Interaction Selection Ambiguities in Multi-Agent Systems

Yoann Kubera
yoann.kubera@lifl.fr

Philippe Mathieu
philippe.mathieu@lifl.fr

Sébastien Picault
sebastien.picault@lifl.fr

LIFL UMR USTL-CNRS 8022
Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
Cité Scientifique - 59655 Villeneuve d'Ascq Cedex – FRANCE

Abstract

To ensure multi-agent based simulation models reproducibility, particular attention must be payed on its possible implementation ambiguities. This concerns every aspect of simulation's architecture, including how the agent selects the actions it performs, and on which agents it is performed. On this point, classical agent-centered design methodologies leave room to implicit design choices.We show in this paper how an interaction-centered design methodology provides guidelines to elicit these choices, and to remove possible implementation ambiguities underlying agent design. To illustrate this issue, we study which ambiguities underly even simple models and how our interaction-based methodology makes them appear.

1 Introduction

Each step of the simulation design process involves choices – both explicit or implicit – regarding ambiguous parts of the model. Those choices have a more or less dramatic influence on the execution and outcomes of the simulation. The simulation biases they introduce must be studied. Otherwise, the ambiguity of the models leads to simulations that do not behave as it was initially expected and thus produce also unexploitable results.

The spectrum of implicit choices is wide, and concerns very different parts of simulation's architecture. To study these choices separately, a decomposition of a MABS in three different functional units was proposed in [3] that manage either the knowledge of agents, their action selection process, or when and under what conditions agents act (see section 2). That paper studied the impact of implicit choices concerning this last unit called *"Agents and Environments Activation Unit"*.

We consider in this paper a second unit of this decomposition called the *"Interaction Selection Unit"*. This unit describes the process that leads an agent to select from his knowledge the interaction if performs.

Interactions between agents – *i.e.* actions involving simultaneously two or more agents – are the source of simulations emergent properties. Thus, they have a major role in Multi-Agent Based Simulations (MABS). But, because current MABS design methodologies focus only on the behavior of independent agent, many design choices concerning interactions are not explicit. In particular, agents only define how they select the action they perform [2, 1], but do not provide guidelines on how target agents are selected (see next section). In this paper we use the interaction-oriented methodology IODA [4] to emphasize these choices.

We uphold that these selection principles have to be made explicit in the earliest steps of every simulation design, to determine precisely what interpretation of the model is made. This provides design guidelines that clear many implementation ambiguities, and makes sure that the model is implemented as it was thought.

2 Multi-agent Simulations

Even if the application domains of multi-agent simulations are heterogeneous, they can be split up into different and weakly dependent functional units [7, 3] **that underlie any kind of simulation**. Especially, it elicits the notion of interaction that often exists in agents code, but does not appear explicitly.

We consider a particular decomposition of a simulation in three main functional units, called *Agents and Environments Activation Unit* (*AEAU*), *Interactions Definition Unit* (*IDU*) and *Interaction Selection Unit* (*ISU*) (see [3] for more details). **Because the design of simulations implies crucial choices about those three units, we claim that it is important to make this separation clear, even in reactive simulations, in order to make modeling choices explicit.**

The significance of the *IDU* specification and generic representation has been addressed in [4], and the impact of implementation choices of the *AEAU* was dealt with in [3]. Thus, we focus in this paper on the latest unit, the *ISU*.

In summary, this unit defines how an agent selects the interaction it performs among its perceived affordances [6].

Interactions

The definition of interactions, and how they are integrated in the knowledge of agents, are based on IODA concepts [4]. Please note that IODA provides advanced methodological and software engineering tools to design interactions in MABS. Since we do not need all refinements it provides, we use a simplified version of [4] definitions.

To make the difference between the abstract concept of agent (for instance Wolves), and agent instances (a particular Wolf), we use the notion of **agent families** as abstract concept of agent and **agent** as an agent instance.

Definition 1 *An **agent family** (or agents equivalence class or agent class) is an abstract set of agent instances, which share all or part of their properties and behavior.*
From this point on, if $\mathcal{F}$ is an agent family and x an agent, $x \prec \mathcal{F}$ means x is an instance of the $\mathcal{F}$ agent family.

Definition 2 *An **interaction** is a structured set of actions involving simultaneously a fixed number of agents instances that can occur only if some activation conditions are met.*
It is represented as a couple $(conditions,\ actions)$, where $condition$ is a boolean function and $action$ is a procedure. Both have agent instances as parameters.

