Nuno David, Jaime Simão Sichman and Helder Coelho (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 vol. 8, no. 4
<http://jasss.soc.surrey.ac.uk/8/4/2.html>

For information about citing this article, click here

Received: 02-Oct-2005 Accepted: 02-Oct-2005 Published: 31-Oct-2005

Abstract

The classical theory of computation does not represent an adequate model of reality for simulation in the social sciences. The aim 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. We attempt to evaluate whether social simulation implies an additional perspective about the way one can understand the concepts of program and computation. We demonstrate that the logic of social simulation implies at least two distinct types of program verifications that reflect an epistemological distinction in the kind of knowledge one can have about programs. Computer programs seem to possess a causal capability (Fetzer, 1999) and an intentional capability that scientific theories seem not to possess. This distinction is associated with two types of program verification, which we call empirical and intentional verification. We demonstrate, by this means, that computational phenomena are also intentional phenomena, and that such is particularly manifest in agent-based social simulation. Ascertaining the credibility of results in social simulation requires a focus on the identification of a new category of knowledge we can have about computer programs. This knowledge should be considered an outcome of an experimental exercise, albeit not empirical, acquired within a context of limited consensus. The perspective of intentional computation seems to be the only one possible to reflect the multiparadigmatic character of social science in terms of agent-based computational social science. We contribute, additionally, to the clarification of several questions that are found in the methodological perspectives of the discipline, such as the computational nature, the logic of program scalability, and the multiparadigmatic character of agent-based simulation in the social sciences.

Keywords:
Computer and Social Sciences, Agent-Based Simulation, Intentional Computation, Program Verification, Intentional Verification, Scientific Knowledge

Introduction

1.1

Following the consolidation of the multiagent paradigm in computer science, the role of computer simulation has acquired a renewed importance in the social sciences. From an interdisciplinary perspective, the discipline of Agent-Based Social Simulation finds its origin in the intersection of the social and the computer sciences.[1] 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.

1.2

The difficulties in constructing and analysing simulations, even the most simplified, have been underlined in the literature. The obstacles found to increase the complexity of models, for example, in terms of the number of agents that constitute them, are not well understood. Nevertheless, for many researchers, simulation is seen as an alternative to other methods in the social sciences, which are unable to provide satisfactory support for certain topics of research.[2]

1.3

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.[3] Even so, the experimental character of simulation remains ambiguous, which raises some interesting questions around the kind of scientific knowledge that simulation is providing. On the one hand, the experimental reference of simulation appears unclear, 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 object 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 Turing-Church thesis[4], and the stakeholders' informal interpretations acquired through direct observation of simulations.

1.4

In effect, the discomfort in regard to the logic of simulation derives from the innumerable concepts that may be acquired from the social phenomena one intends to simulate, which leads to the elaboration of different constructions. But if on the basis of a constructivist approach there is always — even if somehow paradoxically — a reference to reality, how can we reconcile the experience and expectations of the simulation stakeholders with the practical, theoretical and intellectual aims that may result from it? 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?

1.5

The problem is the need for conciliating two scientific traditions within a computational methodology that should be of an experimental kind. A challenge for the research community, therefore, should be to focus on finding methodological perspectives that could reconcile the formal and empirical logic of program verification in computer science with the interpretative and subjective logic of the social sciences. This paper aims to present a theory of program verification for simulating social theory and phenomena, especially with reference to the epistemological basis, limits and particular kind of scientific credibility.

Scientific knowledge and computer programs

1.6

The credibility of scientific results of social simulation is related to the difficulty in putting in harmony two scientific traditions, possibly associated with two kinds of scientific knowledge. This difficulty suggests the elaboration of an alternative vision as to the role played by

computer programs in scientific knowledge, as well as to what it means to verify and validate programs. Insofar as the knowledge acquired through simulation must be recognised according to the interpretative character of the social sciences, we should attempt to evaluate whether simulation implies an additional perspective in the way that one can understand the concepts of program and computation.

1.7

The aim of this paper is to demonstrate that the logic of social simulation implies at least two distinct types of program verifications that reflect an epistemological distinction in the kind of knowledge one can have about programs. Computer programs have a semantical significance related to their causal capability (Fetzer, 1988) and intentional capability. This distinction is associated with two types of program verification, which we call empirical and intentional verification. We demonstrate, by this means, that computational phenomena are also intentional phenomena, and that such is particularly manifest in agent-based social simulation.

1.8

By demonstrating that it is the intentional verification of programs that is doubly contingent with both the behaviour of the programs and the social phenomena, we identify a new category of knowledge we can acquire about computer programs. This knowledge is acquired within a context of limited consensus that, in spite of not being empirical, is an outcome of an experimental exercise. We contribute, additionally, to the clarification of several problems found in the methodological perspectives of social simulation, such as the computational nature, the logic of program scalability, and the multiparadigmatic character of the discipline.

The structure of the paper

1.9

This paper is organised as follows. Section 2 introduces the background assumptions about computer science epistemology that prove relevant to our proposal. It is first recalled that, according to Fetzer's proposal (1999), computer programs seem to possess a causal capability that scientific theories do not seem to possess. Since the meaning of the terms "program verification" and "program validation" is not always consensual in computer science, it is subsequently analysed how those concepts may be understood in social science simulation. The role of Section 3 is showing that computer programs in social simulation seem to possess an intentional capability that surpass their causal capability. A set of examples that are found in literature are then used to identify two categories of program verification, as well as two kinds of knowledge we can have about programs. The final version of the logic of the method of agent-based simulation in the social sciences is presented in Section 4. Section 5 discusses the role of intentional computation in social scientific knowledge. We raise objections to the formal and deductivist perspective of computational social science, advocate the intentional perspective as the alternative, and present conclusions and further challenges.

The concepts of program verification and validation in social simulation

2.1

The epistemological status of social simulation should not be analysed without the contribution of the epistemology of computer science. It is important to know the kind of thing that computer programs are supposed to be, and how knowledge of program executions in computers might be acquired. The first thing that should be borne in mind is that the computer science meaning of the terms 'verification' and 'validation' is different from the meaning usually given in the social sciences. Nevertheless, both terms are used in social simulation, often with disparate meanings.

2.2

The role of this section is to introduce the assumptions about computer science epistemology that will prove relevant in analysing the intentional capability of computer programs. We first

recall that computer programs have a semantical significance related to their causal capability (Fetzer, 1988; 1999; 2001). We next analyse how the concepts of program verification and validation in computer science may be understood and reapplied in the social sciences.

