How to avoid Biases in reactive simulations

Yoann Kubera, Philippe Mathieu and Sébastien Picault

Abstract In order to ensure simulations reproducibility, particular attention must be payed to the specification of its model. This requires adequate design methodologies, that enlightens modelers on possible implementation ambiguities – and biases[1] – their model might have. Yet, because of not adapted knowledge representation, current reactive simulation design methodologies lack specifications concerning interaction selection, especially in stochastic behaviors. Thanks to the interaction-oriented methodology IODA – which knowledge representation is fit to handle such problems – this paper provides simple guidelines to describe interaction selection. These guidelines use a subsumption like-structure, and focus the design of interaction selection on two points : how the selection takes place – for instance first select the interaction, and then select the partner of the interaction, or first a partner and then an interaction – and the nature of each selection – for instance at random, or with a utility function. This provides a valuable communication support between modelers and computer scientists, that makes the interpretation of the model and its implementation clearer, and the identification of ambiguities and biases easier.

1 Introduction

Any Multi-Agent-Based Simulation (MABS) – and more generally any kind of simulation – is implemented according to a model defined by domain specialists. These specialists are not always fully aware of implementation requirements. As a result, computer scientists have to make implementation choices, that may lead to biased results. Even worse, because of programming habits and too permissive methodologies and frameworks, these choices might be implicit. For instance in Epstein and

Yoann Kubera, Philippe Mathieu and Sébastien Picault
Laboratoire d'informatique Fondamentale de Lille, University of Lille, France e-mail: forename.surname@lifl.fr

1

Axtell ecosystem simulation [5], a bias occurred in results because the interactions in which an agent might participate at the same time were not specified.

To ensure simulation reproduction – *i.e.* obtain similar results with implementations of the model made by different persons – and to hedge against ambiguities and biases, domain specialists have to consider the most exhaustive set of questions about what they want or expect of their model. Indeed, their answers elicit implementation choices, and remove ambiguities that may lead to different implementations. Moreover, it makes sure that choices – including the choice to not answer some questions – are made willingly, and aware of the biases they may introduce.

In this paper, "bias" means "erroneous/distorted simulation outcomes". Thus, biases are the result of either defective means such as faulty random number generators, or of wrong implementation choices. This paper focuses on this second point.

Our goal is to provide a generic and domain independent simulation design methodology called *IODA* and framework called *JEDI* [8]. This paper participates in that effort by defining a particular question – and its answer – that all simulation methodologies should consider to prevent implementation biases : *"how agents select the action or interaction they perform among their perceived affordances*[2] *?"* [13]. Since this aspect is well specified for cognitive agents, the focus of this paper is reactive agents. Nevertheless, our proposition remains valid for cognitive ones.

In order to clarify precisely this point, at least two properties are required :

- *knowledge of agents* – what they are able to do – *has to be defined separately from action/interaction selection* – what an agent chooses to do. This separation has to be made even for reactive simulations;
- *interaction* – a notion underlying any simulation – *has to appear explicitly in the methodology as well as in the implementation, as a software entity*.

Yet, simulation methodologies do not meet the requirements of the last point (see section 3), and thus remain ambiguous on how action/interaction selection is handled in reactive simulations. Indeed, agents define only how they select the action or interaction they perform [3, 15], but do not provide guidelines on how target agents are selected (see section 2), even though these processes are deeply bound together.

This paper aims at filling this gap by first specifying an architecture that underlies any kind of multi-agent-based simulation, and that is fit to enlighten modelers on the problem mentioned above (see section 2). Then, a solution of this problem is presented in section 4. In this solution, the modeler elicits the action/interaction selection process of agents in two parts. First he has to specify how selection takes place among three recurrent patterns met in simulations – *first interaction then target* selection, or *first target then interaction* selection or *tuple* selection – and then the nature of every selection among three ones – either *random*, *by preference* or *weighted*. We uphold that such specifications provide a valuable communication support between modelers and computer scientists, that makes the interpretation of the model and its implementation clearer, and that makes the identification of model ambiguities and possible biases easier. We illustrate this solution on a modeling example (see section 5), that shows the importance of such a specification.

[2] What an agent knows it can perform in a given context.

2 Separation in Functional Units

Even if the application domains of multi-agent simulations are heterogeneous, they can be split into different and weakly dependent functional units [4, 16]. We consider a particular functional decomposition that underlies any kind of simulation. This decomposition is done in three main units (see Fig. 1), called ACTIVATION UNIT, DEFINITION UNIT and SELECTION UNIT (see [7]).

