Interaction Biases in Multi-Agent Simulations : An Experimental Study

Yoann Kubera, Philippe Mathieu, and Sébastien Picault

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
{yoann.kubera,philippe.mathieu,sebastien.picault}@lifl.fr

Abstract. How to ensure that two different implementations of a simulation will produce the same results ? In order to assure simulation reproducibility, some domain-independent functional unit must be precisely described. We show in this paper that the management unit that rules the participation of an agent in simultaneous interactions is one of them. Usually, many choices concerning this unit are made implicitly, even if they might lead to many simulation biases. We illustrate this issue through a study of biases that appear even in simple cases, due to a specification lack, and we propose as a solution a classification of interactions that makes those choices explicit.

1 Introduction

Computer simulation is the process leading from a domain-specific model to an operational model – through knowledge representation formalisms – and then from the operational model to its implementation in a given programming language, on a given simulation platform [1].

There is no consensus about what informations each model should contain, since the separation between domain-specific model, operational model and implementation is ambiguous itself [2, 3]. Thence, each step of the simulation design process involves choices – both explicit or implicit – regarding ambiguous parts of the previous step model. Those choices have a more or less dramatic influence on the execution and outcomes of the simulation. Since computer simulation aims at reinforcing, invalidating or comparing hypothesis on a particular phenomenon, the simulation biases these choices 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 agent's and simulation's architecture. To make their study easier, the architecture of a multi-agent simulation is considered through three almost independent functional units. In this paper, a first unit of this decomposition, called the *Agents and Environments Activation Unit* (*AEAU* for short) is considered. This

unit defines how the simulation engine manages the participation of an agent in different simultaneous interaction, depending on the nature of these interactions.

Interactions between agents – *i.e.* actions involving simultaneously two or more agents – are the source of simulation's 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 agents, many design choices concerning interactions are not explicit. In particular, the participation of an agent in interactions occurring at the same time is almost never tackled.

In this paper, the consequences of choices concerning the *AEAU* are elicited in three experiments. In each of them, a particular model is implemented with two different *AEAU*. The study of the different implementation's outcomes leads to an identification of different way to manage simultaneous interactions, that depend on their nature. We uphold that defining the nature of every interaction in a simulation determines precisely how the model's *AEAU* is supposed to work. Thus, it makes sure that the model is implemented without biases.

First, related work concerning the studied functional unit, the *Agents and Environments Activation Unit*, are presented in section 2. Then, the functional decomposition of a simulation, on which this paper's studies are based, and the notions used in that paper are presented in section 3. Section 4 describes the protocol followed by the experiments. Section 5 to 7 describe three experiments , which results are interpreted to identify interaction classes, and consequences of erroneous implementation choices. Eventually, section 8 summarizes the results of experiments, and proposes a classification of interaction in order to avoid implementation biases of the *AEAU*.

2 Related Works

In many simulation platforms, design is centered on agents and the actions they perform, rather than on the interactions that may occur between them. Consequently, the definition of what interaction an agent performs at a particular time does rarely take into account the interactions it already participates in as a target. This leads to many biases in the simulation and its outcomes.

For instance, [4] presents an ecosystem simulation where an agent may reproduce more than twice at the same time (once as a source, and one or more times as a target). This bias was identified in [5], and corrected with **a modification of the model** through the addition of a gestation period.

Yet, this issue is not restricted to ecosystem simulation. Indeed, it applies to every simulation where agents have to perform particular interactions at most once at a time (for instance agents that trade goods). Thus, it has to be dealt with **in the domain-independent architecture of the simulation** rather than in the models.

This problem lead Michel in [3] to manage interactions depending on their *Strong* or *Weak* nature. This solution is adequate if agents are the only participants in interactions. Thus, it does not solve the problem for interactions

between an agent and an object. Indeed, interactions like *Withdraw cash from an automated teller machine* may be performed simultaneously by two different agents with the same machine.

Weyns proposes in [6] a more refined solution through the qualification of the relationship between two actions[1]. Two actions may be *Independent*, *Joint*, *Concurrent* or *Influencing*. This solution manages interactions by getting from each agent *"I intend to perform the* I *interaction with the* A *agent as target"* like messages. A mastering unit then gathers these messages, finds out the relationship between them and executes compatible ones.