Background: Causal capability of programs according to Fetzer (1988; 1999; 2001)

2.3

In computer science, the notion of scientific truth or validity has been related to an old debate, confronting researchers advocating the use of formal methods for verifying programs and those defending the use of empirical methods. During the Eighties, both sides of the debate were extreme, involving those who considered that program verification was reducible to pure mathematics, inspired by Hoare or Dijkstra[5], and those who considered it an activity within applied mathematics or empirical science. By the end of the Eighties, the debate became especially eloquent after James Fetzer had published an article with the somehow provocative title: '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 actually a branch of applied mathematics, ruled by empirical research.

2.4

Fetzer's argument consists of distinguishing programs as encodings of algorithms from the logical structures that they represent. Insofar as a program can be considered a particular encoding of an algorithm suitable for compilation and execution in a computer, that program may be qualified as a causal model of a logical structure that instantiates a particular algorithm. Thus algorithms, rather than programs, appear to be the appropriate candidates for analogies with pure mathematics, while programs bear comparison with applied mathematics.

2.5

The comparison advanced by Fetzer (1988;2001), which we reproduce in Table 1, suggests that the differences between theorems and programs may outweigh their similarities. Programs, like scientific theories, have semantical significance that mathematical proofs do not possess. The lines that make up a program, like the sentences that make up a theory, tend to stand for other things for the users of those programs and those theories. However, even scientific theories do not possess the causal capabilities of computer programs, which can affect the performance of those machines when they are loaded, compiled and executed.

Table 1. Proofs, theories and programs, according to Fetzer (1988;2001).

	Mathematical Proofs	Scientific Theories	Computer Programs
Syntactic Entities	yes	yes	yes
Semantic Significance	no	yes	yes
Causal Capability	no	no	yes

2.6

The causal capability of programs becomes crucial once we realise that, rather than one model, we use many models in the implementation of a single program (Fetzer, 1999). 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, a program can be thought of as a model of a potential solution of a problem, where the language functions as a model of an abstract machine, since many programs can be formulated in that language. Since high-level programming languages, such as Java or Prolog, stand for abstract virtual machines rather than physical machines, it becomes clear that the process of program implementation involves the

construction of a sequence of embedded models that are causally connected to a target physical machine.

2.7

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. The advantage of programming with 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. In fact, 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. Typically, the lowest-level language programmers use is Assembly language, where there is more or less a one-to-one correspondence between commands and operations.

2.8

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.[6]

2.9

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 causal relation. In Figure 1 we reproduce Fetzer's diagram that illustrates this notion — for more details see Fetzer (1999, p.33). 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.

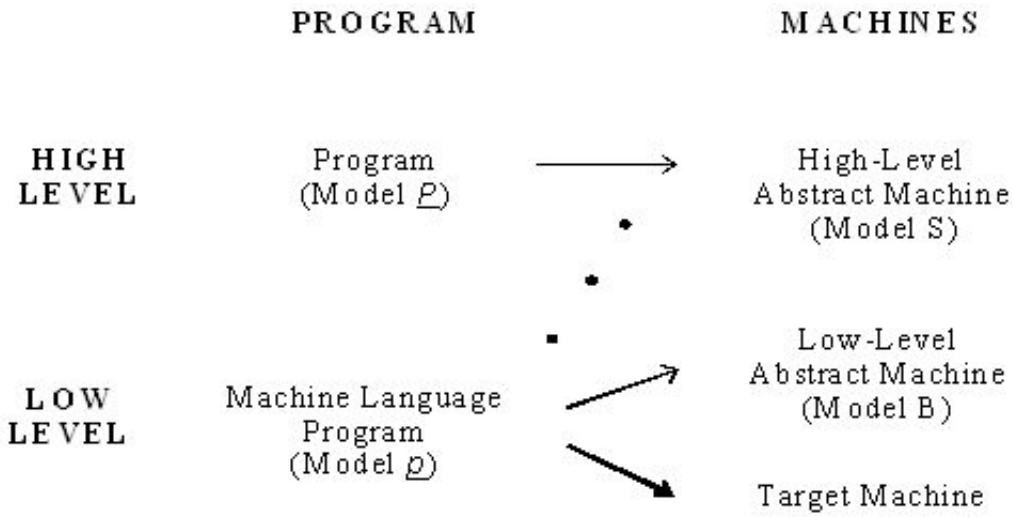

Figure 1. Programs and languages as models, according to Fetzer (1999). The series of three dots stands for the possible existence of compilers and interpreters that effect some causal connection between programs/machines at different levels.

2.10

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 realising 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.

Computer programs as the experimental reference of social simulation

2.11

A question remaining in the epistemology of social simulation consists of characterising what a scientific experiment is. 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 of 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 *program verification* is to ascertain the validity of certain output as a function of given input, regardless of any interpretation given in terms of any theory or any phenomenon not strictly computational. The execution of a program is understood, in this sense, as a calculus of formal inference, which manipulates symbols without regard to their content.

2.12

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.

2.13

One way to understand the roles of verification and validation is to realise that theories in social simulation are doubly contingent, interpreted according to two distinct phenomena. Consider Schelling's model of ethnic residential segregation (1978). This model specifies a set of rules for the movement of agents with different colours in a grid. Depending on different input parameters, the simulation produces diverse patterns of agents with the same colour. A theory of the simulation dynamics may be pondered in line with various interpretative references, namely:

- the model specification, including the rules for the movement of agents;
- the program execution behaviours, such as the observed patterns of colour clusters on the screen;
- the actual phenomenon of ethnical residential segregation, or a theory thereof.

2.14

The specification, however, expresses a theory, which must be ultimately stated in terms of computer programs. Figure 2 illustrates the doubly contingent character of theories in simulation.

2.15

Initially, there is a theory (T0) that represents the social phenomenon, which should be expressed in terms of certain specifications and programs. Programs are also symbolic representations which will be tentatively implemented in a target machine with the help of other programs already executing in the computer, such as a compiler or a simulation environment. The simulation happens by executing the program in a controlled way, once again with the help of other programs executing in the computer, such as simulation environment. Thus, the implementation of a simulation implies an interaction between symbolic and physical processes that implement a causal relation between theory T0 and the simulation behaviour. The goal of the whole exercise is to construct a theory T1 that, although not contradictory to T0, expresses something that T0 does not.