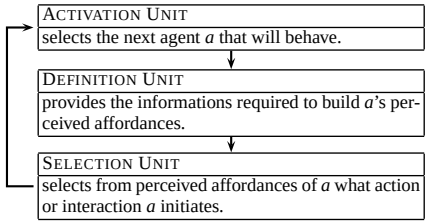

Fig. 1 The three main functional units of a multi-agent simulation described in [7].

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 DEFINITION UNIT specification and generic representation has been addressed in [8], and its relevance to elicit model ambiguities and possible biases is demonstrated in [7] and in this paper. The impact of implementation choices of the ACTIVATION UNIT was dealt with in [7], and studies possible answers to the questions *"when agents trigger their behavior ?"* and *"in which interactions an agent may participate simultaneously ?"*. Thus, we focus in this paper on the latest unit, the SELECTION UNIT.

3 Related Works

The space of implementation choices is really wide. To guide modelers in the hard task of eliciting modeling and implementation choices, many agent-based simulation design methodologies exist and claim to handle this issue.

Some of them are all purpose design methodologies – like VOLCANO [14]. On the opposite, many are specific to particular subsets of simulations – like DESIRE [15] that designs reasoning agents, or ADELFE [1] that designs adaptative agents. Because they are developed for particular use, they target more specific problems, and thus provide a more exhaustive specification of implementation choices for it.

Reactive simulation design methodologies have a particular status among these last. Indeed, even if they claim to be methodologies, many just consist in writing the simulation in the agent language or architecture they provide. Thus, unless the struc-

ture of the architecture forces to make choices, there is no guidelines to build behaviors. Some methodologies and frameworks make the separation between knowledge of agents and action/interaction selection, and provide guidelines to build reactive agents behavior. This is the case of component based frameworks like MALEVA [2], or of hybrid frameworks like InteRRap [11] and PRS-based ones [6].

Nevertheless this separation is a necessary but not a sufficient condition to avoid biases coming from action/interaction selection. Indeed, agents action/interaction selection is the art of selecting the next action or *interaction* it will initiate. The underlying problem is that agents have to consider which interactions they will initiate *and with which other agent it will be performed* (*i.e.* the target agent). Sadly, target agent choice process remains unspecified in such methodologies.

To elicit such issues, we uphold that separation must be made between knowledge declaration, perceived affordances listing and action/interaction selection process. Thanks to that, different patterns of action/interaction selection were identified. The modeler has to take into account these last to ensure that his model will be understood and implemented as it was firstly thought.

4 Unit Specification Proposal

To specify clearly the SELECTION UNIT, we center the action/interaction selection process of agents on the notion of interaction and perceived affordances – *i.e.* the set of all actions and interactions an agent might perform in a particular context.

Perceived affordances construction requires a specific representation of actions and interactions. We use the one of the *IODA* methodology [8], that reifies interactions and perceived affordances even at implementation. *IODA* provides advanced methodological tools to design interactions in MABS. Since we do not need all refinements it provides, we use a simplified version of [8] definitions.

4.1 Knowledge and Affordances Representation

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. Thus, the word **agent** refers to an agent instance.

Definition 1. An **agent family** is an abstract set of agent instances, which share all or part of their attributes and behavior.

Definition 2. An **interaction** is a structured set of actions involving simultaneously a fixed number of agents instances that can occur only if some conditions are met.

An interaction 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 that are involved in an interaction play different roles. We make

a difference between **Source** agents that may initiate the interaction (in general the one selected by the ACTIVATION UNIT) and **Target** agents that may undergo it.

Definition 3. Let $\mathscr{S} \in \mathbb{F}$ and $\mathscr{T} \in \mathbb{F}$ be agent families.
We note $a_{\mathscr{S}/\mathscr{T}}$ the **set of all interactions** that an instance of the $\mathscr{S}$ agent family is able to initiate with an instance of the $\mathscr{T}$ agent family as a target.

Thanks to these definitions, we can specify the knowledge of an agent family $\mathscr{S} \in \mathbb{F}$ as the set $\bigcup_{\mathscr{T} \in \mathbb{F}} a_{\mathscr{S}/\mathscr{T}}$, which contains every interactions it is able to initiate as source with any agent family as target.

To unify knowledge, actions are considered as interactions that occur with no target. We do not add this to our notations, please see [8] for more informations.

The definition of perceived affordances uses the notion of realizable interaction, in order to determine if two agents can participate in an interaction.