In spite of its undeniable advantage of concurrent and influencing interactions handling, this solution has a major issue. Indeed, because interactions not compatible with already occurring interactions are not considered during decision making, an agent may try to perform an impossible interaction. Thus, the agent performs nothing at that time, even if it another interaction is possible.

To fill this gap, simultaneous interactions have to be considered at decision making. This requires an interaction-oriented design of decision making, like the one shortly presented in the next section.

3 Multi-agent Simulations

Even if the application domains of multi-agent simulations are heterogeneous, they can be split into different and weakly dependent functional units [7, 8], like agents scheduling, communications, modification conflicts solving, *etc.*

We consider here a particular decomposition of a simulation (see Fig. 1) in three main units, called *Agents and Environments Activation Unit* (*AEAU*), *Interactions Definition Unit* (*IDU*) and *Interaction Selection Unit* (*ISU*).

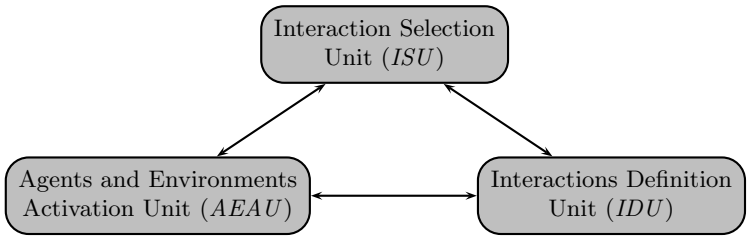

Fig. 1. The three main functional units of a multi-agent simulation.

This kind of separation in different software units is usual in cognitive agent architectures with plans like the *Act-R* [9] or *Soar* [10] language, where knowledge representation is at the center of the simulation, but does not exist in

[1] Although the author uses the term "action", it keeps the same meaning than our "interaction" (see section 3.1).

reactive simulation platforms. Moreover the notion of interaction – *i.e.* semantic block of actions involving simultaneously a fixed number of agents (see Sect. 3.1) – is generally hard-coded in the behavior of agents. 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.

3.1 An Interaction-Oriented Design of Simulations

The definition of interactions, and how they are integrated in the knowledge of agents, are based on *IODA* concepts [11]. 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 [11] 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. Thus, in this paper, the word ***agent*** refers to 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.

Without that ambiguity concerning the word agent, we define interactions as following.

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.*
Thus, 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 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.

The declaration of all possible interactions is summarized in a representation called interaction matrix, which describes what instances of different agent families are able to perform together.

Definition 3. *The* **interaction matrix** $\mathbb{M}$ *of a MABS summarizes all possible interactions between agents.*

If $\mathcal{S}$ and $\mathcal{T}$ are agent families, we note $a_{\mathcal{S}/\mathcal{T}}$ 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.

Thus, $\mathbb{M} = \cup_{\mathcal{S},\mathcal{T}} \left(a_{\mathcal{S}/\mathcal{T}}\right)$. By extension, if $x \prec \mathcal{S}$ and $y \prec \mathcal{T}$, $a_{x/y}$ and $a_{\mathcal{S}/\mathcal{T}}$ are equivalent notations.

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 $x \prec \mathcal{S}$, $y \prec \mathcal{T}$ two agents. The tuple (I, x, y) is* **realizable** *if and only if :*

- *$I \in a_{\mathcal{S}/\mathcal{T}}$ (agents of $\mathcal{S}$ family have the ability to perform I on agents of $\mathcal{T}$ family) 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 $\mathbb{R}_t(x)$ of all realizable tuples that x may perform at time t is :
$\mathbb{R}_t(x) = \cup_{y \in \mathbb{A}_t} \cup_{I \in a_{x/y}} \{(I, x, y) | r(I, x, y)\}$

For convenience, and to unify knowledge representation, we use *degenerate interactions.*

Definition 6. *A* **Degenerate Interaction** *is an interaction with no target agent family.*

Degenerate interactions represent interactions between a source agent and an implicit target (the agent itself or the environment), like to MOVE or to DIE.

3.2 Functional Decomposition of a Multi-agent Simulation

A simulation is a repetition of 3-steps sequences, where each step exploits a different functional unit :

1. the *Agents and Environments Activation Unit* either selects the next agent that will behave, and does 2, or updates the environment and does 1 again;
2. the *Interaction Definition Unit* lists the space of all possible interactions the selected agent may perform as a source – *i.e.* all realizable tuples *(Interaction, Selected Agent, Target agents)*, also corresponding to the perceived affordances of the source, as defined in [12];
3. the *Interaction Selection Unit* selects a tuple among the space of all possible interactions, according to a particular policy, and executes the related action.