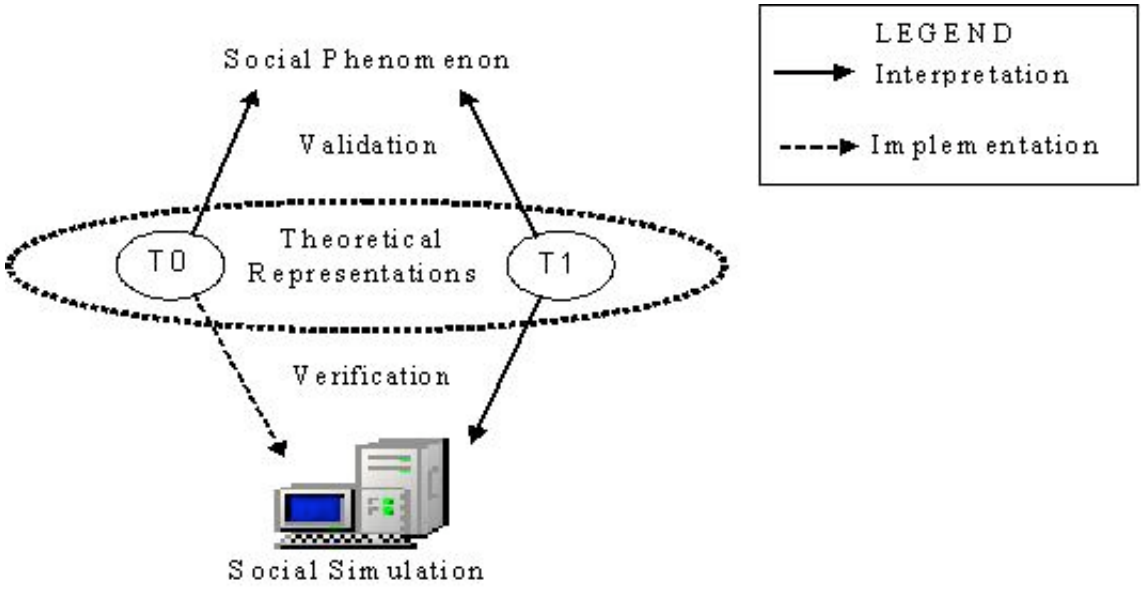

Figure 2. The doubly contingent character of theories in simulation.

2.16

Indeed, unlike theory T0, which will always be a theory of the actual social phenomenon, theory T1 may be interpreted according to the observed characteristics of a set of animated icons on the screen, or the observed characteristics of an actual social phenomenon. However, whereas somehow arbitrarily one may be able to find a common syntactic basis from which to interpret theory T1 in two different ways, this exercise becomes more complex from a technical point of view, for there are innumerable ways to idealize the observed behaviour of a program executing in a computer.

2.17

As we have mentioned, the implementation of a program involves a sequence of models embedded in a target machine. Each one of such models can suggest an alternative interpretation for describing and verifying the behaviour 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). The parts that those vocabularies designate in the world are not even the same; from a strict formal point of view the consistency between the abstract machines is incommensurable.

2.18

Fortunately, from an empirical point of view, the relative consistency between the two abstract machines can be tested against the behaviour 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 lend itself to a doubly contingent interpretation — according to both the observed behaviour of the program and the actual social phenomenon.

2.19

In social simulation, there are no empirical justifications to interpret the observed behaviour of the program in terms of a JAVA abstract machine, instead of the terms of an Assembly abstract machine, or the more subjective terms of the simulation stakeholders. A dilemma which suggests that the logic of the method of agent-based social simulation highlights the presence of intentional aspects in programming and interaction with computers.

Intentional capability of programs

3.1

The role of this section it to show that social simulation programs possess an intentional

capability that surpasses their causal capability. The argument is organised into four parts. We first define the meaning of intentional capability of programs. We then show that computation in social simulation involves more outstanding intentional aspects than does computer science. Subsequently, we concretise our argument with canonical examples of simulations found in literature. Finally, we clarify the role of intentional verification of programs in social simulation, by confronting the philosophical notions of contingent condition of necessity and of contingent condition of intentionality.

Definition of intentional capability of programs

3.2

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 idealise something that should be accordant with the specification intended meanings. Whether the observer's descriptions can actually be regarded as a theory, regarded as an arbitrary form of representation and in accordance with the modeller's intentions, is not so clear. But even in the worst case, if the aim is to infer consequences from a specification, or to 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.

3.3

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*' to distinguish them from the program specification.

3.4

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. 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.[7] 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.

3.5

That being so, one way of looking upon specifications and representations *a posteriori* is to see them as describing laws, permanent properties or conditions of necessity between events or properties about the behaviour of programs. Given a specification O=U(I), if a physical computer has input I and functions properly, then it is reasonable to assume that the output O will be the same in all occasions when given that same input I. Laws in this sense harmonize with the contemporary conception of laws of nature in philosophy of science, contingent dispositions that something cannot lose without losing a corresponding reference property. Theoretically, this holds even when the computation involves 'stochastic' features and/or complex behaviours, since on a computer these are produced by deterministic pseudo-random generators.

3.6

That being the case, a 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 behaviour of the program in the target machine, be shared by the simulation implementer and observers. Secondly, presumably, in the case of social simulation, 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.

3.7

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*, are also evaluated tacitly. But once we realize that almost all specifications and representations in social simulation 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.

3.8

Thus, for example, in Schelling's model of ethnic residential segregation (1978), 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, P1 or P2: (P1) explicitly, by showing that the specification and representations *a posteriori* can be, without loss of generality, expressed by a first order language, or (P2) implicitly, according to any validated methodology able to grant that effect. It would be under these conditions that the following conclusion could be considered a valid conclusion of a simulation program that had been verified empirically:

> 'The Schelling model assumes that a family will move only if more than one third of its immediate neighbours are of a different type (e.g., race or ethnicity). The result is that very segregated neighbourhoods form even though everyone is initially placed at random, and everyone is somewhat tolerant.' (Axelrod, 1997a, p.24)

3.9

The tendency in the literature is just the opposite, however. None of the referred procedures is performed along the conventional development of social simulations. 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 a sufficient condition 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.

