SIMULATION AS FORMAL AND GENERATIVE SOCIAL SCIENCE: THE VERY IDEA*

NUNO DAVID[1], JAIME SIMÃO SICHMAN[2] & HELDER COELHO[3]
[1]*Department of Information Science and Technology, ISCTE*
Av. das Forças Armadas, 1649-026 Lisboa, Portugal
E-mail: Nuno.David@iscte.pt
[2]*Computer Engineering Department, University of Sao Paulo*
Av. Prof. Luciano Gualberto 158 tv.3, 05508-900 Sao Paulo SP Brazil
E-mail: Jaime.Sichman@poli.usp.br
[3]*Faculty of Sciences, University of Lisbon*
Bloco C5, Piso 1, Campo Grande, 1749-016 Lisboa, Portugal
E-mail: hcoelho@di.fc.ul.pt

The formal and empirical-generative perspectives of computation are demonstrated to be inadequate to secure the goals of simulation in the social sciences. Simulation does not resemble formal demonstrations or generative mechanisms that deductively explain how certain models are sufficient to generate emergent macrostructures of interest. The description of scientific practice implies additional epistemic conceptions of scientific knowledge. Three kinds of knowledge that account for a comprehensive description of the discipline were identified: formal, empirical and intentional knowledge. The use of formal conceptions of computation for describing simulation is refuted; the roles of programming languages according to intentional accounts of computation are identified; and the roles of iconographic programming languages and aesthetic machines in simulation are characterized. The roles that simulation and intentional decision making may be able to play in a participative information society are also discussed.

1. Introduction

The analysis of complex systems is being pursued with increasingly more sophisticated information technologies. In particular, the area of computer simulation has acquired a decisive role for analysing societies as complex systems, leaving behind the history of simulation as a secondary methodology in the social sciences. The sources of analogy between agent-based

*The title of this paper is inspired by James Fetzer's article "Formal Verification: The Very Idea". See Ref. 1.

technologies and social scientific models fomented an unprecedented interdisciplinary effort, which has been creating countless interfaces of research, across the computer and social sciences.

Several reasons exist for conducting a philosophical analysis of this scientific domain. In general, social science simulation has contributed to an inter- and multi-disciplinary scientific praxis,[a] thereby establishing new alternatives to traditional scientific methodologies. This should lead to the elaboration of new philosophical perspectives about the rules of the game as they are played in simulation.[b] Within the scientific community, the existence of methodological aspects that deserve better analysis is recognized. 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".[c] For others, the classical theory of computation does not support an adequate model of reality for simulation in the social sciences, and therefore the formal perspective of computation is not enough.[d] At any rate, the difficulties in constructing methodological perspectives on simulation raises interesting questions about the kind of scientific knowledge that simulation is providing.

Once the philosophy of simulation is analysed, it becomes clear that most essays do not take into account methodological and philosophical aspects of computer science, but are grounded mostly on aspects of social science. Among the diversity of perspectives that may be adopted, most should lead to recognizing additional ways to understand the concepts of computation and programming languages. However, few if any philosophical analyses in this field considered theoretical and practical limits of computation, as well as its new approaching challenges, well understood. Specifically, among the questions confronted by social science simulation is the extent to which formal and empirical methodologies are sufficient to describe the goals and methods of the discipline. In this paper we claim that they are not.

As we demonstrated in David et al. (2005)[8], simulation reveals new conceptions about the kind of scientific knowledge that computers provide.

[a]On the interdisciplinary structure of the scientific community in agent-based social simulation see Ref. 2.
[b]See the introduction to the issue of JASSS on epistemological perspectives on simulation, Ref. 3. See also Ref. 4.
[c]See specifically Ref. 5. See also the debate in Ref. 6.
[d]See Ref. 8.

Meanwhile, the meaning of social simulation is strongly connected with two traditional epistemic conceptions of scientific knowledge. The first conception is the formal view of mathematical knowledge, reminiscent of the computer science formal tradition and congruent with the idea of simulation as numerical calculation. The second refers to the experimental character of simulation, insofar as scientists run their programs in computers like an experimental set-up. Programs, in both senses, are viewed as descriptions of social theories or phenomena that, unlike most theories in the social sciences, are viewed as formal mathematical models.

