

Expressing Discrete Inter-Agent Dynamics via Messaging

Nathaniel Osgood

10-25-2009

Discrete Agent Coupling via Messages

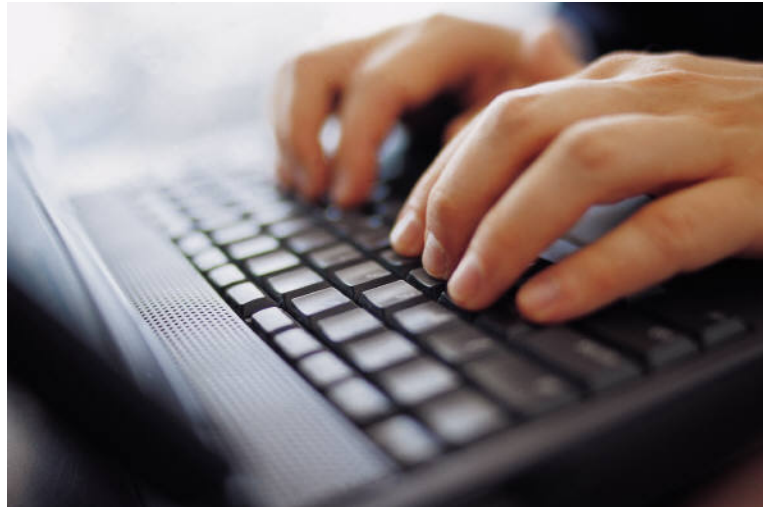
- Within AnyLogic, agents can be coupled by either discrete (instantaneous and individuated) or continuous (ongoing and gradual) coupling
- The preferred mechanism for discrete coupling is *messages* sent between agents
 - Many types of messages payloads are possible
 - An agent can send a given message 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)