Each unit is in charge of a specific feature of a multi-agent simulation :

The *AEAU* tells when agents or the environment may act (or update their state), the time elapsed between their actions, what to do if an agent tries to interact with an already acting agent, *etc.* It describes all time-related elements in the simulation.

The *IDU* lists all interactions in the simulation, under what conditions and between what kind of agents they are possible, and what actions they launch. It is the set of all possible behaviors, defined independently from agents specificities.

The *ISU* describes the cognitive or reactive process an agent uses to select which interaction it performs, and, if many are possible, decides among them the one to perform.

We already argued in [11] for the advantage of agent-independent defined interactions and proposed a formal definition for it, thus creating a software separation between the *IDU*– which is domain dependent – and both *ISU* and *AEAU*.

4 Experimental Frame

The goal of this paper is to measure to what extent modifications of the Agents and Environments Activation Unit (*AEAU*) may change simulation outcomes, and how an adequate one may avoid simulation biases. This point is illustrated through three experiments, each confronting two different implementation of the *AEAU*. Thus, the only variating parameter in an experiment is the *AEAU*: its Interaction Definition Unit (*IDU*) and Interaction Selection Unit (*ISU*) remain the same.

This section presents the *IDU*, *ISU* and protocol used in our experiments.

4.1 Interaction Definition Unit

The Interaction Definition Unit (*IDU*), which defines all domain-dependent informations, will change from one experiment to the other. In order to make the comprehension of our examples easier, every experiment deals with the same overall simulation problem : the evolution of an ecosystem containing predators and preys. Please note that experiments provide only an illustration of the general issue we deal with. The solutions presented in this paper are obviously not restricted to that particular simulation, and do not avoid only the three emphasized biases.

4.2 Interaction Selection Unit

The Interaction Selection Unit (*ISU*), which corresponds to agent's decision making process, will keep the same architecture in the three experiments.

The architecture we use is the mostly used one for reactive agents : a subsumption-like architecture [13] that tries every interaction sequentially until a realizable one is found. Every source agent gives to every interaction it can

perform a priority value, which denotes the order it tries interactions (see Fig. 2).

Let $\mathbb{R}_z(a)$ = the set of all realizable interactions a may perform**.**
Let $\mathbb{P}$ = the decreasing set of all priorities a gave to the interactions it can perform**.**
Let $\mathbb{L} = \emptyset$**.**
For All p **in** $\mathbb{P}$ **Do :**
| **For All** (I, a, t) **in** $\mathbb{R}_z(a)$ **Do :**
| | **If** I has p priority for a **Then :**
| | | **Set** $\mathbb{L} = \mathbb{L} \cup \{(I, a, t)\}$**.**
| | **End If.**
| **End For.**
| **If** $\mathbb{L} \neq \emptyset$ **Then :**
| | a performs a tuple of $\mathbb{L}$ chosen at random**.**
| | The algorithm stops**.**
| **End If.**
End For.
a performs nothing**.**

Fig. 2. Algorithm of the Interaction Selection Unit (*ISU*) used in the experiments of this paper. It defines how an agent a chooses what interaction it performs at a time z.

4.3 Experimental Protocol

The experimental protocol used in this paper is :

1. first, the aim of the experiment is outlined;
2. then, the *IDU* and *ISU* used by both implementations of this experiment are defined;
3. next, the two *AEAU* used in the experiment, and experiment's initial conditions, are described;
4. eventually, the results of the execution of both implementation of the experiment are presented and discussed. From this discussion, an interaction class is emphasized to avoid a possible simulation bias.

Each experiment illustrates a problem in which previously presented solutions introduce biases, and proposes correct solutions. Thus we incrementally define classes of interactions, specified in the modeling step, in order to produce correct implementation and outcomes.

Please note that all experiments presented below are voluntary basic to stress out where the problems lie : they obviously do exist in more complex situations as well.

For All z **in** $[1, MAX]$ **Do :**
| Update the environment.
| **For All** a **in** all agents in the environment at time z **Do :**
| | **Let** $\mathbb{R}_z(a) =$ all realizable tuples that a may perform**.**
| | **Let** $(I, a, t) =$ a tuple of $\mathbb{R}_z(a)$ selected with a particular ISU**.**
| | **If** $(I, a, t) \neq null$ **Then :**
| | | Execute I with a as source and t as target.
| | **End If.**
| **End For.**
End For.

