AROUND THE EMPIRICAL AND INTENTIONAL REFERENCES OF AGENT-BASED SIMULATION IN THE SOCIAL SCIENCES

Nuno David
Instituto de Ciências do Trabalho e da Empresa, ISCTE/DCTI, Lisboa, Portugal

Helder Coelho
Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa (FCUL), Portugal

Keywords: Agent-based social simulation, epistemological perspectives, program verification, intentional verification.

Abstract: The difficulties in constructing and analyzing simulations of social theory and phenomena, even the most simplified, have been underlined in the literature. The *experimental reference* of simulation remains ambiguous, insofar as the logic of its method turns computer programs into something more than a tool in the social sciences, defining them as the experimental subject itself. The goal of this paper is to construct a methodological perspective that is able to conciliate the formal and empirical logic of program verification in computer science, with the interpretative and multiparadigmatic logic of the social sciences. This is a condensed and revised version of David et al. (2006). We demonstrate that the method of simulation implies at least two distinct types of program verifications, which we call empirical and intentional verification. Furthermore, we clarify the experimental reference of simulation by demonstrating that the process of intentional verification is contingent upon both the behaviors of the programs and the observed social phenomena.

1 SCIENTIFIC KNOWLEDGE AND COMPUTER PROGRAMS

The role of simulation has acquired a renewed importance in the social sciences. From an interdisciplinary perspective, the discipline of Agent-Based Social Simulation (ABSS) finds its origin in the intersection of the social and the computer sciences (see e.g. David et al., 2004). Whereas from an interdisciplinary viewpoint the discipline stresses the encounter of two distinct scientific logics, there are undoubtedly good reasons to maintain methodology in the research agenda.

For some, the use of formal models, resulting from the computational nature of simulation, has been considered not only an addition to the established methods but the basis for the emergence of proper social sciences. Even so, the difficulties in constructing and analyzing simulations, even the most simplified, have been underlined in the literature, which raises some interesting questions around the kind of scientific knowledge that simulation is providing.

On the one hand, the *experimental reference* of simulation remains ambiguous, insofar as the logic of its method turns computer programs into something more than a tool in the social sciences, defining them as the experimental subject itself – it is programs, and not the social phenomena they presumably represent, that are executed and tested. On the other hand, the formal tradition of the classic theory of computation creates a semantic gap between the formal interpretation of program executions, derived from the Church-Turing thesis, and the stakeholders' informal interpretations, acquired through direct observation of simulations.

These difficulties suggest the elaboration of an alternative vision as to the role played by computer programs in scientific knowledge. How are we to reconcile the methodologically diverse and multiparadigmatic social sciences with a computer science that has been able to attain a larger consensus in regard to the conception of scientific truth or validity?

This paper aims to present a theory of computation for simulating social theory and phenomena, especially with reference to its epistemological basis, limits and particular kind of scientific credibility. We demonstrate that the method of ABSS implies at least two distinct kinds of program verifications, which we call *empirical* and *intentional* verification.

The method of this paper is that of philosophical analysis. This is a condensed and revised version of David et al. (2005). We address questions that are presently found important in the field, such as: "what kind of credibility can the execution of a computer program ensure for analyzing a social phenomenon?"

By demonstrating that it is the intentional verification of programs that is doubly contingent with both the behaviors of the programs and the social phenomena, we clarify the experimental reference of simulation, and identify a new category of knowledge we can acquire about computer programs.

2 BACKGROUND: CAUSAL CAPABILITY OF PROGRAMS

The role of this section is to introduce an assumption about computer science epistemology, namely that the semantic significance of computer programs conveys a causal capability which affects the performance of machines if those programs are compiled, loaded and executed.

In computer science, the notion of scientific truth or validity has been related to an old debate, which confronts researchers advocating the use of formal methods for verifying programs and those defending the use of empirical methods. By the end of the Eighties, the debate became especially eloquent after James Fetzer published the article "Program Verification: The Very Idea." Fetzer's (1988) aim was to reject the idea of formal verification as a means of verifying programs, demonstrating that computer programming is also a branch of applied mathematics ruled by empirical research.