3.10

By intentional capability we understand the following:

1. the recognition that since computation is in one way or another a symbolic phenomenon, or representational, or information-based, or semantical, it is intentional insofar as we assume that the behaviours of computers stand for other things in the world (see, e.g. Smith, 1996);[8]
2. the recognition that programs implemented in computers possess a causal capability that affects the behaviour of computers, whereby the simulation implementer has the intention of submitting behaviours that stand for other things in the world for a community of

observers, who may or may not accept those intended meanings;
3. the recognition that the simulation implementer and the observers' intended meanings will remain intentional insofar as the propositions used for interpreting the observed behaviour of programs do not represent conditions of necessity between events or properties about the behaviour of programs, and thus are not verified empirically.

Beyond constructivism in computer science: inadequacy of the classical theory of computation

3.11

The attempts to recognize that computation is an intentional phenomenon are not new. As Smith (1996) has mentioned, the classical theory of computation is characterised in an abstract form, focusing on what can be done formally and how hard is it to do it. This recalcitrant notion, derivative from formal logic and metamathematics, sustains the idea of a machine manipulating symbolic or meaningful expressions without regard to their interpretation or semantic content. Notwithstanding, the debate in computer science has not been indifferent to the difficulty that the classic theory of computation finds in explaining how computer technologies adapt so quickly, and expand and exceed our best expectations. The existence of different readings about the theory, which do not seem to stand for conceptual synonyms when submitted to detailed analyses, has been contributing to increasing critiques as to its adequacy. The critique about the classic conception is essentially:

> …the often pretheoretical assumption that computers are formal. In one way or another, just about everyone thinks that computers are formal — that they manipulate symbols formally, that programs specify formal procedures, that data structures are a kind of formalism, that computational phenomena are uniquely suited for analysis by formal methods (…) this antisemantic thesis is contradictory to the tacit recognition that computation, in one form or another, is a symbolic or representational or information-based or semantical phenomenon — in other words, an intentional phenomenon. Though in ways we do not understand, the states of a computer can model, or simulate, or represent, or stand for, or carry information about, or signify other states in the world.

3.12

Smith's contention in defence of an intentional thesis is suited to the case of computer science. Let us consider a classical computer application, such as the operating system Microsoft Windows. Both the negotiation with the application developers, and the experimentation with the application itself, are crucial for helping end-users evaluate the meaning and functionality of a Microsoft Windows folder. Along the software production process, the comparison between an ordinary folder and a computer folder departs progressively from what could have initially been regarded as a simple metaphor. The behaviours of the target machine acquire new ontological status among large groups of end-users, becoming actual objects in the world with lawful properties. Computation, in this sense, is intentional insofar that it is constructivist, insofar as its behaviours can be evaluated empirically as observed properties of new objects in the world.

3.13

We argue, nevertheless, that social simulation involves more outstanding intentional aspects than ordinary computer science. Once we recognize that the typical interactive profile of a simulation observer is considerably more passive than the profile of a computer application end-user, the intentional character of simulation emerges distinctly.

Examples of intentional capability of programs

3.14

The goal is to show de facto that simulation programs possess an intentional capability beyond their causal capability. Let us consider the interactive profile of someone observing a social simulation, who appears to be far more passive than a computer application end-user. We maintain that this reflects yet another kind of intentionalily in computation. We next illustrate three concrete examples from the work of Axelrod (1997b) and Epstein and Axtell (1996). The examples are used in order to show that program verification in social simulation 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. This happens whether programs are described as textual representations in programming languages or supplied as execution processes in target machines.

3.15

Firstly, as we will show next, the intentional character is reflected by the textual and verbal descriptions used for persuading the observer that the behaviours of the target machine stand for other things in the world. Secondly, in general, those behaviours cannot acquire an ontological status, unless when they are submitted to a limited community of observers. Moreover, whereas familiarity with the program code is irrelevant for interacting with an ordinary application, its meaning in social simulation is, in some cases, described persuasively and explicitly. But there will be no attempts to show that the program has a level of expressability comparable with the intended model, or that the representations *a posteriori* stand for conditions of necessity that relate events or properties about the behaviour of program executions.

3.16

Accordingly, a distinction should be made between the intentional capability of programs in ordinary computer science and the intentional capability of social simulation programs. Although both programs are verified experimentally, in the case of social simulation the experimental vinculum does not remain sustainable without explicit and persuasive descriptions in the published articles.

First Example: The Culture Dissemination and the Tribute models of Axelrod

3.17

The immediate origin of the tribute model, as well as Axelrod's culture dissemination model (1997b, pp.121-144, pp.148-177), is a concern for how nation-states form. Axelrod's interest was heightened by the demise of the Soviet Union and Yugoslavia (1997b, p.121). In the tribute model, the intended meaning for each actor is a nation-state, having as fundamental characteristics its wealth and a list of neighbours. World geography is regarded as a unidimensional space arranged on a line, resulting in two constant neighbours for each nation. According to Axelrod (1997b, p.128), the reason for having so few actors and such a simple geography as a line is to keep the data analysis as simple as possible. The model is described as follows:

> '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... The basic cycle of the model is called a year. 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...If B fights rather than pays...' (1997b, p.128, our italics)

3.18

The initial wealth of each actor is chosen from a 'uniform distribution between 300 and 500' (1997b, p.128). These parameters, like all the others in the model, are described by the author as 'somewhat arbitrary and selected for convenience' (1997b, p.128). The basic ingredient of the model is based on the notion of 'commitment.' According to the intended model, when

wealthy nations threaten less wealthy nations with war, the latter are compelled to pay tribute to the former, increasing, nevertheless, 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.

3.19

Details about the implementation of the model are not described in the article.[9] 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 behaviours of the executing programs. 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. Thus, 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.' (1997b, p.127, our italics).

3.20

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 behaviours 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.

3.21

Hence, insofar as the behaviour 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 — 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 behaviour 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.

3.22

Some implementation details are given more explicitly in other models. The goal of the culture dissemination model (1997b, pp.148-177) is to analyse the phenomenon of social influence, and explain how local convergence can generate global polarisation, for example, explain the emergence of stable and homogeneous 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 colour of a belt that is worn (1997b, p.154), a gastronomic or sexual appetite. Each cultural feature can take the values of ten integers, ten for each feature, invariably.[10] 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.

3.23

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[11], 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 as equal to a same trait, resulting in two actors holding the same trait for the same feature.

3.24

Inasmuch as simplicity is assumed openly by the author, it becomes interesting to analyse the simplicity of the 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 (1997b, p.152). According to Axelrod, his model has an advantage over others, insofar that its bit-flipping mechanism takes into account that the effect of one cultural feature depends on the presence or absence of other cultural features (1997b, p.153). 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' (1997b, p.153).