The conflation of the formal and the experimental perspectives of simulation lead many scientists to pose simulation as a way of testing representations of social theories or phenomena that could, in principle, be deduced from general principles. This is based on three tacit methodological assumptions that draw on the formal tradition of classical computational theory, which we will challenge throughout this article:

Assumption 1. The process of executing a program in a computer can be formally modelled, and thus understood as an automatic process of formal inference. This position was explicitly advocated by Epstein (1999)[10], drawing directly on the Turing-Church thesis.[e]

Assumption 2. The sensitivity of complex models to initial conditions does not permit deducing its simulated results from general principles, although it would be possible to do it in principle. Casti, for instance, described this conception in a very explicit way:

"*In principle*, one could trace all the statements in the program representing the simulation and discover exactly why things unfolded the way they did (...) it just cant be done *in practice*." (Casti, 2001, p.14, our emphasis)[11]

Assumption 3. The experimental character of simulation in the social sciences can draw on the experimental character of simulation in the natural sciences.[f] Insofar as the sensitivity of general complex models explains the inability to deduce its simulated results from general principles, the sensitivity of social-scientific models explains similarly the inability to deduce its simulated results from general principles. Casti, for instance, tacitly conflated the present and the previous assumptions as follows:

[e]For an introduction to classic computer theory, also known as the Church-Turing thesis, see e.g. Ref. 9.

[f]For a comprehensive critic see Ref. 8; see also Ref. 12.

> "For the first time in history we are in a position to do bona fide *laboratory experiments on these kind of complex systems.* (...) We can use these surrogates as *laboratories* for carrying out the experiments needed to be able to construct viable theories of complex physical, *social*, biological and *behavioural* processes." (Casti, 1997, p.35, our emphasis)[13]

Recently, a number of essays debated epistemological perspectives on social science simulation.[g] Either by viewing simulation as a process of imitation[12], stylized facts [14], or intentional adequacy between programs and theory[8], the tendency is to emphasize the interpretative character of social science in simulation, notwithstanding its application as a useful methodology for approaching complexity. The essay presented by David et al.[8], in particular, advocates the perspective of intentional computation as the approach able to comprehensibly reflect the multiparadigmatic character of social science in terms of agent-based computational social science. Conversely, essays presented by Moss and Edmonds[15] or Boero and Squazonni[16] argue that simulation provides the social sciences with a powerful instrument to generate empirical evidence, and thereby, contribute to better social sciences. Whereas these two trends may not be contradictory, the formal character of computer science pervades the scientific culture of simulation, as well as most of its methodological arguments. However, the formal perspective does not seem to be compatible with any such views.

In this chapter the classical account of computation is demonstrated to be inadequate to secure the goals of simulation in the social sciences. Simulation does not resemble formal deductive demonstrations or generative mechanisms that explain how certain agent-based models are sufficient to generate emergent macrostructures of interest. The justification of results implies additional epistemic conceptions of scientific knowledge, which are tacitly being used in the discipline. Three kinds of knowledge that account for a comprehensive description of the discipline from an epistemological point of view are identified: formal, empirical and intentional knowledge. Intentional knowledge should be considered an outcome of an experimental exercise, albeit not empirical, acquired within a context of limited consensus.

The structure of our argument in this chapter is composed of two parts. In the first part the formal and generative perspectives of computation in

[g]See the introduction to the issue of JASSS on epistemological perspectives on simulation, Ref. 3.

literature are recalled, proceeding afterwards to refuting the use of such conceptions for describing the scientific practice of simulation. In the second part the role of programming languages in simulation, according to intentional accounts of computation, is discussed. Two types of programs and programming languages in simulation are identified: Programs as text and programs as icons; and languages as abstract machines and languages as aesthetic machines. The use of abstract languages confirms that the method of simulation incorporates formal and empirical methodologies. The use of aesthetic languages demonstrated that it depends fundamentally on intentional methodologies. The roles that intentional decision making may play in a participative information society are also discussed.

2. An Ontological Confusion

Conventional methodologies of computer science model the mechanism of executing a program in a computer like a process of formal inference. In complexity sciences, the process of running a program was described not only as an automatic inference procedure but as a formal deductive procedure itself. The computer is seen as a mechanism of formal calculus, where programs represent mathematical functions that map inputs into outputs. The calculus can be modelled in several ways, one of which ascribes the computer the capacity to prove first-order theorems according to a fixed vocabulary. Considerations of brevity and simplicity lead us to call this tradition the FDE argument, as per 'Formal Deduction through Execution'. Notwithstanding, our goal is quite the opposite, namely to demonstrate that simulation shall not be legitimized under the presumption of resulting from a calculus of formal inference. Additional conceptions of knowledge are needed.

2.1. *Against Generative Sufficiency of Growing Artificial Societies From the Bottom Up*

The relationship of simulation with the formal perspective of computation was rigorously advocated by Epstein (1999, p.44)[10] in his account of "generative social science", where he writes:[h]

> " . . . if one accepts the Church-Turing thesis then every computation — including every agent-based computation — can be executed by a suitable register machine. It is then a theorem of logic

[h]The concept of generative social science was also adopted in Ref. 17.

and computability that every program can be simulated by a first-order language."