Fetzer's argument consisted of distinguishing programs as encodings of algorithms from the logical structures that they represent. The causal capability of programs becomes clear once we realize that, rather than one model, we use many models in the implementation of a single program. Let us think of a computer program as a textual and static entity, which may be read, edited, printed. Given the existence of high-level programming languages, we can think of a program as a model of a potential solution of a problem, where the language functions as a model of an abstract machine.

Thus, insofar as programs are written in languages that model abstract machines, it remains the case that there may or may not be a suitable correspondence between the commands that occur within the language and the operations that are performed by some physical machine. In fact, high-level programming languages are related to physical machines by means of compilers and interpreters. The advantage of programming by means of high-level languages is that there is a one-many relationship between the commands that can be written in a high-level language, and the counterpart operations that are performed by a machine executing them, on the basis of their translation into machine language. The function of interpreters and compilers is to create a causal mechanism so that programs written in high-level languages may be executed by target machines whose operations are causally affected by machine code, which usually consists of sequences of zeros and ones.

Low-level programming languages therefore play two roles: first, that of an abstract machine, in a way analogous to high-level languages but where, second, unlike high-level languages, there is a one-to-one causal relationship between the commands that occur within a programming language and the operations performed by a target machine. The programming language stands for a virtual machine that may be understood as an abstract entity, which may or may not be causally connected with a target machine.

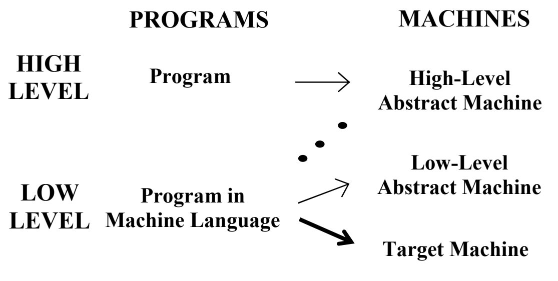
PROGRAMS
MACHINES
HIGH LEVEL
Program
High-Level Abstract Machine
LOW LEVEL
Program in Machine Language
Low-Level Abstract Machine
Target Machine

Figure 1: Programs and languages as models, according to Fetzer (1999).

From this point of view, the implementation of a program can be seen as the action of embedding models in other models, where the notion of embedding may be envisioned as a logical or a causal relation. Figure 1 reproduces Fetzer's (1999) notion. The thin arrows represent a possible relation between a program and the abstract machine represented by a programming language. The thick arrow represents an actual relation between a low-level program and a target machine. The series of three dots stands for the possible existence of compilers and interpreters that effect some causal connection between programs/machines at different levels.

Although the figure shows the set of models in the general case of computer science, it would be possible to identify additional levels of model embedding for the specific case of simulation – for instance, by realizing the existence of simulation platforms, as well as their corresponding simulation languages. At any rate, the causal connection between a simulation program and a target machine can be identified at various levels, e.g., through simulation platforms, compilers or interpreters.

3 INTENTIONAL CAPABILITY OF PROGRAMS

Our aim is to show that ABSS programs possess an intentional capability that surpasses their causal capability. Our argument is organized into four parts. We first show that the experimental reference of simulation involves more complicated aspects than ordinary computer science. We then define the meaning of intentional capability of programs. Next, we concretize our argument with canonical examples of simulations found in literature. Finally, we analyze the role of the intentional capability of programs in ABSS.

3.1 The Experimental Reference of ABSS

A question remaining in the epistemology of ABSS consists in characterizing what a scientific experiment consists of. If computer science is regarded as an empirical science, then the experimental reference of any theory about the computation of a program in an abstract machine consists in executing that program in a target machine. In the software production process, this phase is known as program verification. Thus, for the classical theory of computation, the role of verification is to ascertain the validity of certain outputs as a function of given inputs, regardless of any interpretation given in terms of any theory or any phenomenon not strictly computational. Another kind of experimental evaluation, which may be confounded with the latter, is called program validation. The role of validation is to ascertain that the execution of a program behaves according to the relatively arbitrary expectations of the program end-users.