3.25

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 behaviour of the executing programs with additional meanings, like the ones alluded to by Axelrod, such as 'value adoption', 'the colour of a belt', 'influence', 'experience' and 'culture.'

Second example: The Sugarscape model of Epstein and Axtell

3.26

The semantic gap between specifications, programs and representations *a posteriori* is rather patent in Esptein 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, wealth distribution and a variety of economic models. According to the authors, the broad aim of the book is to begin the development of a more unified social science, towards a generative social science:

> 'The broad aim of this research is to begin the development of a more unified social science, one that embeds evolutionary processes in a computational environment that simulates demographics, the transmission of culture, conflict, economics, disease, the emergence of groups, the emergence of groups, and agent coadaptation with an environment, all from the bottom up.' (1996, p.19)

3.27

The goal is to 'grow' histories — or proto-histories (1996, p.8) — of artificial societies, so as to simulate the emergence of natural civilisations, 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 behaviour 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, populations expand through *sexual reproduction*, and the transmission of *cultural* attributes eventually produces spatially distinct '*tribes*.' But their *splendid* isolation proves *unsustainable*…' (1996, p. 8, our italics)

3.28

Our italics in the text serve the purpose of pointing out the semantic richness of some terms used 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:

3.29

Agent sex rule (1996, 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.

3.30

These characteristics are specified in the program exclusively by bits in a binary word (one of two values for each bit, zero or one). For example, if the first bit in the binary word is equal to one, then the agent is male, and female otherwise. Once again, 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. The spatial distribution of agents is specified in a fix, two-dimensional, grid of fifty by fifty (50x50) co-ordinates, upon which the agents are limited to move in eight possible directions, like a queen in a chess game. Visual acuity is specified in four directions — north, south, east and west.

3.31

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 sequence of bits that can take the value of either zero or one. From here it follows the observation of friendship networks (1996, p.79) — when an agent is born it has no friends, but agents who at some point are neighbours and are close culturally are defined to be friends. Cultural closeness is measured by the Hamming distance[12]: Roughly, two agents are culturally close if they share a majority of ones or zeros, position by position, in the binary word.

3.32

In a small subscript, Epstein and Axtell write (1996, p.79):

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

3.33

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 sociopolitical patterns, such as connections between individual dissidents of repressive regimes (1996, p.80). Again, the author is the one who leads the observer to represent the executing programs with such words as 'friendship', 'culture' or 'sexual gender.' This problem is somehow explicitly raised by the authors, who ask at some point in the book: Had the rules that specify the agents' behaviour 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.' (1996, p.52)

3.34

If the question seems appropriate for us, it does also 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 3 illustrates what two iterations of the simulation could be, with a set of five-by-five (5x5) cultures.

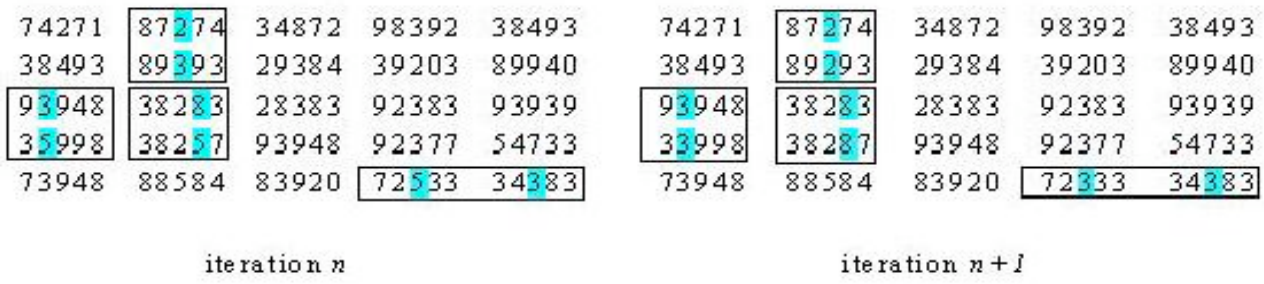

Figure 3. A possible scenario in the culture dissemination model of Axelrod (1997b). Agents interacting at iteration n are marked with a square. Marked numbers are the cultural features that interact.

3.35

The point we are trying to show 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 behaviour, and it would not even be necessary to have an experienced programmer available in order to implement the same simulation with the new rules. It does seem to us, however, that the rules resulting from this empirical inquiry would not be composed by the vocabulary of the original rules. 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 colours of certain pixels or characters observed on the screen.

3.36

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

3.37

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 enquiry — for computer programs seem to possess an intentional capability that surpasses their causal capability.

The role of intentional verification of programs in social simulation

3.38

Whereas a model is built and analysed on the basis of observation and experimentation, it may be considered a representation of reality, notwithstanding that it might not be considered the specific reality that it represents. However, in social simulation, most representations *a posteriori* result from an experimental process, even though they do not represent conditions of necessity between facts about the program behaviours. Computational models in social simulation are built on the basis of something more than observation and experimentation. One does not try to put away the perceptive-emotive data related to elements of subjective analysis. One may even act as if artificial agents themselves were endowed with value, significance, or purpose. The role of intentional verification of programs in social simulation is to find contingent conditions of intentionality in the programs, according to a limited community of observers.

3.39

The implementer's role, in this new context, seems to be significantly strengthened; to the extent that the implementer fosters interpretations that cannot be empirically justified upon the program behaviours, his role is to foster interpretations that exceed the limited empirical expressiveness of the computational 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 behaviour of programs, the role of both the implementer and the observers is to negotiate and ascribe contingent conditions of intentionality to the programs

and simulation outcomes.

3.40

Summing up, there are two complementary scientific logics that play a role in social simulation, one based on the formal and empirical logic of program verification, in which necessary conditions about the behaviour of programs are specified and verified empirically, and another based on the experimental logic of program verification, in which intentionality conditions about the behaviour of programs are specified and verified experimentally, albeit not empirically, according to a limited community of observers. This scenario would seem to be unavoidable, for it seems to be the result of the encounter between the formal and empirical logic of program verification in computer science with the interpretative logic of the social and human sciences.

Table 2. Scientific theories and computer programs - an extension to Fetzer's (1988; 2001) description.