Fig. 3. *"Naive* AEAU *Algorithm"*, where a simulation is executed in Max simulation steps, during which every agent may participate in at least one interaction.

For All z **in** $[1, MAX]$ **Do :**
| Update the environment.
| **For All** a **in** all agents in the environment at time z **Do :**
| | Tag a as *operative*
| **End For.**
| **For All** a **in** all agents in the environment at time z **Do :**
| | **Let** $\mathbb{R}_z(a) =$ all realizable tuples that a may perform**.**
| | **For All** (I, a, t) **in** $\mathbb{R}_z(a)$ **Do :**
| | | **If** a is not operative **or** t is not operative **Then :**
| | | | Remove (I, a, t) from $\mathbb{R}_z(a)$
| | | **End If.**
| | **End For.**
| | **Let** $(I, a, t) =$ a tuple of $\mathbb{R}_z(a)$ selected with a particular ISU**.**
| | **If** $(I, a, t) \neq null$ **Then :**
| | | Execute I with a as source and t as target.
| | | Tag a and t as *not operative*.
| | **End If.**
| **End For.**
End For.

Fig. 4. *"Single Interaction* AEAU *Algorithm"*, where a simulation is executed in Max simulation steps, during which every agent may participate in at least one interaction.

5 First Experiment : Multiple Participation to Interactions Bias

This experiment studies the limits of the usual naive algorithm and introduces as a solution a first interaction class called *exclusive* interaction.

Model used : This simulation studies an ecosystem composed by grass and sheep. Because sheep can move, classical analytical models cannot be used to model the population of species : this simulation requires multi-agent systems.

The environment is a two dimensions toroidal continuous space split into unitary square parcels. Every parcel $\mathcal{P}$ has a grass quantity attribute $q(\mathcal{P})$ in-

creasing of one unit at every environment update. $\mathcal{P}$ is said containing grass when $q(\mathcal{P}) > 0$. If $\mathcal{P}$ is emptied by an agent, then $q(\mathcal{P}) = 1 - r_{grass}$ (*i.e.* r_{grass} is the time grass needs to grow) . A sheep $\mathcal{S}$ is an agent with an energy attribute $e(\mathcal{S})$ representing its health which can perform the interactions :

1. To **Die** if $e(\mathcal{S}) \leq 0$. Then :
   - $\mathcal{S}$ is removed from the environment.
2. To **Reproduce** with an other sheep $\mathcal{S}'$ at a maximal distance of 1 from $\mathcal{S}$ if $e(\mathcal{S}) > 0$ and $e(\mathcal{S}') > 0$. Then :
   - A new sheep $\mathcal{S}''$ is created at $\mathcal{S}$'s location, and $e(\mathcal{S}'') = Min(e(\mathcal{S}), e_{repr}) + Min(e(\mathcal{S}'), e_{repr})$, where e_{repr} stands for the energy consumed by reproduction.
   - $e(\mathcal{S})$ and $e(\mathcal{S}')$ are decreased by e_{repr}.
   - $\mathcal{S}$, $\mathcal{S}'$ and $\mathcal{S}''$ execute the WANDER interaction (see below).
3. To **Eat** grass on $\mathcal{S}$'s parcel if it contains some. Then :
   - $e(\mathcal{S})$ increases from e_{eat}, where e_{eat} is the energy gained by eating.
   - $\mathcal{S}$ empties the parcel he is onto.
4. To **Wander** with no conditions. Then :
   - $\mathcal{S}$ turns his-self from an angle in $[-\pi, \pi[$ and moves forward from 1 unit.
   - $e(\mathcal{S})$ decreases from e_{wan}, where e_{wan} is the energy consumed by moving.

Sheep behave by using the order $1 > 2 > 3 > 4$ in their *ISU*, thus they first try to DIE, if they don't, they try to REPRODUCE, if they don't, they try to EAT, ...

Experimental design : We used this model in a 33×33 environment containing 1089 parcels, where 30% have $q(\mathcal{P}) = 0$ and 70% $q(\mathcal{P}) \in]-r_{grass}, -1]$, and 70 sheep such that $e(\mathcal{S}) = 2 \times e_{repr}$. We also set $r_{grass} = 10$, $e_{repr} = 15$, $e_{wan} = 2$ and $e_{eat} = 7$.