As we have mentioned, the implementation of a program involves a sequence of models embedded in a target machine. Each one of these models can suggest an alternative interpretation for verifying and describing the behavior of the program. For instance, the vocabularies of the low-level abstract machine (e.g., memory registers, bit logical operations) are neither identical to the vocabularies of the high-level abstract machine (e.g., complex data structures, objects, graphics) nor to the vocabularies of the model specification (e.g., agents, grid, movement, segregation rules). From a strict formal point of view the consistency between the abstract machines is incommensurable. From an empirical point of view, the relative consistency can be tested against the behavior of the program.

But even the empirical perspective does not seem to be able to provide any criterion to decide upon which embedded model should be used to describe both the behaviour of the program and the social phenomena (that such program presumably represents). This dilemma suggests that the logic of the method of ABSS highlights the presence of intentional aspects in programming and interaction with computers.

3.2 Definition of Intentional Capability of Programs

The intention of someone in implementing a program is to produce processes in a target machine, according to a certain specification, which should be meaningful for a group of people observing the machine. Presumably, the role of the observer is to idealize something that should be in accordance with the specification intended meanings. Whether the observer's idealizations can actually be represented by a theory, regarded as some form of abstraction, is not so clear. But even in the worst case, if the aim is to infer consequences from a specification, or establish additional premises thereon, then the execution of a program presumes the construction of a new theory that should disclose something more than the theory that was considered in the first place.

This is in line with the whole idea of computing: the belief that the execution of a program consists in manipulating representations, which give rise to yet other representations. Accordingly, insofar as new representations may be formed during or after program executions, we will use the term "representations *a posteriori*" so as to distinguish them from the program specification.

Using these terms, the arguments presented in the preceding sections can be reformulated. Among the models embedded in the target machine, there are no definite reasons to choose a specific set of representations *a posteriori* in terms of one model or another. If those representations are to be justified as valid formal consequences of the specification, they must be tested for empirical adequacy. Nevertheless, this depends on a fundamental condition: according to the classic theory of computation, both the specification and the representations must be formulated in a first-order language. Should this condition be granted, we could say – in a certain sense – that the execution of a program deduces representations *a posteriori* from its specification.

That being so, one way of looking upon specifications and representations *a posteriori* is to see them as describing laws, i.e. material conditions of necessity between events or properties about the behavior of programs, whose test for empirical adequacy is related to two tacit methodological conditions: Firstly, that the intended meanings of the specifications and representations, with reference to the behavior of the program in the target machine, be shared by the simulation implementer and the observers. Secondly, presumably, in the case of ABSS, that the intended meanings of the specifications and representations, with reference to the actual social phenomenon, be shared by the simulation observers. Two remarks should be made, nevertheless. The former condition is the only one relevant to regard simulation as an automated procedure of formal inference, whereas the latter is irrelevant to that effect. Consequently, that same condition is the only one relevant to regarding program verification within the scope of a logic of empirical adequacy.

The way to comply with these conditions can vary, however. Insofar as we have suggested that they are satisfied more or less tacitly, we should presume that the expressability of the specification language, as well as the expressability of the representations *a posteriori*, is also evaluated tacitly. But once we realize that almost all specifications and representations in ABSS are formulated in a rather informal way, there is no other alternative but to presume that the relevance of such structures must be established through explicit and verifiable methods. Unless the specifications and representations have been formulated in the formal language of the execution model, it is not appropriate to assume that any specification or any representation *a posteriori* can be translated, without loss of generality, to a first-order language.

Thus, for example, in Schelling's (1978) model of ethnic residential segregation, there should be a considerable consensus around a first-order language capable of expressing the specification and *a posteriori* representations disseminated in the literature, where such terms as "ethnicity", "segregation" or "tolerance" should convey the same meanings to the simulation implementer and to the community of observers. This may be achieved following one of two procedures: (i) explicitly, by showing that the specification and representations *a posteriori* can be, without loss of generality, expressed by a first order language, or (ii) implicitly, according to any validated methodology able to grant that effect.

