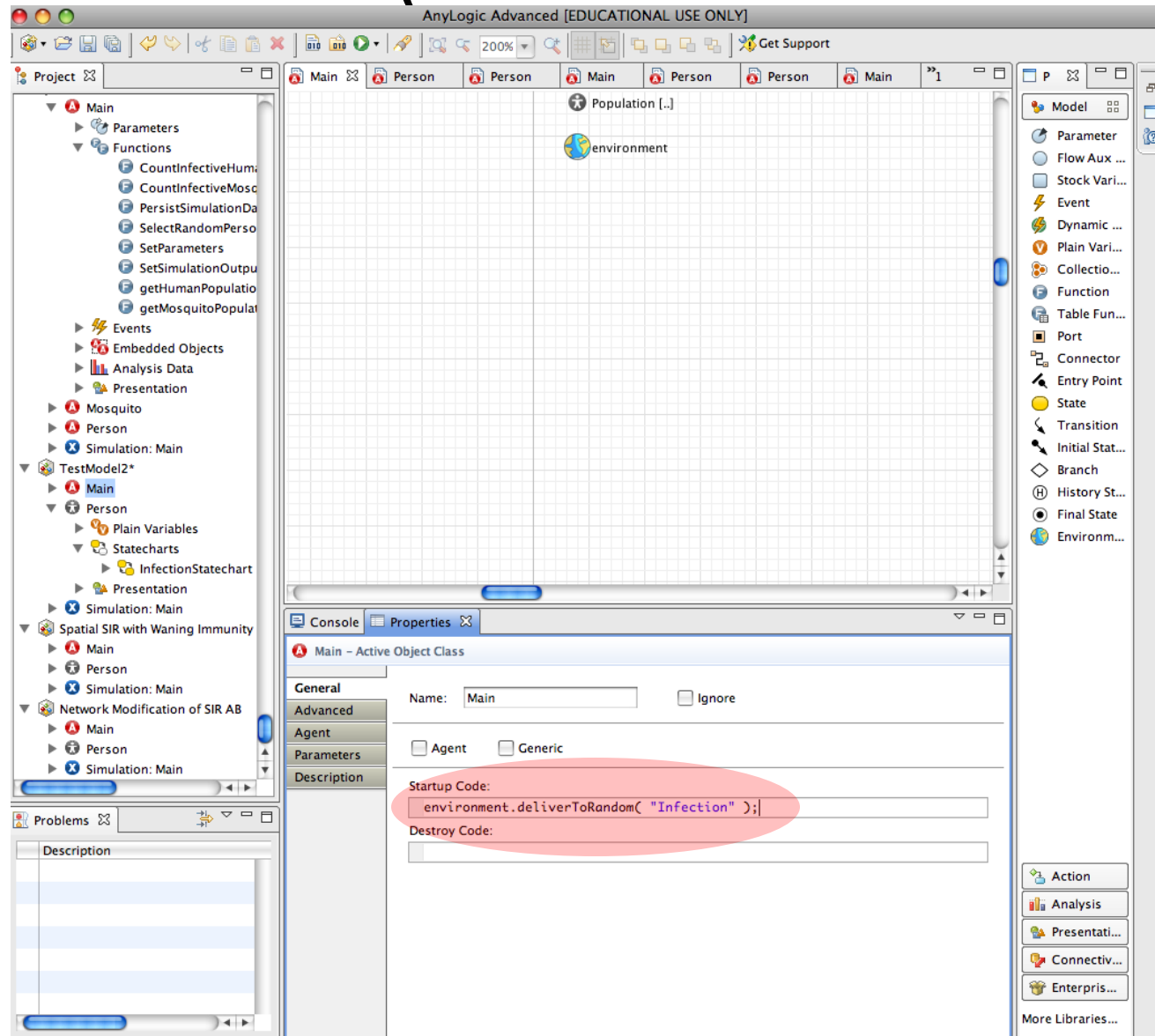
The screenshot displays the AnyLogic Advanced interface. The main workspace shows a statechart for the 'Person' agent. The statechart starts at an initial state (black dot) and transitions to a 'Susceptible' state (yellow rounded rectangle). From 'Susceptible', there is a transition to an 'Infected' state (yellow rounded rectangle). From 'Infected', there is a transition back to 'Susceptible'. A variable 'color' is shown as a plain variable (yellow circle with 'V').

Below the statechart, the 'Person - Active Object Class' configuration window is open. The 'Agent' tab is selected, and the 'On Message Received' event is configured with the following code:

```
InfectionStatechart.receiveMessage( msg );
```

Green text and arrows highlight the 'Agent' tab and the 'On Message Received' code block. A red oval highlights the code block.

Setting Startup Code So Initially Infects a Random Person (so start with 1 infective)



The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a diagram with a 'Population [..]' and an 'environment' object. The left sidebar contains a project tree with various models and objects. The bottom-right pane shows the 'Properties' window for the 'Main' object class. The 'Startup Code' field is highlighted in red and contains the following code:

```
environment.deliverToRandom( "Infection" );
```

The 'Destroy Code' field is currently empty.

Infection Percolation over the Network