Figure 5 compares the evolution of the sheep population of this model implemented with respectively the *naive* (*single interaction*) algorithm as *AEAU*, from Fig. 3 (from Fig. 4).

Results and Discussion : The experiment using the *naive* algorithm produces in overall 68 more sheep than the one using the *single interaction* algorithm.

This difference lies in the number of interactions an agent may participate in during a simulation step. In the *naive* algorithm, a sheep targeted by a REPRODUCE interaction can be the source of an other interaction. On the opposite, in the *single interaction* algorithm an agent participates at maximum in one interaction, either as a source or as a target. This difference has a great impact on the sheep energy dynamics, their reproduction and death rate, and their density, and, as a consequence, on the sheep population dynamics.

In this simple experience, this difference may be considered as the result of different interpretations of the model, thus it cannot be considered as a bias. Nevertheless, it corresponds to a bias in other experiences, as shown in [3] : if a sheep participates in more than one reproduction per time step, then the sheep reproduction probability is different from the one designed in the model.

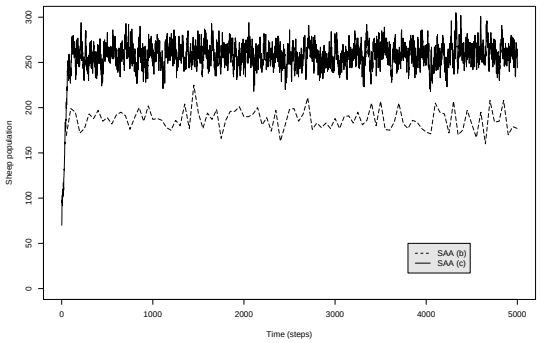

Fig. 5. Sheep population evolution against time (in simulation steps) respectively displayed in dots (in line), for the model 5 implemented with the *naive* (*single interaction*) algorithm of Fig. 3 (Fig. 4).

We outlined with this experience that the number of interactions an agent can participate in has to be restricted. To cope with this problem, we identified a first interaction class called *exclusive* interactions. An agent can participate in such an interaction only once per time step, either as source or as target.

6 Second Experiment : Single Participation to Interactions as Target Bias

This section illustrates a simulation problem concerning the target of an interaction that *exclusive* interactions alone are too restrictive to solve.

Model used : This model adds to the one of experiment 5 a new agent named wolf, and a new interaction to a sheep $\mathcal{S}$:

5. To **Flee** from a wolf $\mathcal{W}$ at a maximal distance of 10 from $\mathcal{S}$. Then :
   - $\mathcal{S}$ turns his back towards where $\mathcal{W}$ is and moves forward from 1 unit.
   - $e(\mathcal{S})$ decreases from e_{wan}.

Sheep behave using the order $1 > 5 > 2 > 3 > 4$ in their *ISU*, and wolves only Wander in the environment.

In this simulation, sheep Flee systematically wolves. As a consequence, wolves are supposed to be at the center of an empty area.

Experimental design : They are the same as in experiment 5, except that there is a wolf at a random position, and that the simulation is implemented with :

- firstly with the *single interaction* algorithm as $AEAU$ (from Fig. 4);
- then with the *parallel interactions* algorithm as $AEAU$ (from Fig. 6) where
  - Flee is from $\mathcal{I}_2$ interaction class;
  - other interactions are from $\mathcal{I}_1$ class.

The outcomes of such implementations of this model are displayed in Fig. 7.

For All z **in** $[1, MAX]$ **Do :**
| Update the environment.
| **For All** a **in** all agents in the environment at time z **Do :**
| | Tag a as *operative*
| **End For.**
| **For All** a **in** all agents in the environment at time z **Do :**
| | **Let** $\mathbb{R}_z(a)$ = all realizable tuples that a may perform**.**
| | **For All** (I, a, t) **in** $\mathbb{R}_z(a)$ **Do :**
| | | **If** I is from $\mathcal{I}_1$ class **and** (a is not operative **or** t is not operative) **Then :**
| | | | Remove (I, a, t) from $\mathbb{R}_z(a)$
| | | **Else If** I is from $\mathcal{I}_2$ class **and** a is not operative **Then :**
| | | | Remove (I, a, t) from $\mathbb{R}_z(a)$
| | | **End If.**
| | **End For.**
| | **Let** (I, a, t) = a tuple of $\mathbb{R}_z(a)$ selected with a particular ISU**.**
| | **If** $(I, a, t) \neq null$ **Then :**
| | | Execute I with a as source and t as target.
| | | Tag a and t as *not operative.*
| | **End If.**
| **End For.**
End For.