Definition 4. Let I be an interaction, and $x \prec \mathscr{S}$, $y \prec \mathscr{T}$ two agents. The tuple (I,x,y) is **realizable** (written $r(I,x,y)$) if and only if :

- $I \in a_{\mathscr{S}/\mathscr{T}}$, *i.e.* agents of $\mathscr{S}$ family are able to perform I with agents of $\mathscr{T}$ family;
- the conditions of I hold true with x as source and y as target.

A realizable tuple represents one interaction that an agent can initiate with a particular target agent. Moreover, an agents perceived affordances are the set of all interactions it can initiate in a given context. Thus, at a time t, the perceived affordances of the x agent are the set of all realizable tuples that x may perform.

Definition 5. Let $\mathbb{A}_t$ be the set of all agents in the simulation at a time t, and $x \in \mathbb{A}_t$. Then, the **perceived affordances** $\mathbb{R}_t(x)$ that x may perform at time t is the set :

$$\mathbb{R}_t(x) = \bigcup_{y \in \mathbb{A}_t} \bigcup_{I \in a_{x/y}} \{(I,x,y) | r(I,x,y)\}$$

4.2 Selection Unit

In reactive simulation, agents try in general to perform actions and interactions sequentially until a realizable one is found – *i.e.* they use nested if/else structures. We propose to use a similar principle in the SELECTION UNIT (see Fig. 2) : every possible interaction I between a source $\mathscr{S}$ and a target agent family $\mathscr{T}$ is assigned a priority $p(I,\mathscr{S},\mathscr{T})$, just as [12] did for classical actions (that do not involve any target). Selection takes place on interactions in decreasing order.

4.2.1 Interaction Selection Policies

Thanks to the interaction-oriented study of experiments, different policies used to select a tuple from a set of realizable tuples $\mathbb{R}_t^p(x)$ were identified. An interaction

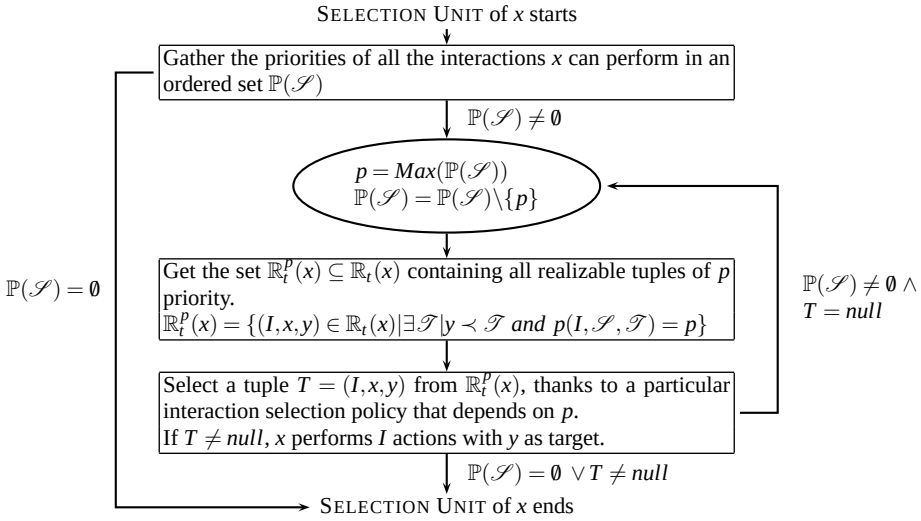

Fig. 2 Generic description of a reactive agent's (named *x*) SELECTION UNIT.

selection policy is decomposed in two parts : the nature of the selection, and on which elements the selection takes place. Indeed, the SELECTION UNIT can :

- **First** select the **interaction** that will occur, and **then** select its **target**. If the selected interaction is degenerate (*i.e.* is an action), no target selection takes place;
- **First** select the **target** on which an interaction will occur, and **then** select the **interaction** that will occur. This selection cannot involve degenerate interactions;
- Directly select a **tuple** $(Interaction/Target)$. If an interaction is degenerate, the corresponding tuple is only $(Interaction)$.

The selection of each element – interaction, target or tuple – has one nature chosen among three different ones :

- the element is selected at **random**;
- every element is given a **preference** value. The selected element is the one with the highest preference value. If more than one have this value, one of them is selected at random. This selection is intensively used in cognitive agents;
- every element e is given a **weight** $w(e) \in [0,1[$, and an interval $\mathscr{W}(e) \subset [0,1[$ of length $w(e)$ such that intervals are pairwise not intersecting. A number $r \in [0,1[$ is chosen at random. The selected element e is the element such that $r \in \mathscr{W}(e)$.