Agents involved in an interaction do not play the same role. We make a difference between **Source** agents that may perform the interaction (in general the one selected by the *AEAU*) and **Target** agents that may undergo it.

This definition is more general than the coordination language's one where an interaction is restricted to a structured set of messages between agents : we consider here all kinds of actions, including messages exchanges.

Definition 3 *If $\mathcal{S}$ and $\mathcal{T}$ are agent families, $a_{\mathcal{S}/\mathcal{T}}$ is the set of all interactions an instance of the $\mathcal{S}$ agent family is able to perform with an instance of the $\mathcal{T}$ agent family as target. By extension, if $x \prec \mathcal{S}$ and $y \prec \mathcal{T}$, then $a_{x/y} \equiv a_{\mathcal{S}/\mathcal{T}}$.*

Adding to that, we use the notion of realizable interaction to determine if agents can participate in an interaction.

Definition 4 *Let I be an interaction, and x, y two agents. The tuple (I, x, y) is **realizable** (**r(I, x, y)**) if and only if : $r(I, x, y) = I \in a_{x/y}$ and $I.conditions(x, y) = TRUE$.*

Thus, at a time t, x agent's perceived affordances $\mathbb{R}_t(x)$ are the set of all realizable tuples that x may perform.

Definition 5 *Let $\mathbb{A}_t$ be the set of all agents present in the simulation at a time t, and $x \in \mathbb{A}_t$.*
Then, the list of all realizable tuples that x may perform is : $\mathbb{R}_t(x) = \cup_{y \in \mathbb{A}_t} \cup_{I \in a_{x/y}} \{(I, x, y) | r(I, x, y)\}$

3 Experimental Frame

In many simulation platforms, the declaration of interactions, that belongs to the *IDU*, is mixed with the implementation of agent behavior, that belongs to the *ISU*. As a consequence, an agent uses implicitly particular algorithms to select among all realizable tuples it can perform – *i.e.* the perceived affordance of the agent – the one it will perform. These algorithms are more-or-less suited to represent the model, and may introduce biases in simulation outcomes.

The aim of this paper is to stress out the benefits brought by an algorithmic study of the Interaction Selection Unit (*ISU*), and how it can be used as model design guidelines. This point is illustrated through the experiment below, where the focus is on the results of a single agent behavior, instead of the whole simulation results. Experiment's main issue is to define an *ISU* corresponding to the specifications of the problem – *i.e.* how a source agent selects the interaction it performs, and with which target agent it is performed.

4 Multiple Interpretations of Probabilities

This experiment aims at representing the behavior of an agent that selects an interaction with a particular probability. It illustrates a first advantage of the early study of the *ISU* : the detection of different possible interpretation of a single model. Moreover, it identifies two different patterns to select a perceived affordance of an agent. Even if the solution is provided for the ecosystem example, it remains valid for the generic problem in bold fonts.

Experiment Definition The experiment describes a simple ecosystem where Goat agents may :

EAT an agent, to reduce the source agent tiredness. This removes the target agent from the environment;

REPRODUCE with an agent if both source and target are not tired. This creates a copy of the source agent on a neighboring place, and raises source and target agent tiredness.

A Goat agent REPRODUCES with other not tired Goats with a $p_r = 60\%$ probability.

Experimental Frame We suppose that a Goat labeled g_0 is not tired, and is in a situation such that it cannot EAT Grass. Moreover, we suppose that there are m other Goat agents in the environment, that we label $(g_i)_{i \in [1,m]}$, and that $n < m$ among them are not tired. We consider the two different implementations of g_0's *ISU* presented in Fig. 1.

Let i **=** 1**.**
Let $select$ **=** $\emptyset$**.**
While $i \leq m$ **Do :**
| **If** $\neg tired(g_i)$ **and** $random([0,100[) < 60$ **Then :**
| | **Set** $select$ **=** $select \cup \{(\text{REPRODUCE}, g_0, g_i)\}$**.**
| **End If.**
| **Set** i **=** $i + 1$**.**
End While.
If $select = \emptyset$ **Then :**
| **Return** $nill$**.**
Else :
| **Return** $random(select)$**.**
End If.