Fig. 6. *"Parallel Interaction* AEAU *Algorithm"*, where a simulation is executed in Max simulation steps, during which every agent may participate in at least one interaction.

Results and Discussion : The *parallel interactions* algorithm produces the expected result (right on Fig. 7), and the *single interaction* algorithm (left on Fig. 7) is obviously biased : there is no empty halo around the wolf.

The difference lies in the number of interactions a wolf can undergo. When a wolf is the target of a Flee interaction, it is set not operative, thus it cannot be the target of an other Flee interaction. Consequently, a wolf is fled once per simulation step, and other sheep behave as if there was no wolf.

We outlined with this experiment that all interactions do not put the same restrictions onto their target agent. Interaction classes have to reflect this difference, thus, in addition to *exclusive* interactions, we introduce *parallel* interactions, where agents may be targeted as many times as needed.

7 Third Experiment : Single Participation to Interactions as Source Bias

This section illustrates a simulation problem concerning the source of an interaction that both *exclusive* and *parallel* interactions are too restrictive to solve.

Model used : This experiment aims to study the propagation of a disease in a model similar to experiment 5. In this experiment, an ill sheep infects an healthy one with a p_{inf} probability, only if they are close enough (distance of 2). Thus,

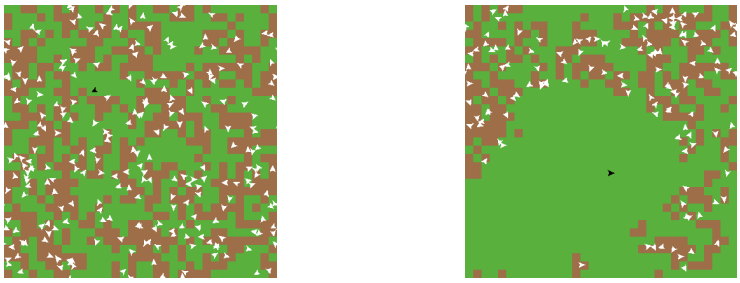

Fig. 7. Outcomes screenshot of experiment 6 respectively implemented with the *single interaction* (*parallel interactions*) algorithm from Fig. 4 (from Fig. 6), displayed to the left (right), where respectively light (dark) arrows are sheep (wolves), and light (dark) squares are empty (full) parcels.

a sheep $\mathcal{S}$ possesses an attribute called $i(\mathcal{S})$ set to true if it is infected by the disease, and can perform :

6. To **Infect** a sheep $\mathcal{S}'$ at a maximal distance of 2 from $\mathcal{S}$ if $i(\mathcal{S}) = True$, $i(\mathcal{S}') = False$ and $random(100) < p_{inf}$. Then :
   - $i(\mathcal{S}') = True$

Sheep behave using the order $1 > 6 > 2 > 3 > 4$ in their *ISU*.

Experimental design : They are the same as in experiment 5, except that $p_{inf} = 50\%$, that 50% of the initial population of sheep is infected, and that the simulation is implemented with :

- firstly with the *parallel interactions* algorithm as $AEAU$(from Fig. 6) where
  - INFECT is from $\mathcal{I}_2$ interaction class;
  - the other interactions are from $\mathcal{I}_1$ interaction class;
- then with the *multiple interactions* algorithm as $AEAU$(from Fig. 8) where
  - INFECT is from $\mathcal{I}_3$ interaction class;
  - the other interactions are from $\mathcal{I}_1$ interaction class;
  - INFECT checks that a source agent did not already tried to infect a target within the current time step.

Figure 9 compares the evolution of the effective infection rate of healthy sheep by infected sheep – *i.e.* the ratio between the number of times an ill sheep infects an healthy one and the number of its neighboring healthy sheep.

Results and Discussion : The implementation using the *parallel interactions* algorithm (from Fig. 6) produces an effective infection rate greatly below the expected 50% obtained with the *multiple interactions* algorithm (from Fig. 8).

This difference comes from the *parallel interactions* algorithm, where an agent is the source of at maximum one interaction. As a result, an ill sheep infects at maximum one healthy sheep per simulation step. Thus, an ill sheep, which has n healthy ones close to it, cannot reach the average $\frac{n}{2}$ infection number.