The tendency in the literature is just the opposite, however. In the first place, the published articles remark that the meanings of specifications in relation to the target machine lose extensive generality to what is intended originally. In the second place, the published articles do not report any attempt to formulate representations *a posteriori* in a first-order language. This is sufficient to encumber the possibility of understanding the execution of a program as a process of formal inference that validates its results empirically. The acceptance of a social simulation by a community of observers depends on interpretative aspects that go beyond empirical adequacy, *for the semantic significance of computer programs conveys not only a causal capability, but also an intentional capability*.

By intentional capability we understand the following: (i) the recognition that since computation is a symbolic phenomenon, or representational, or semantical, it is intentional insofar as we assume that the behaviors of computers stand for other things in the world (Smith, 1996), (ii) the recognition that programs implemented in computers possess a causal capability that affects the behavior of computers, whereby the simulation implementer has the intention of submitting behaviors that stand for other things in the world for a community of observers, who may or may not accept those intended meanings, (iii) the recognition that the simulation implementer and the observers' intended meanings will remain intentional insofar as the propositions used for interpreting the observed behavior of programs are not verified empirically.

3.3 Examples of Intentional Capability of Programs

We argue that ABSS involves more outstanding intentional aspects than ordinary computer science. We illustrate three concrete examples from the work of Axelrod (1997) and Epstein & Axtell (1996). The examples will be used in order to show that program verification is supported in the published articles by means of persuasive descriptions, which are different in kind from the ones supporting program verification in ordinary computer science.

First Example: Axelrod (1997)

The immediate origin of the tribute model, as well as Axelrod's culture dissemination model (1997), is a concern for how nation-states form. Axelrod's interest was heightened by the demise of the Soviet Union and Yugoslavia (p.121). In the tribute model (pp.121-144), the intended meaning for each actor is a nation-state, having as fundamental characteristics its wealth and a list of neighbors. World geography is regarded as a unidimensional space arranged on a line, resulting in two constant neighbors for each nation. The model is described as follows (p.128, our italics):

"The basic units of the model are ten actors arranged on a line. The actors can *be thought of* as independent political units, such as nations… In each year, three actors are chosen one after another at random to become active…The selection of actors to be active is based upon the notion that ambitious leaders and potential disputes arise at random…"

The initial wealth of each actor is chosen from a "uniform distribution between 300 and 500." These parameters, like all the others in the model, are described by the author as "somewhat arbitrary and selected for convenience" (p.128). The basic ingredient of the model is based on the notion of "commitment." When wealthy nations threaten less wealthy nations with war, the latter are compelled to pay tribute to the former, increasing the levels of commitment between the nations. The simulation suggests that high levels of commitment encourage the formation of new political actors, alliances regarded as sets of nations that act jointly for the benefit of common interests.

Details about the implementation of the model are not described in the article. The notion of commitment seems to define a meaning only to the observers of the target machine. Somewhat tacitly, the observers must infuse specific meanings into the specific behaviors of the executing programs. Unlike soft artificial intelligence, it does not seem to be a goal of the author to show that the notion of commitment means anything for the executing programs themselves. The tribute model "…assumes that actors develop more or less strong commitments to each other based upon their prior actions. These *commitments can be thought of* as the result of psychological processes or the result of political rules of thumb." (p.127, our italics).

Summing up, the first goal of Axelrod is to suggest that his model is representative of the problem of the emergence of new political actors, even though he assumes its simplicity very openly. The second is to suggest that the behaviors of the executing programs may be thought of as specific actors and commitments, as well as the result of the emergence of new political actors.

Hence, it is insofar as the behavior of the executing programs should be thought of as an arena of commitments, alluding to other things in the world – and that it is not found necessary to presume that the notion of commitment actually means anything for the actors considered in the executing programs – that it becomes unnecessary to show that the executing programs are actually representative of that notion. It follows from here that the propositions formulated to interpret the behavior of the program executions, in terms of the notion of commitment, are not verified empirically. From this point of view, the program code does not prove relevant for the observer, but it is rather the intention underlying its implementation that prevails.