	Mathematical Proofs	Scientific Theories	Computer Programs
Syntactic Entities	yes	yes	yes
Semantic Significance	no	yes	yes
Causal Capability	no	no	yes
Intentional Capability	no	no	yes

3.41

We mentioned previously that programs possess a causal capability that affects the performance of target machines when they are loaded, compiled and executed. Whereas a causal connection also certainly exists in social simulation, it is now possible to identify yet another kind of connection.

3.42

Whereas the role of compilers and/or interpreters is to grant a causal connection between programs and the target machine, the role of the implementer's persuasive descriptions is to grant an intentional connection between programs and the target machine. There are no fewer reasons for verifying the causal effects of programs — in the context of an empirical methodology — than reasons for verifying its intentional effects in social simulation — in the context of an experimental methodology. For, the experimental reference is neither the encoding of algorithm structures nor the objective effects that result from its causal capability, but the subjective effects that result from its intentional capability, according to a limited community of observers.

3.43

As it is all too well known, the theoretical-methodological context of the social sciences is multiparadigmatic. Table 2 seems to be the only possible way to reflect the multiparadigmatic character of social science into social simulation. Once we recall that the theoretical and methodological diversity of social sciences agglutinates relatively few consensuses, the appearance of multiple and limited consensuses as to the meaning of each simulation becomes unavoidable.

3.44

This permits us to explain limitations in scaleability of social simulations. The scaleability — the successive reuse and composition of relatively unorganised programs that generate implementations of increasing complexity, while not hindering the capacity for verifying its behaviours (e.g. an operating system, a word processor) — does not coexist easily with the

intentional character of simulation. Insofar as the function of a program is to be acquired in the scope of limited consensus, the successive composition with other more complex programs becomes increasingly more limited.

3.45

Indeed, it seems a mistake to imagine that simulation may integrate the archipelago of the social sciences, at least as far as that may depend on the establishing of wide consensuses, like those found in the natural and computer sciences.

The logic of the method of agent-based simulation in the social sciences

4.1

The relationship of social simulation to nuclear epistemological concepts becomes clearer once we understand the role of implementation in the process of simulation. We have now reached the conditions for presenting the final version for the logic of the method of agent-based simulation in the social sciences.

4.2

Two aspects need to be considered: (i) the existence of a causal capability in programs, which permits the verification of contingent conditions of necessity upon the behaviour of program executions in computers, and (ii) the existence of an intentional capability in programs, which permits the verification of contingent conditions of intentionality upon those executions.

4.3

This logic is illustrated in Figure 4. We combine the process of program implementation with the modelling of the social phenomena and the modelling of the program executions. The modelling of the program execution is illustrated in the left square (verification square). The modelling of the social phenomenon is illustrated in the right square (validation square). The dashed line in the middle denotes representations, such as specifications, high level programs, low level programs and representations *a posteriori*.

4.4

Figure 4 seems to be the only possible way to reflect the multiparadigmatic character of social science into simulation. The series of black dots in the implementation process stands for the existence of causal connections between programs and the target machine, by means of simulation platforms, compilers and interpreters. The series of unfilled dots stands for the existence of intentional connections between specifications and the target machine, as well as between programs and the target machine, by means of the implementer's persuasive descriptions. Finally, whereas the set of representations may be interpreted empirically or intentionally against the program executions, the conditions of intentionality in the lower part of the figure are the ones that are liable to a doubly contingent interpretation.

4.5

We shall establish a parallel between the roles of empirical verification of programs 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, always in the context of some limited community of observers.

4.6

Hence, as a consequence of the causal capability of a program, empirical verification tries to find out contingent conditions of necessity in the program execution behaviours — presumably, according to some epistemic conception of how those conditions are subjected to relevant scientific tests. Those conceptions usually take the form of deductive, inductive or abductive clichés.

4.7

Conversely, as a consequence of the intentional capability of a program, intentional verification tries to find out contingent conditions of intentionality in the program execution behaviours. 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 behaviour of the program executions and the social phenomena.

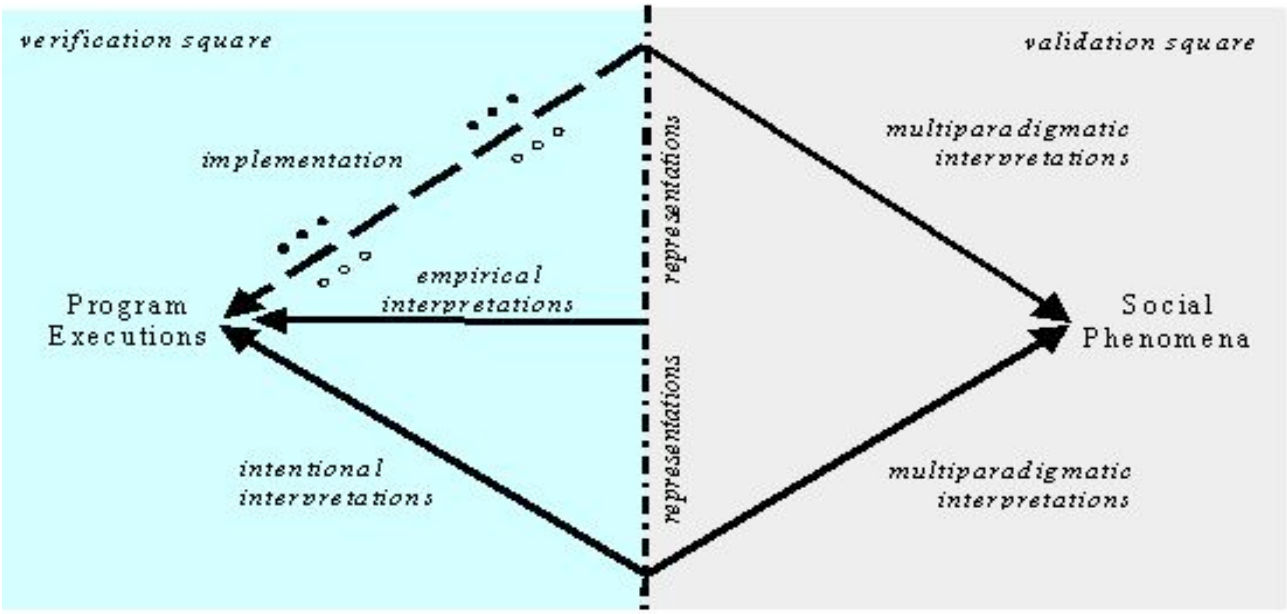

Figure 4. The logic of the method of social simulation.