The point is the following:

> " . . . for every computation there is a corresponding logical deduction, and this holds even when the computation involves 'stochastic' features since, on a computer, these are produced by deterministic pseudo-random number generation. Even if one conducts a statistical analysis over some distribution of runs, each run is itself a deduction." (Epstein, 1999, p.44)[10]

The methodological point of social science simulation would be to generate observed social phenomena in computers and thus deductively explain the social phenomena, insofar as there is an intellectual tradition upon which "we deduce propositions expressing observations from other more general propositions".[i] Accordingly, scientists "seek to explain macroscopic social phenomena by generating it in an agent-based computational model". Moreover, "in that event, we can claim that they [the explanations] are *strictly deductive*." (p.43, our emphasis and brackets)[10]

We may state Epsteins argument of generative sufficiency as follows:

The Generative Sufficiency Argument. Agent-based models provide formal demonstrations that a given microspecification is a *sufficient* condition, albeit not a necessary condition, to generate a macrostructure of interest.

And this leads Epstein to conclude:

"From an epistemological stand point, generative social science, while empirical, is not inductive, at least as that term is typically used in the social sciences." (1999, p.43)[10]

2.2. *'Generative' from a Philosophy of Computer Science Perspective*

The arguments of Epstein can be analyzed from the perspective of the philosophy of computer science. The term 'formal' is ubiquitous in computer science. Smith[19], for instance, has acknowledged the ambiguity of the term:

[i]Epstein's account of scientific explanation seems to be inspired by the work of classical empiricists, such as in Ref. 18.

> "People may believe that developing an idea means formalizing it, and that programming languages are formal languages, and that theorem provers operate on formal axioms — but few write 'formal' in formal axioms or daily equations. Moreover, a raft of different meanings and connotations lie just below the surface. Far from hurting, this apparent ambiguity has helped to cement popular consensus. Freed of the need to be strictly defined, formality has been able to serve as a lightning rod for a cluster of ontological assumptions, methodological commitments, and social and historical biases. Because it is tacit, goes deep, has historical roots, and permeates practice, formality has been an ideal foil, over the years, with which to investigate computation. Almost a dozen different readings of 'formal' can be gleaned from informal usage: precise, abstract, mathematical, a-contextual, digital, explicit, syntactic, non-semantic, etc."

In a Church-Turing theoretic account, the meaning of 'formal' appears connected to the *antisemantical* reading mentioned above. That is, the idea that a symbolic structure is formal just in case it is manipulated independently of its semantics, in which it is assumed that a theorem is derived by an automatic inference regimen.

To some extent, the pervasive use of the term 'formal' arises from conflating the terms 'program computation' and 'program execution' into one single meaning, conveying the same ontological status to two fundamentally distinct processes.[7] The concept of 'abstract machine' in classical computation is conflated with the concept of 'physical machine' in software engineering. The observed behaviour of a program executing in a computer, which should be the subject of research in the first place, gives way to a computation that is, in essence, a formal model or a theory itself. As Epstein himself states (1999, p.44)[10], "each run is itself a deduction" — the actual execution of a program in a physical machine is a theory or a computation in itself.

It would be reasonable to question whether the idea of Epstein is not to illustrate anything more than an ideal perspective of computation in real computers. Yet, his efforts are not limited to illustrating that simulation implies deductive and generative conceptions of scientific research but that simulation implies a new kind of scientific empirical research. The point of simulation, in such a sense, is to provide agent-based specifications for which the corresponding program execution should generate patterns that match empirical data — for instance, to implement in a computer an agent-based archaeological model that succeeds in generating behaviours that match empirical data and thus explain why the Kayenta Anasazi population of Long House Valley[j] vanished at some point from the valley (1999, p.44)[10]. The role of computer science, therefore, is one of formal inference, whereas the role of social science simulation is to use computers to generate behaviours that should match social scientific data.

Apart from the arguments of Epstein, there is wide evidence confirming a tacit association of simulation with formal deduction, for instance, by suggesting the high levels of control and objectivity that scientists attribute to simulation results. The claim of Casti[11] that it is possible to trace all statements in the simulation is obvious evidence. Prietula et al.[20] advocate that: "computational models are generally less noisy, easier to control, more flexible, more objective". Sawyer explains: "the contributions of artificial societies to sociological theory will be primarily to theories that are characterized by logical rigor (. . .), which allow precise and logical deductions from abstract principles to empirical hypotheses (. . .) Artificial societies resemble axiomatic theory in the sense that their propositions are explicitly stated in the form of algorithms or program code and valid derivations may be drawn systematically by running the program" (2003, p.332)[21]. For Axelrod, one of the advantages of simulation is that "there are no messy problems of missing data or uncontrolled variables as there are in experimental or observational studies" (1997, p.27)[22].