Some implementation details are given more explicitly in other models. The goal of the culture dissemination model (pp.148-177) is to analyze the phenomenon of social influence, and explain how local convergence can generate global polarization, for example, explaining the emergence of regions in the world that share identical cultural values. Actors are distributed among constant co-ordinates in a grid. The culture of each actor is a set of five numbers, which we will call a *quintet* of numbers. Each position in the quintet represents a cultural feature, which can be thought of as anything by the simulation observers, such as the color of a belt that is worn (p.154), a gastronomic or sexual appetite. Each cultural feature can take the values of ten integers, ten for each feature invariably. For example, an agent with the culture given by the quintet "23637" means that the first feature has value 2 and the fourth has value 3. Again, these values can be thought of as any cultural trait in the scope of any cultural feature, such as blue or pink for the colour of a belt.

The intended idea of social influence is that actors who have similar cultures should be likely to interact and become even more similar. In the actual program this is specified through a mechanism called *bit-flipping*, upon which the probability of interaction between two actors is set proportional to a measure of similarity between two quintets. Thus, at the point where the program specifies that two actors interact, a feature upon which its traits differ is selected and set equal to a same trait, resulting in two actors holding the same trait for the same feature.

Inasmuch as simplicity is openly assumed by the author, it becomes interesting to analyze the simplicity of model in comparison with the scientific literature that is used to describe it. For instance, it is usual to view culture as a system of symbols which depend on the many interconnections between the many traits that make up a culture, by which people confer significance on their own experience (p.152). According to Axelrod, his model has an advantage over others, insofar as its bit-flipping mechanism takes into account that the effect of one cultural feature depends on the presence or absence of other cultural features. Paradoxically, he states that "the emphasis is not on the content of a specific culture, but rather on the way in which any culture is likely to emerge and spread" (p.153).

It becomes clear that the influence mechanism in the model, as well as the dissemination and emergence of culture, does not depend on the experience of the actors as to the particular significance of the features and traits that make up their cultures. The observer of the simulation must, somehow arbitrarily, infuse the behavior of the executing programs with additional meanings, like the ones alluded to by Axelrod, such as "value adoption", "the color of a belt", "influence" and "culture."

Second Example: Epstein and Axtell (1996)

The semantic gap between specifications, programs and representations *a posteriori* is rather patent in Epstein and Axtell's sugarscape model (1996). The work analyses a series of phenomena involving concepts such as culture dissemination, racial segregation, friendship, sexual reproduction, epidemiology, and a variety of economic models. The goal is to "grow" histories – or proto-histories (p.8) – of artificial societies, so as to simulate the emergence of natural civilizations, by demonstrating formally and deductively that certain specifications are sufficient to generate the phenomena in which the researcher is interested.

One of the book's aims is to grow an entire history of an artificial civilization, where concepts like sex, culture and conflict are explored. The storyline is presented with the following text:

"In the beginning, a small population of agents is randomly scattered about a landscape. *Purposeful* individual behavior leads the most *capable* or *lucky* agents to the most fertile zones of the landscape: these *migrations produce spatially segregated* agent pools. Though *less fortunate* agents die on the wayside, for the *survivors* life is good: *food is plentiful, most live to ripe old ages* (…)" (p.8, our italics)

Our italics in the text serve the purpose of pointing out the semantic richness of some terms in the text, notwithstanding the descriptive richness of the whole storyline. However, the need to implement the model in the target machine implies decreasing the level of expressiveness from the storyline to the program specification, and from the specification to the program code, resulting in very simple rules. For example, each agent in the simulation is associated with a set of characteristics, such as fertility, visual acuity or gender. A typical rule in the model could be:

Agent sex rule (p.56):

– Select a neighbouring agent at random;
– If the neighbour is fertile and of the opposite sex and at least one of the agents has an empty neighbouring site (for the baby), then a child is born;
– Repeat for all neighbours.

These characteristics are specified in the program exclusively by bits in a binary word. For example, if the first bit in the binary word is equal to one, then the agent is male. The observer should somehow infuse the behaviour of the executing programs with the intended meanings of "female," specifically all agents that have the bit turned off.