4.2.2 Design Guidelines of the SELECTION UNIT

To design an SELECTION UNIT containing fewer ambiguities, the modeler has :

- to provide priorities to every interaction an agent may perform;
- to provide for each couple (source agent family, priority) :

- on what element the selection is made (either *interaction then target*, or *target then interaction*, or *tuple*);
- the nature of each selection (either *random*, *by preference* or *weighted*);
- how preference and weights are computed.

Obviously, he has to understand what his choices imply. This kind of specification is possible only if interactions are at center of simulation, like in IODA [8].

5 Illustration on a Modeling Problem

Reactive MABS application fields widen everyday, and tackle very different problems. Among these appears chemistry, for which MABS provide more realistic diffusion behaviors than in numerical simulations. In this application field, one of the most difficult issue of multi-agent programing has to be tackled : defining the behavior of agents according to macroscopic rules. These rules use probabilities, and thus require to define stochastic behaviors for agents. These kind of behaviors introduce issues that do not appear in non-stochastic multi-agent simulations. Consequently, biases may occur in situations that might seem correct for regular simulations design methodologies. We illustrate this point on a modeling example, and show how our solution provides guidelines that leads modelers to identify biases.

5.1 The Modeling Problem

We consider simulations that describe chemical reactions. In those kinds of simulations, the behavior of agents is almost completely summarized in reaction rules. The modeling problem we consider is the implementation of the rules :

$$\mathrm{A} + \mathrm{B} \xrightarrow{k_1} \mathrm{C} \qquad \text{(R1)}$$

$$\mathrm{A} + \mathrm{D} \xrightarrow{k_2} \mathrm{E} \qquad \text{(R2)}$$

The reaction rule R1 means that an agent of A family can react with an agent of B family in order to from a new agent of C family. The two agents of A and B family are then destroyed. This reaction occurs only at a particular reaction rate k_1.

This rate is deeply bound with the probability that a R1 reaction occurs. For convenience, we consider that k_1 is the probability that the reaction R1 occurs[3]. The same goes for k_2 and R2.

Due to the lack of space, we focus only on the definition of A chemical species behavior. Moreover, since this is not the topic of this paper, we do not describe how A, B, C, D and E agents move in the environment.

This modeling problem is common in chemical reaction modeling, since chemical species are often involved in many different chemical reactions.

[3] Usually, this probability is obtained through computations, and is different from k_1.

5.2 First Encountered Problem

To implement such agents, the reaction rate of R1 and R2 have to be integrated into their behavior. Many different implementations of this behavior can be made (see [9]). These implementations correspond to different interpretations of reaction rates. Indeed, k_1 can :

- Either represent the probability that the reaction R1 occurs if an agent of A family is close to at least one agent of B family. In that case, the probability $\mathscr{P}(R1)$ that an agent of A family executes R1 depends only on the presence of one nearby agent of B family. Thus, if at least one agent of B family is close-by, $\mathscr{P}(R1) = k_1$;
- Or represent the probability that the reaction R1 occurs with one particular agent of B family. In that case, the probability that an agent of A family performs R1 depends on the number of nearby agents of B family. Thus, if there is n_b close-by agents of B family, $\mathscr{P}(R1) = 1-(1-k_1)^{n_b}$;

With our solution, the two interpretation correspond to two different SELECTION UNIT, where R1 and R2 have the same priority :

1. Either a *First interaction then target* selection, with a *weighted* selection for interactions (where $w(R1) = k_1$ and $w(R2) = k_2$), and a *random* one for targets;
2. Or a *Tuple* selection, where a tuple is realizable only if it meets the probability : the stochastic factor is tried in the condition of R1 and R2. The performed tuple is selected at random among realizable tuples.

The different interpretations appear clearly in the SELECTION UNIT. Indeed the use of the *First interaction then target* selection policy implies that the probability to trigger a reaction is independent from the number of neighboring agents. The use of the *tuples* selection policy implies that the probability to trigger a reaction is proportional to the number of neighboring agents.

5.3 Second Encountered Problem

Let us consider the second implementation, where the probability that an agent of A family performs R1 depends on the number of neighboring agents of B family.

Because an agent of A family that performs R1 disappears from the environment, such simulations are sometimes written like in figure 3.