(a) "Equitable Tuple" selection policy

Let r **=** $random([0,100[)$**.**
If $r < 60$ **Then :**
| **Let** $select$ **=** $\emptyset$**.**
| **For All** g **in** $(g_i)_{i\in[1,m]}$ **Do :**
| | **If** $\neg tired(g)$ **Then :**
| | | **Set** $select$ **=** $select \cup \{\text{REPRODUCE}, g_0, g\}$**.**
| | **End If.**
| **End For.**
| **If** $select = \emptyset$ **Then :**
| | **Return** $nill$**.**
| **Else :**
| | **Return** $random(select)$**.**
| **End If.**
Else :
| **Return** $nill$**.**
End If.

(b) "Weighted Interaction-driven" selection policy

Figure 1. Implementations of the ISU used by a g_0 Goat agent in the studied experiment.

Selection Policies Interpretation : In the first algorithm (a in Fig. 1) 60% is the probability that g_0 REPRODUCES with an other Goat agent. Thus, the probability that g_0 does not reproduce depends on the number n of not tired Goats agents, and is equal to $(1 - p_r)^n$.

The second one (b in Fig. 1) considers 60% as the probability that g_0 performs REPRODUCE independently from the number of not tired Goats agents. Thus, the probability that g_0 does not reproduce does not depend on the number n of not tired Goats agents, and is equal to $(1 - p_r)$.

Both reflect two different interpretations of the model. In the first one, the REPRODUCTION probability is tried independently by every possible target t. A tuple $(\text{REPRODUCE}, g_o, t)$ is listed in the `select` set only if this probability is met. Thus, the probability test belongs to the condition of REPRODUCE interaction, and the *ISU* just selects a realizable tuple at random. We name this first interaction selection policy "Equitable Tuple" selection policy.

In the second one, the REPRODUCTION probability is tried once. If this probability is met, then a target t for the REPRODUCE interaction, such that $r(I, g_0, t)$, is selected at random. Thus, in that process, a particular interval $[0, 60[\subseteq [0, 100[$ is associated with the REPRODUCE interaction. The length of this interval depends on the probability to trigger REPRODUCE (the length of this interval is called the weight of REPRODUCE). This *ISU* gets a number $r \in [0, 100[$ at random. If r belongs to the interval of REPRODUCE, then a realizable tuple of the REPRODUCE interaction is selected at random. We name this second interaction selection policy "Weighted Interaction-driven" selection policy.

Interaction-Oriented Design : If this problem was to be represented in an interaction-oriented methodology, then, in the first interpretation (a in Fig. 1), REPRODUCE condition would be "$\neg tired(Target) \wedge random([0,100[) < 60$", and its selection policy would be an *Equitable Tuple* one. If this problem was to be represented in an interaction-oriented methodology, then, in the first interpretation (a in Fig. 1), REPRODUCE condition would be "$\neg tired(Target) \wedge random([0,100[) < 60$", and its selection policy would be an *Equitable Tuple* one.

On the opposite, in the second interpretation (b in Fig. 1), the condition of REPRODUCE would be "$\neg tired(Target)$", the interaction selection policy would be the *Weighted Interaction-driven* one, and the weight of REPRODUCE in this policy would be p_r.

Thus, the specification of the *ISU* in an interaction-oriented methodology clears such interpretation ambiguity.

5 Summary of Experimental Results

The experiment above is part of an experiment set we led. Their study ended up with two different observations.

First they show that two different implementations of a same model are the fruit either of different interpretations of the model, or of wrong implementations of that model. To avoid implementation biases and ambiguities, the modeler has to elicit the choices concerning the *ISU*. This requires to have *ISU* design guidelines, and to identify the different possible choices that occur in that process.

Secondly, their interpretation and confrontation elicited the overall structure that the *ISU* follows. This section summarizes its main elements, and how to express them in the interaction-oriented methodology IODA.

Overall structure of the *ISU* In reactive simulation, agents try in general to perform actions sequentially until a realizable one is found. We propose to use a similar overall principle in the *ISU* : every possible interaction I between a source $\mathcal{S}$ and a target agent family $\mathcal{T}$ is assigned a priority $p(I, \mathcal{S}, \mathcal{T})$, just as [5] did for normal actions.

The execution of $x \prec \mathcal{S}$ agent's *ISU* consists in :

1. gathering the priorities of all the interactions x can perform in an ordered set $\mathbb{P}(\mathcal{S})$;
2. selecting the next priority p of $\mathbb{P}(\mathcal{S})$. If there is not such a priority, then the agent does nothing;
3. getting the set $\mathbb{R}_t^p(x) \subseteq \mathbb{R}_t(x)$ containing all realizable tuples of p priority. $\mathbb{R}_t^p(x) = \{(I, x, y) | \exists \mathcal{T} | y \prec \mathcal{T} \; and \; p(I, \mathcal{S}, \mathcal{T}) = p\}$;
4. applying a particular interaction selection policy, that depends on p, to select a tuple (I, x, y) from $\mathbb{R}_t^p(x)$.
5. returning to step 2 if no tuple is selected;
6. executing the selected tuple (*i.e.* $I.actions(x, y)$).