The most cited sentence of Axelrod, describing simulation as a bridge between induction and deduction, is actually not far from the idea. Whereas it is claimed that simulated data must be analysed inductively, it is suggested that the data are a necessarily valid consequence of the rules specified:

> "Simulation is a third way of doing science. Like deduction, it starts with a set of explicit assumptions. But unlike deduction, it does not prove theorems. Instead, a simulation generates data that can be analyzed inductively. Unlike typical induction, however, the simulated data comes from a rigorously specified set of rules rather than direct measurement of the real world. While induction can be used to find patterns in data, and deduction can be used to find consequences of assumptions, simulation modelling can be used as

[j]A small region in northeastern Arizona.

an aid to intuition." (Axelrod, 1997, p.25)[22]

However, insofar as the data can be analysed inductively, it seems difficult to conceive how it was generated in the first place without "direct measurement of the real world". Computers in this sense do not belong to the "real" world. That is to say, the tacit methodological assumption of social science simulation would propose the same ontological status to both a program execution in a physical machine and a formal computation in an abstract machine. We shall call this assumption the FDE argument, as per Formal Deduction through Execution:

The Argument of Formal Deduction through Execution. It is based on three methodological beliefs, which we will show to be contradictory with one another:

A) The mapping of classical computer theory into the logic of the method of social science simulation. B) The conflation of the terms 'computation' and 'execution'. C) The experimental character of simulation, upon which an unexpected result can be a reflection of a mistake in the implementation (bug) or a surprising consequence of the program itself (see e.g. Axelrod, 1997, p.27)[22].

2.3. *Refutation of Formal Deduction through Execution*

In this research, it is claimed that the role of computation in social simulation is not one of formal deduction. The goal is to refute the FDE argument. From a scientific-philosophical perspective our goal may seem somehow trivial. However, we demonstrated that the formal perspective of computer science is recalcitrant within social science simulation. Moreover, the objection to the FDE argument suggests yet another objection to current philosophical thinking in the literature: The objection to characterizing simulation as a basic and alternative epistemic conception to deduction or induction.

The objection to the FDE argument parallels an old, but still omnipresent, debate in computer science, centered around the merits of the so-called "formal verification of programs". The merits of formal methods for verifying programs became particularly controversial by the end of the Eighties and in the beginning of the Nineties after James Fetzer published an eloquent article in the 'Communications of the ACM'[1,23]. On one side of the debate were gathered those who considered that computer programming was reducible to mathematics — inspired by Hoare or Dijkstra[k] — and the other side was composed of those who saw it as applied mathematics or empirical science. The claim of Fetzer was that the formal verification project carried misleading ideas. The seminal work of Hoare is an obvious example:

> "Computer programming is an exact science in that all the properties of a program and all of the consequences of executing it in any given environment can, in principle, be found out from *the text* of the program itself by means of purely deductive reasoning." (Hoare, 1969, p.576, our emphasis)[25]

These ideas often turned out to be misleading. For instance, it was often claimed that the intended behaviours of computers could be completely specified and verified before the corresponding programs were executed on specific computers, by means of purely formal methods. Computer science would be viewed as pure mathematics rather than as applied mathematics or empirical science.

Fetzer‘s philosophical refutation of the formal verification project consisted of distinguishing two kinds of programs: those that are and those that are not in a suitable form to be compiled and executed in a machine. The difference between the programs is that the former is verified by reference to abstract machines, whereas the latter require the existence of compilers, interpreters and target machines. Compilers, interpreters and processors are properly characterized according to specific target physical machines. Insofar as a program in an abstract machine does not possess any significance for the performance of a target machine, the performance of that program can be verified formally and conclusively. Conversely, to the extent that the performance of a program possesses significance for the performance of a target machine, that program cannot be conclusively verified *a priori*. The program must be verified empirically by means of program testing. Hence, the formal verification project is not viable.[l] Program testing should be the crucial technique to ascertain the proper behaviour of programs and computers, notwithstanding the use of formal methods during the stages of analysis and design.

Our goal is to informally reduce the FDE argument in social science simulation to the formal verification project in computer science. Consider the

[k]See e.g. Ref. 24.
[l]For more details see Ref. 1 or 23.

formal verification project according to its most radical terms. The intent would be to create formal methodologies that could guarantee that a given specification would correspond to the behaviour of a program executing in a computer. That is, to find deductive procedures to verify conclusively the correctness of a program P in relation to a specification $F : I \rightarrow O$, in order to guarantee that the execution of P with inputs I would result exactly into the specified outputs O. The following argument reduces FDE to the formal verification project.