This kind of implementation provides biased results. Indeed, agents of A family perform R2 only if they failed to perform R1. Thus, conditional probabilities are introduced : $\mathscr{P}(R2) = (1-\mathscr{P}(R1)) \times (1-(1-k_2)^{n_d})$

The greater k_1 or the density of agents of B family are, the greater the bias coming from conditional probabilities becomes. Thus, if the simulation is verified with low densities – or with a low reaction probability k_1 – the error has a weak impact on simulation results, and simulation seems unbiased.

```
ask every agent of A family [
    ;; List in its perceived affordances realizable R1
    ;; reactions with close-by B agents.
    ;; If at least one such tuple exists, the agent performs
    ;; one of them, and disappears from the environment.
]
ask the remaining agents of A family [
    ;; List in its perceived affordances realizable R2
    ;; reactions with close-by D agents.
    ;; If at least one such tuple exists, the agent performs
    ;; one of them
]
```

Fig. 3 An implementation example of our modeling problem

Even if this bias seems obvious, it exists in real implementations, for instance in the Netlogo [17] implementation of Henry-Michaelis-Menten kinetics [10].

If the reaction rates raise, experiments showed that a huge gap appeared between the reaction speed in biased implementations and the reaction speed in the unbiased ones. Because the reaction speed is at the center of many chemical reactions (like in Henry-Michaelis-Menten kinetics), such a bias is not acceptable. Thus, particular attention must be payed to this point.

With our solution, to obtain such a bias, different priorities must be given to R1 and R2. Thus, the fact that an agent of A family performs an R2 interaction only if it could not perform an R1 interaction appears explicitly in the SELECTION UNIT.

6 Conclusion

Designing simulations implies making implementation choices. These choices have a deep impact on simulation results, and might even introduce biases in them. To avoid this problem, modelers have to provide a precise description of implementation choices, to ensure the reproducibility of the model. This is only possible if the modeling methodology they use provides guidelines that elicits all these choices.

Current reactive MABS design methodologies do not specify clearly how the target of interactions are selected, because they do not provide both the separation between knowledge and action/interaction selection process of agents, and the reification of interactions.

Thanks to the IODA methodology – that meets the requirements mentioned above – we built guidelines to design the behavior of agents in order to solve this issue. The guidelines consist in providing knowingly a priority to every interaction an agent may perform, and then specifying for every priority :

- on what element the selection is made (either *first interaction then target*, *first target then interaction*, or *tuple*);
- the nature of each selection (either *random*, *by preference* or *weighted*);
- how preference and weights are computed.

The importance of such guidelines was illustrated in the case of chemical reactions simulations. It avoids the misuse of probabilities, that could introduce critical biases in results.

We uphold that such specifications provide a valuable communication support between modelers and computer scientists for the design of any kind of reactive simulations. It makes the interpretation of the model and its implementation clearer – and thus avoids ambiguities in the model – and the identification of possible biases easier.

Acknowledgements This research is supported by the FEDER and the "Contrat-Plan État Région TAC" of Nord-Pas de Calais.

References

1. C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology . In *Agent-Oriented Methodologies* . 2005.
2. 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.
3. R. A. Brooks. A robust layered control system for a mobile robot. *iEEE journal of robotics and automation*, 2(1), 1986.
4. Y. Demazeau. From interactions to collective behaviour in agent-based systems. In *Proceedings of ECCS'95*, Saint-Malo, France, 1995.
5. J. Epstein and R. Axtell. *Growing Artificial Societies*. Brookings Institution Press, Washington D.C., 1996.
6. M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of AAAI'87*, Seattle, WA, 1987.
7. 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.
8. Y. Kubera, P. Mathieu, and S. Picault. Interaction-oriented agent simulations : From theory to implementation. In *Proceedings of ECAI'08*, Patras Greece, 2008.
9. Y. Kubera, P. Mathieu, and S. Picault. Interaction selection ambiguities in multi-agent systems. In *Proceedings of IAT'08*, Sydney, Australia, 2008.
10. L. Michaelis and M. L. Menten. Die kinetik der invertinwirkung. *Biochemische Zeitschrift*, 49, 1913.
11. J. P. Müller and M. Pischel. Modelling interacting agents in dynamic environments. In *Proceedings of ECAI'94*, Amsterdam, The Netherlands, 1994.
12. N. J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1, 1994.
13. D. A. Norman. *The Psychology of Everyday Things*. Basic Books, 1988.
14. P.-M. Ricordel and Y. Demazeau. Volcano, a vowels-oriented multi-agent platform. In *Proceedings of CEEMAS '01*, London, UK, 2002.
15. I. van Langevelde, A. Philipsen, and J. Treur. Formal specification of compositional architectures. In *Proceedings of ECAI '92*, 1992.
16. 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.
17. U. Wilensky. Netlogo. http://ccl.northwestern.edu/netlogo, 1999. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.