4.8

For example, in the Schelling model, whether or not an observer is willing to describe the program behaviours with the term 'segregation' depends on his inclination to consider aggregations of like-coloured agents in the grid. The level of aggregation might be expressed as some qualitative or quantitative measure, more or less systematised, for example, according to some statistical measure. However, insofar as the term 'segregation' becomes interpreted according to the social phenomenon, the verification of the program execution behaviours becomes subjected to an intentional logic. For instance, the following proposition reveals essentially empirical contents, the observation and test of contingent conditions of necessity in the behaviour of the program:

> 'There is a critical value for parameter C [the minimum proportion of like-coloured agents], such that if it is above this value the grid self-organises into segregated areas of single colour counters. This is lower than 0.5' (Edmonds, 2003, p.123).

4.9

And this leads Edmonds, with the social phenomena in mind, to conclude something that conveys a logic of intentional verification of programs: the observation and test of contingent conditions of intentionality, liable to a doubly contingent interpretation, according to a limited community of observers:

> 'Even a desire for a small proportion of racially similar neighbours might lead to self-organised segregation' (p.123, our italics).

Conclusions

5.1

The logic of social simulation involves an additional category of program verification in computer science, whereon we observe the existence of three kinds of program verifications,

reflecting an epistemological distinction in the kind of knowledge one can have about programs: formal, empirical and intentional.

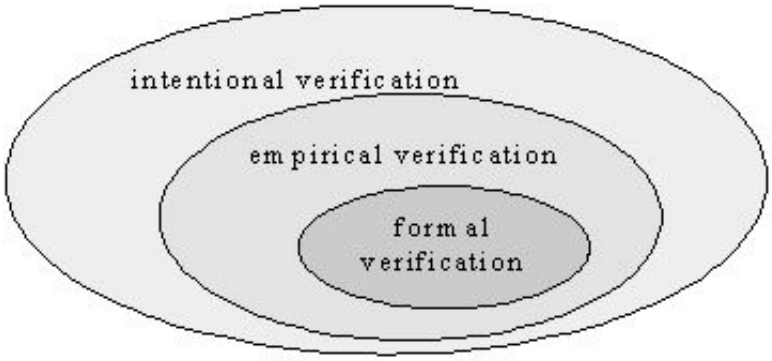

Figure 5. Three kinds of program verification and knowledge in social simulation.

5.2

In all cases, there is still a question that prevails: what kind of credibility can each verification category ensure? In computer science, the limits as to the role played by formal verification reached a significant level of consensus in the late Nineties. Likewise, with respect to intentional verification, the answer will partly result from the confrontation between the researchers' pragmatic, theoretic and intellectual aims.

5.3

A further clue can be found by realising 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 logic of the interpretation of human social action — intentional verification is characterised by the acquisition of subjective elements from the programs. This appears to be the context in which the relationship of simulation with the concept of scientific truth or adequacy must be found.

5.4

The question that would seem to arise is how to release the researcher from the semantic conflict between the formal perspective of computation and the stakeholders' informal descriptions, both of which are used to interpret the behaviour of simulations. Yet, whereas in computer science or artificial intelligence the lack of expressiveness of formal computational models has been understood as an obstacle to scaling up programs, it should be understood as instrumental to the method of simulation in the social sciences.

5.5

The semantic gap between the formal and the informal perspective of computation has been somewhat ignored by the research community. The apparent assumption is that both formal and informal representations can be legitimised deductively, insofar as the process of program execution can be understood as a calculus of deductive inference. This perspective can be found in Epstein and Axtell (1996), among many others. Epstein (1999), in particular, analyses it under the designation of generative social science. According to that line of reasoning, even if the specification of a simulation cannot be considered as a set of necessary conditions for deducing certain simulation results, it ought to be considered as a set of sufficient conditions. Yet, it is possible to show that this presumption is based on a logical confusion. Only in the context of some limited community of observers can a specification or a program be considered as a set of sufficient conditions to explain the behaviour of a simulation.

5.6

The solution to this problem is to devise a non-formal conception of computation in social simulation, such as the one we have proposed. The main aspect of our analysis is to contrast the empirical and the intentional program verification logics. Particularly, we were able to demonstrate that it is the intentional verification of programs that is contingent with both the

behaviour of the programs and the social phenomena. Moreover, whereas it is certainly possible to empirically legitimise certain formal structures for representing the execution of a program — such as the programming languages — we have shown that the informal structures are the ones that are usually subjected to a logic of intentional verification. This does not mean that the formal structures can not be considered representative of some social phenomenon according to a logic of intentional verification. But in any case, we should insist that the empirical verification of a representation in terms of a structure does not legitimise the intentional verification of another representation given in terms of that same structure.

5.7

From here it follows that the generative social science perspective is based on a misunderstanding of a logical nature. Insofar as the verification of a representation is of an intentional kind, but not of an empirical kind, that representation must not be considered as deducible from the specification of the simulation. Only in the context of some limited community of observers can a specification and a program be considered a set of sufficient conditions to explain the behaviour of a simulation. The conditions for the success of a social simulation depend on the particular theoretical-methodological context of the social scientist, can be interpretative and subjective, and may depend on socioeconomic and sociocultural contexts. Consequently, the adoption of a stakeholder participative approach seems fundamental. Assertions like Goldspinks's (2002, paragraph 1.2) are therefore inevitable. As they are generally in the social sciences — multiparadigmatic, with diverse theoretical and methodological contexts:

> '...researchers who adopt this 'third discipline' often report that they have experienced difficulty in having research results accepted within traditional peer reviewed journals.'

5.8

However, it is important to realise that this problem may become less contentious if the logic of simulation renounces its major bias, that is, the one which confers to it, as a consequence of its experimental character, a superior scientific objectivity in comparison with other methodologies in the social sciences. In that case, the social sciences could play an important role in computer science: To determine that there is no computation without intention. If in computer science there is no computation without representation (e.g. Smith, 1995), and as such there is no computation without interpretation (e.g. Fetzer, 1999), in social simulation there is no computation without intention.

5.9

But, in the end, the best slogan might be that there is no program execution without intention. Contrary to the logic of empirical verification, where we find the presumable neutrality of the implementer, the social sciences have created, by means of a multiparadigmatic logic, a new methodological conception in computer science, where the implementer plays a decisive role. We are convinced that there will not be consensual methods in social simulation. Agent-Based Social Simulation, like the social sciences, is multiparadigmatic.