For All z **in** $[1, MAX]$ **Do :**
| Update the environment.
| **For All** a **in** all agents in the environment at time z **Do :**
| | Tag a as *operative*
| **End For.**
| **Let** $\mathbb{L}$ = all agents in the environment at time z**.**
| **For All** a **in** $\mathbb{L}$ **Do :**
| | Remove a from $\mathbb{L}$
| | **Let** $\mathbb{R}_z(a)$ = all realizable tuples that a may perform**.**
| | **For All** (I, a, t) **in** $\mathbb{R}_z(a)$ **Do :**
| | | **If** I is from $\mathcal{I}_2$ class **and** a is not operative **Then :**
| | | | Remove (I, a, t) from $\mathbb{R}_z(a)$
| | | **End If.**
| | **End For.**
| | **Let** (I, a, t) = a tuple of $\mathbb{R}_z(a)$ selected with a particular ISU**.**
| | **If** $(I, a, t) \neq null$ **Then :**
| | | Execute I with a as source and t as target.
| | | **If** I is from $\mathcal{I}_2$ class **Then :**
| | | | Tag a and t as *not operative*
| | | **Else If** I is from $\mathcal{I}_3$ class **Then :**
| | | | Add a to $\mathbb{L}$
| | | **End If.**
| | **End If.**
| **End For.**
End For.

Fig. 8. *"Multiple Interaction* AEAU *Algorithm"*, where a simulation is executed in Max simulation steps, during which every agent may participate in at least one interaction.

We outlined with this experiment that all interactions do not put the same restrictions onto their source agent. Interaction classes have to reflect this difference, thus, in addition to *exclusive* and *parallel* interactions, we introduce *systematic* interactions, in which agents may participate as many times as needed.

8 Experiments Summary : A Classification of Interactions

Through these experiments, we have shown that the model has to answer the question *"is the source (or target) agent of an interaction allowed to be the source (or target) of an other interaction during a simulation step ?"*. Otherwise the lack of specifications leads to ambiguities from which many biases may result.

As a solution, we identified three interaction classes, each answering differently to the question above (a summary is proposed in Fig. 10). The association of a class to each interaction in a model describes explicitly how they are managed, and thus avoids many biases at implementation. Those classes are [2] :

[2] Note that in those definitions, *"later"* means *"later in the same simulation step"*

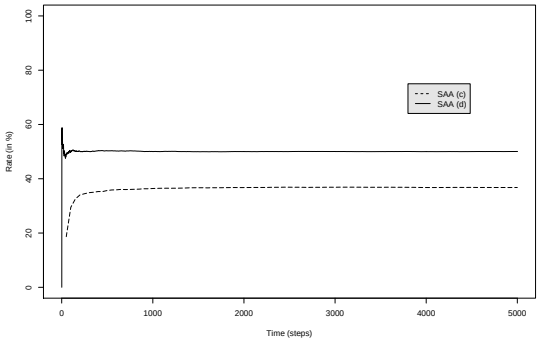

Fig. 9. Effective infection probability evolution against time (in simulation steps) respectively displayed in dots (in line) for the model 7 implemented with the *parallel interactions* (*multiple interactions*) algorithm from Fig. 6 (from Fig. 8).

		Exclusive		Parallel		Systematic	
		S	T	S	T	S	T
Exclusive	S				×	×	×
	T				×	×	×
Parallel	S				×	×	×
	T	×	×	×	×	×	×
Systematic	S	×	×	×	×	×	×
	T	×	×	×	×	×	×

Fig. 10. Summary of the interaction classes an agent can still participate in after participating in a particular interaction. The gray cell is read *"The source (S) of an exclusive interaction can be later in the simulation step the target (T) of a parallel interaction"*.

Definition 7. *An* **exclusive interaction** *is an interaction that forbids :*

- *to source and target agents to be later the source or target of an other exclusive interaction;*
- *to source and target agents to be later the source of a parallel interaction.*

In the experiments and in the algorithms of Fig. 4 and 6, it corresponds to $\mathcal{I}_1$ interaction class. It is the case of the REPRODUCE interaction.

Definition 8. *A* **parallel interaction** *is an interaction that forbids :*

- *to source agents to be later the source or target of an exclusive interaction;*
- *to source agents to be later the source of an other parallel interaction.*

In the experiments and in the algorithms of Fig. 6 and 8, it corresponds to $\mathcal{I}_2$ interaction class. It is the case of the FLEE interaction.