Interaction Selection Policies Thanks to the interaction-oriented study of experiments, four interaction selection policies, used to select a tuple from a set of realizable tuples $\mathbb{R}_t^p(x)$, were identified :

An **"Equitable Tuple"** selection policy that selects one tuple of $\mathbb{R}_t^p(x)$ at random. This is the most implicitly used policy in reactive simulations.

An **"Equitable Interaction"** selection policy that first selects an interaction present in tuples of $\mathbb{R}_t^p(x)$ at random, and then selects one tuple of $\mathbb{R}_t^p(x)$ that contains this interaction at random.

A **"Weighted Interaction-driven"** selection policy that gives to every interaction I present in tuples of $\mathbb{R}_t^p(x)$ a weight $p(I)$. It associates to every interaction I pairwise non-intersecting subsets $\mathbb{Y}(I) \subseteq [0, 100[$ of length $p(I)$. Then, it selects $r \in [0, 100[$ at random, and selects the interaction I such that $r \in \mathbb{Y}(I)$. Finally, it selects one tuple of $\mathbb{R}_t^p(x)$ that contains this interaction at random. To use this policy, the user has to define the weights of interactions.

A **"Preferred Tuple"** selection policy, that gives to every tuple $t \in \mathbb{R}_t^p(x)$ a utility value $v(t)$. It selects the tuple that has the maximal value among tuples of $\mathbb{R}_t^p(x)$. If more than one tuple has the maximal value, then the selection is made at random between these last tuples. To use this policy, the user has to define how the value of a tuple is computed. This last one is well known for its applications in reactive simulations such that pheromone following, and for its intensive use in cognitive agents.

Design Guidelines of the *ISU* To design an *ISU* containing the fewest ambiguities, the modeler has to provide :

- priorities to every interaction an agent may perform;
- an interaction selection policy for each couple (source agent family, priority);
- if necessary, their weight or value computing function.

This is possible only if interactions are at center of simulation, with a modeling methodology like IODA [4][1].

6 Conclusion

The biases that may result from implementation choices must be identified and quantified. Otherwise simulations remain ambiguous and are not reproducible nor viable [3].

This paper shows that the reproducibility of a simulation is not possible without specifying a domain-independent functional unit underlying any simulation : the interaction selection unit. This unit defines how an agent selects an interaction among its perceived affordances.

Experiments concerning this unit showed that, even in simple cases, a single choice may greatly influence simulation outcomes. For instance, it may change the actual interpretation of the probabilities used to trigger behaviors.

Those problems can be pointed out through the identification of the interaction selection concepts used in the model. This identification requires to elicit the notion of interaction underlying any kind of simulation, just like in the IODA methodology and JEDI framework [4], and to separate the domain-dependent interaction declaration from the domain-independent selection of a performed interaction.

This paper identifies an overall description of this unit for reactive agents, and the four main interaction selection policies that it uses. It provides design guidelines for that unit, that elicit the choices that were otherwise implicit.

Without the specification of this point, different developers will likely obtain different outcomes for the same model.

References

[1] J.-P. Briot and T. Meurisse. An experience in using components to construct and compose agent behaviors for agent-based simulation. In *Proceedings of IMSM'07*, 2007.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *iEEE journal of robotics and automation*, 2(1), 1986.

[3] Y. Kubera, P. Mathieu, and S. Picault. Biases in multi-agent based simulations : An experimental study. In *Proceedings of ESAW 08*, St Etienne, France, 2008.

[4] Y. Kubera, P. Mathieu, and S. Picault. Interaction-oriented agent simulations : From theory to implementation. In *Proceedings of ECAI 08*, Patras Greece, 2008.

[5] N. J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1, 1994.

[6] D. A. Norman. *The Psychology of Everyday Things*. Basic Books, 1988.

[7] D. Weyns, H. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multiagent systems: State-of-the-art and research challenges. In *Environments for Multiagent Systems*, New York, NY, USA, 2004.

[1] See `http://www.lifl.fr/SMAC/projects/ioda`