Acknowledgements

We would like to thank the anonymous reviewers of this paper. Nuno David was partially supported by FCT/PRAXIS XXI, Portugal, grant number BD/21595/99. Jaime Simão Sichman is partially supported by CNPq, Brazil, grants 482019/2004-2 and 304605/2004-2.

Notes

[1] For a historical perspective, see Troitzsch (1997). For a recent study on the interdisciplinary

structure of the research community, see David et al. (2004).

[2] There are several perspectives, which sometimes seem to rely on contradictory positions. See, among others, Axelrod (1997a), Byrne (1997), Conte et al. (1997), David et al. (2004), David et al. (2005), Epstein (1999), Epstein and Axtell (1996), Gerhenson (2002), Gilbert (1995), Gilbert and Troitzsch (1999), Goldspink (2002), Gross and Strand (2000), Kluver (2003), Troitzsch (1997). See also subscript number 3.

[3] See, specifically, Kluver et al. (2003, paragraph 5.5). For an epistemological perspective based on the classical theory of computation, *a la* Turing-Church, see Epstein (1997). Conversely, this article attempts to present an alternative — and possibly incompatible — perspective.

[4] The classical theory of computation is also known as the Turing-Church thesis. See also subscript number 7.

[5] See, e.g. Dijkstra (1976).

[6] As Fetzer remarks (1999), high-level languages may not even be related to any physical machine, when appropriate compilers and interpreters are not provided. For example, the role of the programming language presented in Dijkstra (1976) is to reason about programs without even mentioning their behaviour on real computers. Compiling details are left unstudied.

[7] If we accept Turing-Church thesis, according to the classical theory of computation, all computation that terminates can be simulated by a first-order language. The understanding of the computational process according to higher-order logics finds apparent insuperable obstacles (see e.g. Papadimitriou, 1994).

[8] This conception does not necessarily suggest or imply the conception of artificial intelligence as to the use of the intentional stance for modelling artificial agents. Those are relatively independent issues.

[9] The code is made accessible on the Web for consultation.

[10] For example, an agent with the culture given by the quintet 23637 means that the first feature has value 2, the second attribute has value 3 and the fifth attribute has value 7.

[11] Epstein and Axtell (1996,p.73) call it *tag-flipping*.

[12] The Hamming distance between two binary strings is obtained by comparing the strings position-by-position and totalling the number of positions at which they are different.

References

AXELROD R (1997a) Advancing the Art of Simulation in the Social Sciences. In *Simulating Social Phenomena*, *Lecture Notes in Economics and Mathematical Systems*, v.456, Springer-Verlag, pp. 21-40.

AXELROD R (1997b). *The Complexity of Cooperation — Agent-Based Models of Competition and Collaboration*. Princeton University Press.

BYRNE D (1997). Simulation - A Way Forward? *Sociological Research Online,* v.2, n.2, http://www.socresonline.org.uk/socresonline/2/2/4.html.

CONTE R, Hegselmann R and Terna P (1997). Social Simulation: A new disciplinary synthesis. In *Simulating Social Phenomena*, *Lecture Notes in Economics and Mathematical Systems*, v.456,

Springer-Verlag, pp. 1-17.

DAVID N, Marietto MB, Sichman JS and Coelho H (2004). The Structure and Logic of Interdisciplinary Research in Agent-Based Social Simulation. *Journal of Artificial Societies and Social Simulation (JASSS)*, v.7, n.3, http://www.soc.surrey.ac.uk/JASSS/7/3/4.html.

DAVID N, Sichman JS and Coelho H (2005). The Role of Intentional Decision-Making in the Context of Complex Systems and Information Technologies. *Complexity, Science and Society Conference - Session on Philosophy and Complexity*, 11-14th September, The Centre for Complexity Research, University of Liverpool, UK.

DIJKSTRA E (1976). *A Discipline of Programming*, Prentice-Hall.

EDMONDS B (2003). Towards an Ideal Social Simulation Language. In *Multi-Agent-Based Simulation II*, *Lecture Notes in Artificial Intelligence*, v.2581, Springer-Verlag, pp. 105-124.

EPSTEIN J (1999). Agent-Based Computational Models And Generative Social Science. *Complexity*, v.4, n.5, pp. 41-59.

EPSTEIN J and 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, 1988, pp. 1048-1063.

FETZER J (1999). The Role of Models in Computer Science. *The Monist*, v.82, n.1 (General Topic: Philosophy of Computer Science), La Salle, pp. 20-36.

FETZER J (2001). Philosophical Aspects of Program Verification. In *Computers And Cognition: Why Minds are not Machines*, Kluwer Academic Publishers, pp. 220-245.

GERSHENSON C (2002). Philosophical Ideas on the Simulation of Social Behaviour. *Journal of Artificial Societies and Social Simulation (JASSS)*, v.5, n.3.

GILBERT N (1995). Simulation: an emergent perspective. New Technologies in the Social Sciences, 27-29th October, Bournemouth, UK.

GILBERT N and TROITZSCH KG (1999). *Simulation for the Social Scientist*, Open University Press.

GOLDSPINK C (2002). Methodological Implications Of Complex Systems Approaches to Sociality: Simulation as a Foundation for Knowledge. *Journal of Artificial Societies and Social Simulation*, v.5, n.1, http://jasss.soc.surrey.ac.uk/5/1/3.html.

GROSS D and Strand R (2000). Can Agent-Based Models Assist Decisions on Large-Scale Practical Problems? A Philosophical Analysis. *Complexity*, v.5, n.6, pp. 26-33.

KLUVER J, Stoica C and Schmidt J. (2003). Formal Models, Social Theory and Computer Simulations: Some Methodological Reflections. *Journal of Artificial Societies and Social Simulation*, v. 6, n. 2, http://jasss.soc.surrey.ac.uk/6/2/8.html.

PAPADIMITRIOU C (1994). *Computational Complexity*. Addison-Wesley.

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

SMITH BC (1995). Limits of Correctness in Computers. In *Computers, Ethics, & Social Responsabilily* (D. Johnson and H. Nissembaum, eds), Prentice Hall, pp. 456-469.

SMITH BC (1996). *The Age of Significance*, volumes I-IV, MIT press.

TROITZSCH KG (1997). Social science simulation — origins, prospects, purposes. In *Simulating*

Social Phenomena, *Lecture Notes in Economics and Mathematical Systems*, v.456, Springer-Verlag, pp. 41-54.

Return to Contents of this issue