The Argument of FDE Refutation. Consider a specification $F1 : I1 \rightarrow O1$ and a program $P1$ as *text* that can be read, edited, printed. The computation of $P1$ with inputs $I1$ is denoted by $P1(I1) = O1$ and the execution of $P1$ after implementation is denoted by $P1^{\rightarrow}(I1) \approx O1$. Suppose that $P1(I1) = O1$, according to a proof of partial correctness.[m] Suppose, however, that $P1^{\rightarrow}(I1) \approx O2$, that is the computation and the execution of $P1$ leads to different results, compatible with assumptions B and C in the FDE argument. However, according to assumptions A and B in the FDE argument, there is a specification $F2 : I1 \rightarrow O2$ and some program $P2$ such that the computation with input $I1$ leads to $O2$, i.e. $P2(I1) = O2$. So, a specification $F2 : I1 \rightarrow O2$ exists such that the execution of $P1$ and the computation of $P2$ satisfy $F2$. The formal verification project is thus possible: the behaviour of $P1$ execution (as well as $P2$ computation) *necessarily* corresponds to the specification $F2 : I1 \rightarrow O2$.

In short, from a methodological point of view, the FDE argument cannot be sustained. At the very least it is misleading. Even though it does not suggest that both the computation and the execution of $P1$ necessarily give the same outputs, which would *prima-facie* instantiate the formal verification project, it may suggest that the execution of $P1$ *necessarily* corresponds to a formal computation of *some* program $P2$, which is a methodological absurdity. The formal perspective of computation is not enough to support an adequate model of reality for simulation in the social sciences. The logic of simulation implies distinct types of program verifications that reflect epistemological distinctions in the kind of knowledge one can have about programs. One obvious type of knowledge is empirical knowledge obtained through program testing. Another one is intentional knowledge,[8] which is discussed in subsequent sections.

[m]On proofs of partial correctness see e.g. Ref. 25.

A final comment emerges regarding the perspective of Axelrod[22] that poses simulation as a contrast to both induction and deduction, which is after all not far from the formal perspective of Espstein. Whereas Axelrod defines induction as the "discovery of patterns in empirical data", deduction is understood as the specification of "a set of axioms and proving consequences that can be derived from those assumptions" (1997, p.24)[22]. Yet, there is no reason for not viewing deduction as a kind of empirical enquiry. Popper[26], as many other 'deductivists' in the philosophy of science, would say that there is no such thing as induction.

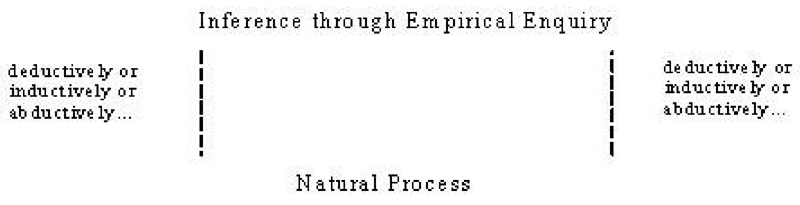
Inference through Empirical Enquiry
deductively or inductively or abductively...
deductively or inductively or abductively...
Natural Process

Figure 1. A theory square.

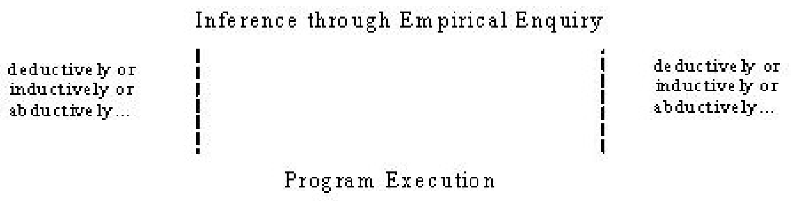
Inference through Empirical Enquiry
deductively or inductively or abductively...
deductively or inductively or abductively...
Program Execution

Figure 2. A second theory square.

Consider Figure 1. The theory square represents a traditional epistemic question in the philosophy of science with respect to empirical science: Are correlations between natural properties enquired inductively or deductively? Likewise, it is equally fair to use a theory square in relation to the behaviour of a program executing in a computer, just like in Figure 2. Indeed, whether the enquiry may be inductive, deductive, or even abductive depends on the relevant methodological conception of scientific enquiry, which is by no means a specific dilemma of simulation. To define the epistemic specificities of simulation based on such contrast does not seem to be significantly informative to the point in question.