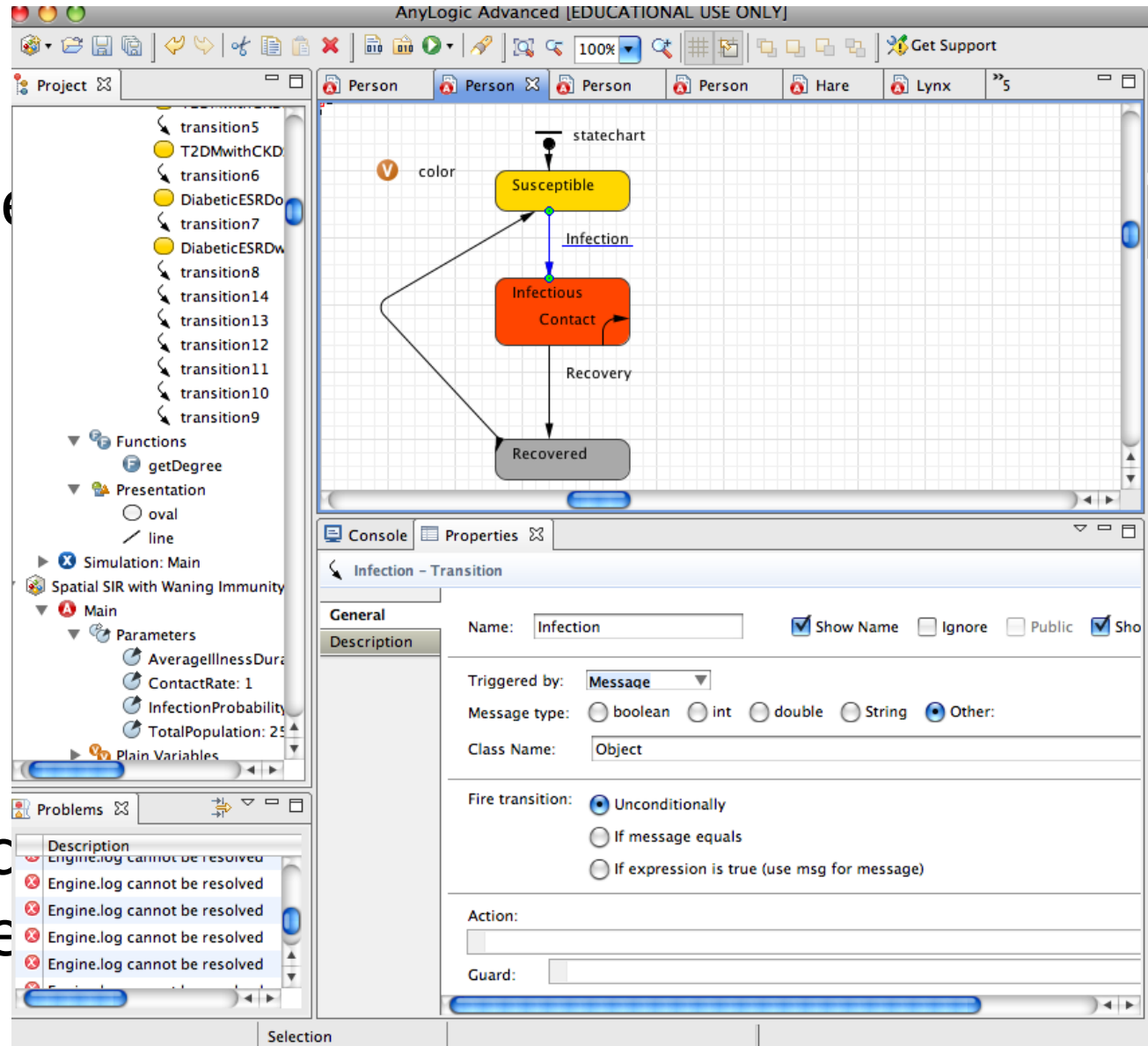
Hands on Model Use Ahead



Load model:Spatial SIR with Waning
Immunity.alp

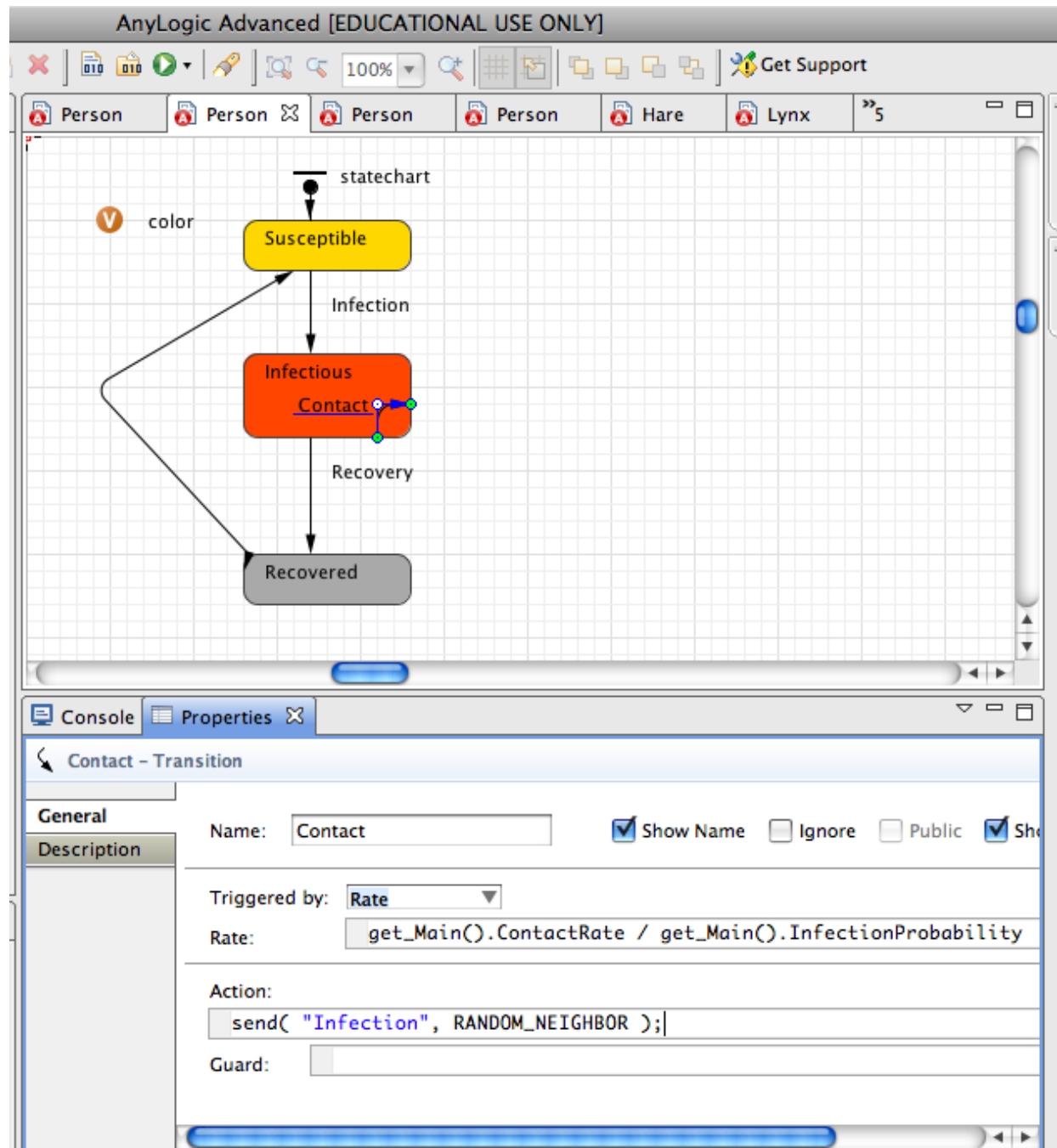
Receiving a Message

- 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 content



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)`

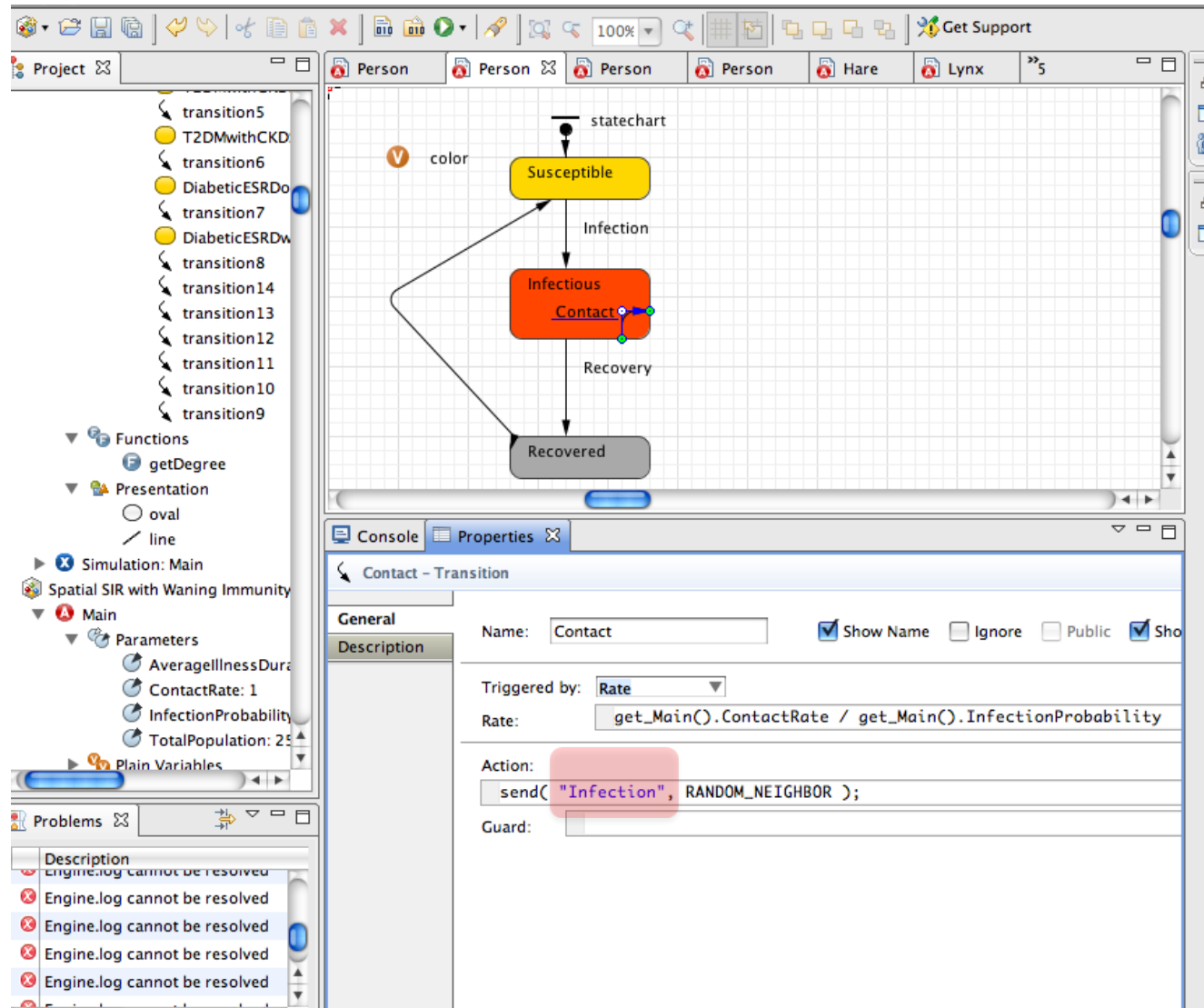
Synchronous vs. Asynchronous Delivery

- Messages may be sent in two ways
 - Via ***send***: Asynchronous
 - Delivery occurs sometime after call to send
 - Via ***deliver***
 - Synchronous
 - Risks infinite loops in delivery (if destination also calls deliver in the reverse direction)

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 multiple message types
 - Boolean/int/double/String/Other (can specify class type)

Sending a Message with a String Payload



Sending a Message with Object Payload

The screenshot displays the AnyLogic Advanced Modeling and Simulation environment. The main workspace shows a statechart for a **Person** object, illustrating the progression of an SIR model:

- Susceptible** (Yellow oval) transitions to **Infectious** (Red rounded rectangle) via the **Infection** transition.
- Infectious** transitions to **Recovered** (Grey rounded rectangle) via the **Recovery** transition.
- A self-loop transition labeled **Contact** is shown on the **Infectious** state.