An aim in Epstein and Axtell's research is to explain how transmission of culture can eventually produce spatially distinct tribes with different cultures. As in Axelrod's model, they use a bit-flipping mechanism. A culture is a binary string of bits that can take the values of either zero or one. From here it follows the observation of friendship networks (p.79): when an agent is born it has no friends, but agents who become neighbors and are close culturally are defined to be friends. Cultural closeness is measured by the Hamming distance, which is obtained by comparing the binary strings position-by-position and totaling the number of positions at which they are different.

In a small subscript, Epstein and Axtell write (p.79):

"We offer this definition of 'friendship' as a simple local rule that can be implemented efficiently, not as a faithful representation of current thinking about the basis for human friendship."

Nevertheless, by drawing connections between friends, Espstein and Axtell offer a set of graphical figures illustrating friendship networks in the simulation, and comparing them to socio-political patterns, such as connections between individual dissidents of repressive regimes (p.80). Again, the authors are the ones who lead the observer to represent the executing programs with such words as "friendship", "culture" or "sexual gender." This problem is explicitly raised by the authors, who ask at some point in the book (p.52): Had the rules that specify the agents' behavior not been described, would anyone be able to guess that the agents follow this or that rule? And their answer is:

"We do not think we would have been able to divine it. But that really is all that is happening."

If the question seems appropriate for us, it does seem that the answer is based on a subtle confusion. Let us consider Axelrod's culture dissemination model, where the executing programs are illustrated by a grid of ten-by-ten (10x10) quintets of integers, ranging from zero to nine, in constant variation according to the bit-flipping rule. Figure 2 illustrates what two iterations of the simulation could be, with a set of four-by-three (4x3) cultures.

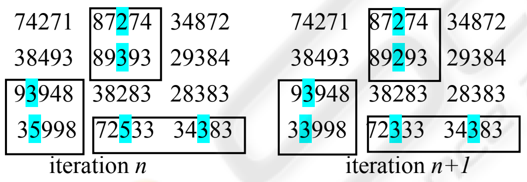

Figure 2: Agents interacting are marked with a square.

Epstein and Axtell's misunderstanding becomes clear here. In fact, it would be possible to find out that the agents follow this or that rule, for the implemented rule is very simple. By simple observation of the simulation, we would be able to formulate the rules that govern its behavior. It does seem to us, however, that the rules resulting from this empirical inquiry would not be composed by the vocabulary of the original ones. Instead of terms such as "culture" or "friendship," we would find sets of integers adorned by logical or mathematical operations or, possibly – in the case of Epstein and Axtell's model – by the names that stand for the colors of certain pixels or characters observed on the screen.

And we could say, all in all, "that really is all that is happening."

By all means, the intentional significance of the original rules surpasses the causal significance of the new rules, insofar as the interpretation of the original rules does not result from an empirical inquiry.

3.4 The Role of Intentional Capability of Programs

Whereas a model is built and analyzed on the basis of observation and experimentation, it may be considered a representation of reality. However, in ABSS, most representations *a posteriori* result from an experimental process, even though they do not need to represent contingent conditions of necessity between facts about the program behaviors.

The implementer's role, in this new context, seems to be significantly strengthened. His role is to foster interpretations that exceed the limited empirical expressiveness of the model, according to the opinion of a limited community of observers. Those interpretations should be in accordance with the intention that underlies the implementation, and only in that scope should they be acquired experimentally. Hence, apart from the empirical facts about the behavior of programs, the role of both the implementer and the observers is to negotiate and ascribe contingent conditions of intentionality to the simulation outcomes.

Summing up, there are two complementary scientific logics at stake in ABSS, one based on the formal and empirical logic of program verification, in which necessity conditions about the behavior of programs are specified and verified empirically, and another based on the experimental logic of program verification, in which intentionality conditions about the behavior of programs are specified and verified experimentally, albeit not empirically, according to a limited community of observers.

We shall establish a parallel between the roles of empirical and intentional verification of programs. The role of empirical verification is to exercise the construction of programs in order to achieve empirical adequacy between program executions and the causal meaning of those programs. The role of intentional verification is to exercise the construction of specifications and programs in order to achieve experimental adequacy between program executions and the intentional meaning of those programs, in the context of some limited community of observers.