Definition 9. *A* **systematic interaction** *is an interaction with no restrictions on agents.*

In the experiments and in algorithm of Fig. 8, it corresponds to $\mathcal{I}_3$ interaction class. It is the case of the INFECT interaction.

These interaction classes describe part of the algorithms used to process interactions at implementation. Thus, knowledge on which interaction classes are present in an operational model determines if a simulation platform is fit to implement it. For instance, a simulation platform like *Netlogo*[14] which implements by default the *naive* algorithm of Fig. 3, is fit to implement models with only parallel interactions, or else the user has to develop his own *AEAU*.

The implementation of such specifications is made easier by a software separation between *AEAU*, *IDU* and *ISU*. Indeed, it forces the user to choose interaction classes explicitly, and thus forces to understand the underlying algorithms. It is the case of the *IODA* methodology and the *JEDI* simulation platform [11] where this separation is made by the reification of interactions through the whole simulation process.

Note that even if these classes are defined for discrete time simulations – *i.e.* with simulation steps – they remain valid for other simulations. In that case, they answer the question *"is the source (or target) agent of an executing interaction allowed to be the source (or target) of an other interaction ?"*.

9 Conclusion

Most simulations assume and compare hypothesis on a given phenomenon. Thus the biases that may result from modeling and implementation choices must be identified and quantified. Otherwise simulations remain ambiguous and are neither reproducible nor viable.

In this paper we have shown that the reproducibility of a simulation is not possible without specifying a particular domain-independent functional unit, called Agents and Environments Activation Unit, underlying any simulation.

This unit indicates to what interactions an agent can participate in at the same time. Experiments showed that the lack of specifications of the particularities of these interactions may introduce biases in simulation outcomes. Indeed, as an example, the target of a reproduction behavior cannot perform an other interaction, otherwise an agent may reproduce twice at the same time.

To solve this kind of problems, we propose to characterize each interaction of the model with one of the three interaction classes : *exclusive* interactions that forbid source and target agents to perform any other interaction as a source, or to participate in an other *exclusive* interaction; *parallel* interactions that forbid source agent to perform any other interaction as a source; and *systematic* interactions that constrains no agents.

Taking into consideration these classes while conceiving the model removes ambiguities that would have led to biases. Without the specification of this point, two different developers will likely obtain very different outcomes for the same model.

References

1. Fishwick, P.A.: Computer simulation: growth through extension. Trans. Soc. Comput. Simul. Int. **14**(1) (1997)
2. Nuno, D., Sichman, J.S.a., Coelho, H.: Towards an emergence-driven software process for agent-based simulation. In: Proceedings of MABS 2002. (2002)
3. Michel, F., Gouach, A., Ferber, J.: Weak interaction and strong interaction in agent based simulations. In: Proceedings of MABS 2003, Melbourne, Australia (2003)
4. Epstein, J., Axtell, R.: Growing Artificial Societies. Brookings Institution Press, Washington D.C. (1996)
5. Lawson, B., Park, S.: Asynchronous time evolution in an artificial society mode. Journal of Artificial Societies and Social Simulation (2000)
6. Weyns, D., Holvoet, T.: Model for simultaneous actions in situated multi-agent systems. In: Proceedings of MATES 2003, Erfurt, Germany (2003)
7. Demazeau, Y.: From interactions to collective behaviour in agent-based systems. In: Proceedings of ECCS'95, Saint-Malo, France (1995)
8. Weyns, D., Parunak, H., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems: State-of-the-art and research challenges. In: Environments for Multiagent Systems, New York, NY, USA (2004)
9. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychological Review **111**(4) (2004)
10. Newell, A.: Unified theories of cognition. Harvard University Press, Cambridge, MA, USA (1994)
11. Kubera, Y., Mathieu, P., Picault, S.: Interaction-oriented agent simulations : From theory to implementation. In: Proceedings of ECAI 08, Patras Greece (July 2008)
12. Norman, D.A.: The Psychology of Everyday Things. Basic Books (1988)
13. Brooks, R.A.: A robust layered control system for a mobile robot. iEEE journal of robotics and automation **2**(1) (March 1986) 14–23
14. Wilenski, U.: Netlogo. Center for connected learning and computer-based modeling, http://ccl.northwestern.edu/netlogo/ (1999)