Multi-agent Simulation Programming Environment with Error-pruning Support based on Typical Pattern Description

Masanori Akiyoshi, Kota Itakura, Norihisa Komoda

Osaka University, JAPAN
Contents

1. Research background
2. Outline of our research
3. Graph-based model editor
   Event description
   Event flow and event group
   Source code generation
4. Model debugging
   Error-inducing factors tree
   Trimming the factors
5. Experiment
6. Conclusion
What is Multi-agent simulation (MAS)?

- One of simulation methods for complex systems such as social system, economy system
- Behavior comes from autonomous interaction among elements

Design flow of MAS

Rough sketch by a designer

Model description

Program source code

Execution of simulation

It takes a lot of time to execute MAS.

Programming skill is necessary

Consistent and understandable notation is not developed so far.
Outline of our research

Research purpose

✓ Ease-of-use method for describing a target model
✓ Reduce programming time

MAS development environment

Problem to be tackled

✓ How to reflect rough sketch of a user’s MAS model based on the agent model description; Attributes of an agent, Behavior of an agent, Mutual effects among agents
✓ How to debug the model description errors without source code debug
Graph-based model editor

Target Model (shopping mall)

**Distinctive feature of the notation**

- Event description is based on typical action patterns of agents using “interrogative (4W1H) and verbs”.
- Mutual interactions are described with “event flow diagram of agent events”.

**MAS Model Editor**

- **Design of Model Elements**
  - shop
  - customer
  - sale
  - buy
  - permission
  - go to shop
  - leave shop
  - select shop

- **Design of Event Flow**
  - <customer>
    - circulate
    - select shop
    - go to shop
    - permission(wait)
    - buy
    - leave

- **Design of Event Contents**
  - Event <buy>
    - When: Always
    - Who: shop
    - What: my. sale
    - Verb: Change
    - How: Add(1)
Event description – typical action pattern

Without programming, it enables to design events along with narrative scenario of a user.
Event description – verbs and interrogates

verb
Create
   agent creation
Delete
   agent deletion
Make Decision
   agent behavior firing/not-firing
Select One
   interaction agent selection
Get
   get attributes of an target agent
Set
   set attributes of an target agent
Change
   set own attributes

When
Always
Poisson distribution (λ)
   interval based on poisson distribution
Compare(conditional description)
After acting
After not acting
Event description – interrogates

**How**

- **Probability(P)**
  - act “verb” under P

- **Probability(((P1, ID1) (P2, ID2) ⋯)**
  - act “Idxx action” under Pxx

- **Normal Add(value)**
  - act “verb” after adding value

- **Int Random Add(minValue, maxValue)**
  - act “verb” after a random value between minValue and maxValue

- **Substitute(value)**
  - act “verb” after substituting a value

- **Random Substitute(minValue, maxValue)**
  - act “verb” after substituting random value between minValue and maxValue

- **Assign To(variableName)**
  - assign acted results to variableName

- **Min(variableName)**
  - act “verb” with minimum values of variableName

- **Max(variableName)**
  - act “verb” with maximum values of variableName

- **Compare(conditional description)**
  - act “verb” if conditional description is true
“Event group” is designed as events are executed in the same simulation step. For instance, “select shop” and “go shop” are the same group.
Source code generation

Design of event flow

Declaration of variables

Description of transition

Definition of functions

Utilize the programming form

design of event contents
Model debugging

What is model debug?

- When detecting **unintended behavior** on execution, it is necessary to find causes in reverse manner and modify the model.

Examples of description error

- Mistake in the description of an event
- Loss of transition in an event flow
- Mistake of initial value setting

Propagation of effects  
Finding of causes  
Unintended behavior

Why such findings are difficult?

- MAS is not intrinsically traceable.
- MAS components are intertwined.
Model debugger with error-pruning

Candidates of errors are induced from unintended behavior.

Outline of Model Debugger

- Report data format
  Typical report patterns for unintended behavior

- Construction of error candidates
  Error-inducing factors tree and trimming the factors based on execution log
Error-inducing factors tree

**Typical report patterns**

- Reports are done by report vocabulary related to event, agent and attributed of agents.

**Generation of error-inducing factors tree**

- Iterative matching with reports and error-inducing patterns

**Report of unintended behavior**

- Not firing <buy> event
  - Not designing transition to <buy> event
  - Not to shift to <buy> event
  - Not to shift to <buy> event
    - Incorrect condition in transition
    - Invalid value of <customers>
    - Not firing <permission> event

**Error patterns for target model**

- Not firing <buy> event
- Not designing transition to <buy> event
- Not to shift to <buy> event

**Match the corresponding pattern**

**Error-inducing pattern DB**
Trimming the factors

Tree elements are trimmed by using execution log such as the number of event firing, the number of action firing, the attributes values and so on.
Experiment

Target model and scenario

- Customers arrive at the west gate or the south gate by some probability.
- Customers move in the shopping mall freely.
- Shop B passes coupons at the west gate and customers receive them at a constant rate.
- Customers go into the shop if distance between a customer and a shop is within a constant value.

.......

Model generation

<table>
<thead>
<tr>
<th></th>
<th>Event description; artisoc(programming) and our description method (artisoc ⇒ our method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Event description; artisoc(programming) and our description method (our method ⇒ artisoc)</td>
</tr>
</tbody>
</table>

Model debugging

3 errors are embedded in advance.
Experimental results – model generation

In case of all the description of the target model (examinee “C”)

<table>
<thead>
<tr>
<th>Agents</th>
<th>2 (shop, customer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of attributes</td>
<td>7 (shop), 9 (customer)</td>
</tr>
<tr>
<td>Num. of events</td>
<td>8</td>
</tr>
<tr>
<td>Num. of event flows</td>
<td>3</td>
</tr>
<tr>
<td>Description time</td>
<td>240 minutes</td>
</tr>
</tbody>
</table>

It is effective to reduce the time for description.

It is effective even if a user does not have any knowledge on programming.

<table>
<thead>
<tr>
<th>Artisoc</th>
<th>Our description method</th>
</tr>
</thead>
<tbody>
<tr>
<td>examinee</td>
<td>Time (min)</td>
</tr>
<tr>
<td>A</td>
<td>210</td>
</tr>
<tr>
<td>B</td>
<td>210</td>
</tr>
</tbody>
</table>

* Examinee “B” cannot complete the programming codes.
Experimental results – model debug

<table>
<thead>
<tr>
<th>Contents of Error</th>
<th>Unintended behavior</th>
<th>With log</th>
<th>Without log</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Reversed an inequality sign in condition</td>
<td>Event not firing</td>
<td>17</td>
<td>7</td>
</tr>
<tr>
<td>2 Loss of transition in an event flow</td>
<td>Agent not moving</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>3 Loss of an event description</td>
<td>Attribute not changed</td>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>

It is effective to reduce the time for debugging

- Using log information helps to narrow the list down to about 60%.
- 4 minutes to find errors assuming 30 seconds to check if a candidate is error, on the other hand debugging without our model debugger takes 40 minutes to find errors.
Conclusion


- A model debugger that generated error-inducing factors with typical report patterns of unintended behavior and execution log

- Through the experiments, our programming environment for MAS is effective to reduce the time compared to writing the source codes.
Screen shot – Model elements design
Screen shot – Typical report patterns