The role of the community of observers, while acting freely, is to negotiate the intentional conditions meant by the implementer, as well as to reject, accept or interpret other conditions, according to both the behavior of the program executions and the social phenomena. Whereas the set of representations used in ABSS may be interpreted empirically or intentionally against the program executions, the conditions of intentionality are the ones that are liable to a doubly contingent interpretation.

For example, in Schelling's model, whether or not an observer is willing to describe the program behaviors with the term "segregation" depends on his inclination to consider aggregations of like-colored agents in the grid. The level of aggregation might be expressed as some qualitative or quantitative measure. However, insofar as the term "segregation" becomes interpreted according to the social phenomenon, the verification of the program execution behaviors becomes subjected to an intentional logic. For instance, the following proposition reveals essentially empirical contents:

"There is a critical value for parameter C [the minimum proportion of like-colored agents], such that if it is above this value the grid self-organizes into segregated areas of single color counters. This is lower than 0,5" (Edmonds, 2003, p.123).

And this leads Edmonds, with the social phenomena in mind, to conclude something that conveys a logic of intentional verification of programs, now liable to a doubly contingent interpretation:

"Even a desire for a *small proportion* of *racially similar neighbors* might lead to self-organized *segregation*" (p.123, our italics).

4 CONCLUSIONS

The experimental reference of agent-based social simulation (ABSS) becomes clear once we realize that the knowledge acquired from the simulations is an outcome of doubly contingent exercise that, in spite of not being empirical, is an outcome of an experimental exercise.

In any case, there is a question that prevails: what kind of credibility can each verification category ensure? With respect to intentional verification, the answer may become clearer once we realize that the existence of yet another kind of program verification results from the encounter of the formal and empirical logic of computer science with the multiplicity of methodologies in the social sciences, which cannot be dissociated from the multiparadigmatic logic of the interpretation of human social action – intentional verification is characterized by the acquisition of subjective elements from the programs.

Contrary to artificial intelligence, where the lack of expressiveness of formal models has been an obstacle to scaling up programs, this lack of expressiveness is instrumental to the method of ABSS. Hence, from an epistemological perspective our theory solves a semantic dilemma by deflating it: It releases the social simulation researcher from the semantic conflict between the formal perspective of computation and the informal or negotiated perspective of computation. However, the responsible use of a simulation suggests that not even the social simulation researcher should invoke his neutrality as to his own evaluation of the simulation results.

REFERENCES

Axelrod, R. (1997). *The Complexity of Cooperation – Agent-Based Models of Competition and Collaboration*. P.U. press.

David, Nuno; Marietto, Maria; Sichman, Jaime; Coelho, Helder (2004). "The Structure and Logic of Interdisciplinary Research in Agent-Based Social Simulation". *Journal of Artificial Societies and Social Simulation (JASSS)*, 7(3).

David, N.; Sichman, JS; Coelho, H. (2005). "The Logic of the Method of Agent-Based Simulation in the Social Sciences: Empirical and Intentional Adequacy of Computer Programs". *Journal of Artificial Societies and Social Simulation (JASSS)*, 8(2).

Edmonds, B. (2003). "Towards an Ideal Social Simulation Language". *Multi-Agent-Based Simulation II*, *LNAI*, v.2581, Springer-Verlag, pp.105-124.

Epstein, J.; Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*, MIT press.

Fetzer, J. (1988). "Program Verification: The Very Idea". *Communications of the ACM*, v.31, pp.1048-1063.

Fetzer, J. (1999). "The Role of Models in Computer Science". *The Monist*, 82(1), La Salle, pp. 20-36.

Schelling, T. (1978). *Micromotives and Macrobehavior*, W. W. Norton & Company.

Smith, C. (1996). "The Foundations of Computing". Smith's introduction to a series of books that report his study of computing in the books *The Age of Significance: Volumes I–VI*. Available at <http://www.ageosig.org/people/bcsmith/papers>.