3. The Role of Programming Languages

An informative way to analyse the epistemic status of simulation is to examine how simulations are programmed. Program implementation plays a fundamental part in constructing a simulation. The role of intentional methodologies becomes crucial once it is realised that, rather than one kind of program and programming language, two kinds of program and programming language are used in the implementation of a simulation. Simulations are constructed by means of successive language translators, from program to program, by using simulation platforms, compilers and interpreters. In computer science we usually view a programming language as an abstract machine and a program as a textual and static entity, which may be read, edited, printed. However, in social simulation we have identified yet another type of programming language and kind of program, which involve the use of intentional knowledge. At least two kinds of programming languages are used in the process of implementing simulations, iconographic and textual programming languages.

Iconographic Programming Languages. Iconographic programming languages consist of a set of subjective rules that model the behaviour of an aesthetic machine. A subjective model, eventually associated with a specific domain composed of organisational or aesthetic abstractions, such as groups of agents, grids, movement, constraints, roles, levels, messages, societies, or specific behaviours, 'segregation rules', 'sexual rules' and 'genetic crossing', 'culture', 'race', 'influence', 'friendship', 'innovation', 'state nations' or 'political actors'. For instance, the CORMAS platform,[n] as well as the Swarm simulation system,[o] support aesthetic human-machine interfaces to specify interactions between individuals and groups sharing renewable natural resources. The commands in the programming languages are icons represented on the screen rather than structures in textual languages. A program is a set of selected icons with no definitive explicit order. The user selects icons with the mouse, after which an automatic code generator transforms icons into sections of code in high-level programming languages, such as SmallTalk or Objective C. However, the first order logics of classic

[n]Common-pool Resources and Multi-Agent Systems, see http://cormas.cirad.fr.
[o]See http://www.swarm.org.

computer theory can hardly describe these transformations. Moreover, the sections of code are later linked in some arbitrary way by the user. The specific mapping from icon-level language to high-level language is subjective, domain specific and validated according to a limited level of consensus. In the case of participative-based simulation, stakeholders may be involved in the specification of iconographic programs, but hardly in the semantic process of mapping iconographic programs to high-level programs.

Textual Programming Languages. Textual programming languages include the usual high-level and low-level languages. A *high-level language*, as defined by Fetzer[27], is a set of abstract rules that model the behaviour of an abstract machine. High-level languages have clear formal semantics, and contrary to icon-level languages, the meaning of their commands must not be subjective. The advantage of programming with high-level languages, such as Java or Objective C, is that there is a one-to-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 mapped to low-level languages and later executed by target machines whose operations are causally affected by machine code, which usually consists of sequences of zeros and ones.[27] A *low-level language* is a set of abstract rules that model the behaviour of abstract or target machines. 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. 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.

The distinction between programs as text and programs as icons, as well as between abstract and aesthetic machines, reveals the intentional character of social simulation methodologies. Whereas the consistency between two different abstract machines can be specified formally and verified empirically, the consistency between abstract and aesthetic machines must be verified intentionally. Indeed, the modelling of target machines by means of abstract machines can be defined according to formal logic, drawing on classic computer theory. Hence, the relative consistency between the high-level and low-level abstract machines can be demonstrated formally, as well as tested empirically against the observed behaviour of the program. Moreover, insofar as most high-level programming languages stand for abstract machines rather than physical machines, we can say that the process of implementing a high-level program involves the construction of a sequence of embedded models that are causally connected to a target machine.

In contrast, whereas abstract machines are specified with well-defined formal semantics, the meaning of iconographic languages and aesthetic machines is negotiated intentionally by the members of the team implementing the simulation, as well as by the stakeholders involved. Since iconographic languages must be mapped into high-level, and ultimately into low-level languages, it becomes clear that the process of implementing icon-level programs involves the construction of a sequence of embedded models connected intentionally to a target machine. The target machine is modelled intentionally according to a limited level of consensus and tested experimentally, albeit not empirically, against the observed behaviour of the icon-level program.

From this point of view, the implementation of a program can be viewed as the action of embedding models within models, where the notion of embedding may be envisioned as an intentional, causal or logical relation. In Figure 3, Fetzer's diagram of language embedding[27] is expanded with icon-level programs and aesthetic machines. The thin arrows represent a possible relation between programs and machines represented by programming languages. The thick arrow represents an actual relation between a low-level program and a target machine. The series of three black dots stands for the possible existence of automatic code generators, compilers and interpreters that effect some causal connection between programs/machines at different levels of embedding. In addition, the series of unfilled dots stands for the existence of implementation teams and stakeholders that exercise intentional connections between abstract and aesthetic programs/machines at different levels, according to a limited level of consensus.

The use of iconographic programming languages demonstrates that the logic of simulation incorporates formal and empirical methods, but largely surpasses the use of formal and empirical methodologies. The results of a simulation are outcomes of experimental set-ups, but the results of the experiments can hardly be represented by material conditions of necessity

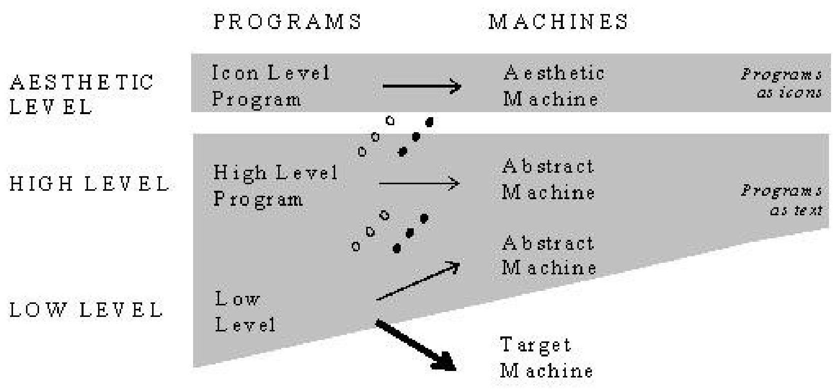

Figure 3. Programs and languages as models — an extension to the diagram of Fetzer with icon-level programs and aesthetic machines.

relating facts about the objective behaviours of the program. The results are appropriately characterized by conditions of intentionality that relate aesthetic components in the program, negotiated according to a limited level of consensus.

4. A Different Idea of Simulation

We shall recall the traditional assumptions of simulation in the social sciences, presented in the introduction, which we have been challenging throughout this article. Firstly, whereas it is true that the process of executing a program can be modelled according to a mechanism of formal inference, it does not seem to be significantly relevant to the methodology of social science simulation. Secondly, whereas sensitivity to initial conditions is an important issue in the social sciences, it is not the primary reason explaining the inability to deduce simulation results from general principles, at least from a technical point of view.

A tremendous semantic gap exists between the formal-empirical and the intentional perspective of computation, both of which are used to interpret the behaviour of simulations. The misleading assumption of generative social science is that both formal and intentional representations can be legitimised deductively, insofar as the process of program execution can be understood as a formal deductive mechanism. This reasoning is unsustainable. Firstly, the vocabularies of the low-level abstract machine (e.g. memory registers, bit logical operations), as well as the vocabularies of the high-level machine (e.g. complex data structures, objects, graphics), are not identical to the vocabularies of the icon-level machine (e.g. agents, grid, movement, culture, segregation rules). The parts that those vocabularies designate in the world are not the same; from a strict formal point of view the consistency between machines is incommensurable. Secondly, although the consistency between low-level and high-level abstract machines can be verified empirically, the consistency between abstract and aesthetic machines must be verified intentionally. But unlike the process of empirical adequacy, no formal computational theory is available to justify the process of intentional adequacy.[p]

The characterization of simulation as a research practice implies additional epistemic conceptions of scientific knowledge that are being used tacitly in the discipline. There are at least three different kinds of knowledge that can be acquired from computer simulation: formal, empirical and intentional knowledge. Intentional knowledge should be considered an outcome of an experimental exercise, albeit not empirical, acquired within a context of limited consensus. Social science simulation, like the social sciences, is interpretative and diverse theoretically and methodologically. To imagine that simulation could 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 sciences, would be a mistake. The conditions for the acceptance of a simulation depend on the particular theoretical-methodological context of the social scientist, can be interpretative and subjective, and may depend on the socioeconomic and sociocultural context. The perspective of intentional computation seems to be the one able to reflect the multiparadigmatic character of social science into social science simulation.

Finally, the observation that social science simulation is multiparadigmatic demonstrates the need for participative contexts. As a growing information technology, simulation is being used to assess concrete socioeconomic and environmental/ecological problems. Simulation is a useful methodology to approximate scientists and stakeholders. However, only within a specific interpretative context can a specification or a program be considered as a set of sufficient conditions to explain the observed behaviour

[p]Moreover, as we mentioned before, even if we ought to stay on the empirical plan (which is not the case), whether the epistemic conception of inquiring the behaviour of computers could be an inductivist or a deductivist one that would be by no means a specific dilemma of simulation.

of a simulation. Indeed, from a sociological perspective, simulation should help us pose the process of science as critical thinking in a democratic context.

Acknowledgements

Jaime Simão Sichman is partially supported by CNPq, Brazil, grants 482019/2004-2 and 304605/2004-2.

References

1. James Fetzer, Program Verification: The Very Idea, *Communications of the ACM*, 31, 1048-1063 (1988).
2. Nuno David, Maria Marietto, Jaime S. Sichman, Helder Coelho, The Structure and Logic of Interdisciplinary Research in Agent-Based Social Simulation, *Journal of Artificial Societies and Social Simulation*, 7(3), ¡http://www.soc.surrey.ac.uk/JASSS/7/3/4.html¿ (2004).
3. Ulrich Frank and Klaus Troitzsch, Epistemological Perspectives on Simulation, *Journal of Artificial Societies and Social Simulation*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/7.html¿ (2005).
4. Carlos Gershenson, Philosophical Ideas on the Simulation of Social Behaviour, *Journal of Artificial Societies and Social Simulation*, 5(3), ¡http://jasss.soc.surrey.ac.uk/5/3/8.html¿ (2002).
5. J. Kluver, C. Stoica and J. Schmidt, Formal Models, Social Theory and Computer Simulations: Some Methodological Reflections, *Journal of Artificial Societies and Social Simulation*, 6(2), ¡http://jasss.soc.surrey.ac.uk/6/2/8.html¿ (2003).
6. Rosaria Conte, Bruce Edmonds, Scott Moss, Keith Sawyer, Sociology and Social Theory in Agent Based Social Simulation: A Symposium, *Computational and Mathematical Organization Theory*, 7(3), 183-205 (2001).
7. Nuno David, Empirical and Intentional Verification of Computer Programs in Agent-based Social Simulation, Ph.D., University of Lisbon (in Portuguese) (2005).
8. Nuno David, Jaime S. Sichman, Helder Coelho, 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*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/2.html¿ (2005).
9. C. Papadimitriou, *Computational Complexity*, Addison-Wesley (1994).
10. Joshua Epstein, Agent-Based Computational Models And Generative Social Science, *Complexity*, 4(5), John Wiley & Sons, 41-59 (1999).
11. John Casti, Would-Be Business Worlds. *Complexity*, 6(2), John Wiley & Sons, 13-15 (2001).
12. Günter Küppers and Johannes Lenhard, Validation of Simulation: Patterns in the Social and Natural Sciences, *Journal of Artificial Societies and Social Simulation*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/3.html¿ (2005).

13. John Casti, *Would-be Worlds: how simulation is changing the frontiers of science*, John Wiley & Sons (1997).
14. Bernd-O. Heine, Matthias Meyer and Oliver Strangfeld, Stylised Facts and the Contribution of Simulation to the Economic Analysis of Budgeting, *Journal of Artificial Societies and Social Simulation*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/4.html¿ (2005).
15. Scott Moss and Bruce Edmonds, Towards Good Social Science, *Journal of Artificial Societies and Social Simulation*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/13.html¿ (2005).
16. Riccardo Boero and Flaminio Squazzoni, Does Empirical Embeddedness Matter? Methodological Issues on Agent Based Models for Analytical Social Science, *Journal of Artificial Societies and Social Simulation*, 8(4), ¡http://jasss.soc.surrey.ac.uk/8/4/6.html¿ (2005).
17. Joshua Epstein, Robert Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*, MIT press (1996).
18. C. G. Hempel and P. Oppenheim, Studies in the Logic of Explanation, *Philosophy of Science*, 15, 567-579 (1948).
19. Brian Cantwell Smith, The Foundations of Computing, Smiths introduction to a series of books that report his study of computing in the books The Age of Significance: Volumes IVI. Available at ¡http://www.ageosig.org/people/bcsmith/papers¿ (1996).
20. M. J. Prietula, K. M. Carley and L. Gasser, *Simulating Organizations*, MIT Press (1998).
21. Keith Sawyer, Multiagent Systems and the Micro-Macro Link in Sociological Theory, *Sociological Methods and Research*, 21(3), 37-75 (2003).
22. Robert Axelrod, Advancing the Art of Simulation in the Social Sciences, *Simulating Social Phenomena*, Springer Verlag, 21-40 (1997).
23. James Fetzer, *Computers And Cognition: Why Minds are not Machines*, Studies in Cognitive Systems, 25, Kluwer Academic Publishers (2001).
24. Edsger Dijkstra, *A Discipline of Programming*, Prentice-Hall (1976).
25. C. A. R. Hoare, An Axiomatic Basis for Computer Programming, *Communications of the ACM*, 12, 576-580 (1969).
26. Karl Popper, *The Logic of Scientific Discovery*, Routledge Classics (1935/1992).
27. James Fetzer, The Role of Models in Computer Science, *The Monist*, 82(1), La Salle, Illinois 61301, 20-36 (1999).