The **Properties** panel on the right provides details for the **Contact - Transition**:

- Name:** Contact
- General:** ☒ Show Name, ☐ Ignore, ☐ Public, ☒ Show
- Description:**
- Triggered by:** Rate
- Rate:** `get_Main().ContactRate / get_Main().InfectionProbability`
- Action:** `send(this, RANDOM_NEIGHBOR);`
- Guard:**

The **Project** browser on the left shows the hierarchy: **Simulation: Main** > **Spatial SIR with Waning Immunity** > **Main** > **Person** > **Statecharts** > **statechart** > **statechart** > **Susceptible** > **Infection**.

The **Problems** panel at the bottom left lists several errors related to the `DataSet()` constructor and the `Engine.log` property.

Receiving a Message: Forwarding Messages on to the Statechart

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart for a 'Person' object. The statechart has four states: 'Susceptible' (yellow), 'Infectious' (orange), 'Recovered' (grey), and 'statechart' (black). Transitions are labeled 'Infection' (from Susceptible to Infectious), 'InfectionTransmission' (from Infectious to Susceptible), and 'Recovery' (from Infectious to Recovered). A 'getDegree' function is connected to the 'Recovered' state. The left sidebar shows the project hierarchy, including 'Person' and 'statechart'. The bottom right pane shows the 'Person - Active Object Class' properties, with the 'On Message Received' event set to 'statechart.receiveMessage(msg);'. A blue arrow points from the 'statechart' state to the 'On Message Received' event, and a red arrow points from the 'On Message Received' event to the 'statechart.receiveMessage(msg);' code.

Project

- environment
 - Embedded Objects
 - Analysis Data
 - Presentation
- Person
 - Plain Variables
 - color
 - Statecharts
 - statechart
 - Susceptible
 - Infection
 - Infectious
 - Recovery
 - Recovered
 - ImmunityWaning
 - InfectionTransmission
 - Functions
 - getDegree
 - Presentation
 - oval
 - line
- Simulation: Main
 - Presentation
- Network Modification of SIR AB
 - Main
 - Person
 - Simulation: Main

Person

statechart

color

Susceptible

Infectious

Recovered

getDegree

statechart

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 Stop:

Problems

Description	Location
-------------	----------

Selection

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

Receiving a Message

The screenshot displays the AnyLogic Advanced [EDUCATIONAL USE ONLY] interface. The main workspace shows a statechart 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 'statechart' label is at the top. The left sidebar contains a 'Project' tree with folders for 'Functions' (containing 'getDegree'), 'Presentation' (containing 'oval' and 'line'), and 'Simulation: Main' (containing 'Spatial SIR with Waning Immunity' and 'Main'). The 'Main' folder is expanded, showing 'Parameters' (with 'AverageIllnessDuration', 'ContactRate: 1', 'InfectionProbability', and 'TotalPopulation: 25') and 'Plain Variables'. The bottom-left pane shows a 'Problems' list with five entries: 'Engine.log cannot be resolved'. The bottom-right pane shows the 'Infection - Transition' configuration. The 'General' tab is active, showing 'Name: Infection', 'Show Name' checked, 'Ignore' unchecked, 'Public' unchecked, and 'Show' checked. The 'Description' tab is also visible. The 'Triggered by:' dropdown is set to 'Message'. The 'Message type:' options are 'boolean', 'int', 'double', 'String', and 'Other' (selected). The 'Class Name:' is 'Object'. The 'Fire transition:' options are 'Unconditionally' (selected), 'If message equals', and 'If expression is true (use msg for message)'. The 'Action:' and 'Guard:' fields are empty.

AnyLogic Advanced [EDUCATIONAL USE ONLY]

Project

- transition5
- transition6
- transition7
- transition8
- transition14
- transition13
- transition12
- transition11
- transition10
- transition9

Functions

- getDegree

Presentation

- oval
- line

Simulation: Main

- Spatial SIR with Waning Immunity
- Main
 - Parameters
 - AverageIllnessDuration
 - ContactRate: 1
 - InfectionProbability
 - TotalPopulation: 25
 - Plain Variables

Problems

- Description
- Engine.log cannot be resolved
- Engine.log cannot be resolved
- Engine.log cannot be resolved
- Engine.log cannot be resolved
- Engine.log cannot be resolved

statechart

Susceptible

Infectious Contact

Recovered

Infection

Recovery

console

Properties

Infection - Transition

General

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:

Guard: