This paper was published in: *Salzburger Geographische Arbeiten*, Band 48, S. 111 - 120, Salzburg 2012. (ESSA Congress 2012)

How Can We Simulate a Society?

Wolfgang Balzer LMU, Karl R. Brendel and Solveig Hofmann

Abstract We introduce a notion of a simulation system for societies which is similar to the notion of theory found in the literature about empirical theories. In a theory the data are produced empirically whereas in pure simulation systems the data are all hand made or generated by a computer. We describe how a series of simulations can be generated automatically by a list of systematically constructed sets of constants.

I Introduction

A society, as we all know, is a very complex system; also it is partially non-transparent. In the natural sciences data are found by experiment and by the analysis of a real system. This does not really work for the humanities because there are often moral or economical problems. The method of simulation can avoid these problems for data. But at what cost?

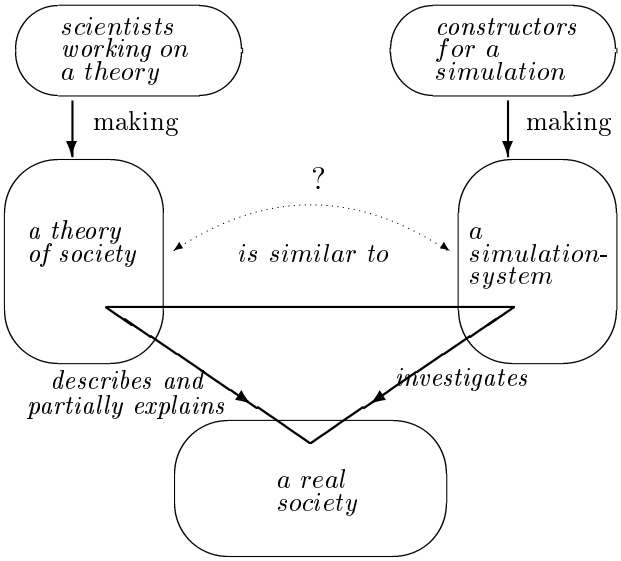

Fig. 1 Social theory vs. simulation: an overview

In a simulation we intend to build a system which is – in some way – similar to

some societies we want to understand. It is clear that, today simulation programs for 'societies' are rather simple and rather small, in relation to the complexity of a real society. Nevertheless, computer simulation for social systems is a success story. In the 1970s, the first attempts by Axelrod, Holland, Schelling and others opened the door. In a collection *Artifical Societies* (GILBERT & CONTE, 1995), papers are written with the goal to improve our understanding of societies with the help of computers. Today we have world wide conferences, like WCSS and SCS, journals like JASSS, and many articles and books.[1]

There are now many simulations which investigate social phenomena, but the understanding of societies improves rather slow. The big problem is 'the' structure of one society and the structure of a *class* of societies, as far as we can grasp it. A part of this problem is to delineate the boundary or closure of a system at the level of reality and at the level of a model. In sociology there is a number of rather different approaches, and also the methodologies are divers. We approach simulation systems here in a structuralistic manner, in a way similar to scientific theories as described in many works (BALZER et al., 1987), (BALZER et al., 2000), (DIEDERICH et al., 1989), (DIEDERICH et al., 1994).

In one area of philosophy of science (STEGMÜLLER, 1976) a theory is made by a group of scientists with the goal of describing and explaining an intended domain of real systems from real world. This implies to clarify which parts, dimensions, properties, aspects of a real system are necessary in a description and explanation. In other words, how can the scientists formulate the closure of a definitive system? In a similar way a simulation system is made by a group of researchers which investigate an intended domain of real societies. At this moment it is clear that we cannot claim to describe, let alone explain a real society, even partially.

2 Theory and Simulation System

A *theory* has several parts. On the elementary level, a theory consists of a list of *hypotheses*, a list of *data*, a class of *models*, a *set of intended applications*, and of an approximation apparatus *approx*. An intended application is a real system, which had been, or will be, investigated by a group of scientists. This implies that the scientists delineate the boundaries of a real system. In the following we pick one model, and we suppose that this model is a model for an intended application.[2] Data and models are described by sets of *sentences* of the theory. Metaphorically speaking, the data are an anchor for keeping the hypotheses fastened to the ground. In this way, we can separate *successful* intended applications from *failures*.

In the same way, a *simulation system* has a (*computer*) *program* (a list of program rules), a list of *input data*, a class of (*computer*) *runs* – relative to a given computer or a computer system –, and a domain of *successful* runs, and a *statistical apparatus*. In addition to a theory, a simulation system has a class of *outputs*. Using the input data, a program run produces new facts that did not exist before the program start. A

[1] In our group, for instance, there are German books: (BRENDEL, 2010), (HOFMANN, 2009), (PITZ, 2000).

[2] Details are found, e.g. in (BALZER et al., 1987), Chap. 2.

run is normally described by a list of state transitions, so that a state of the program will be transformed into 'the next' state, by a program rule. Graphically, a run can be represented by states and by changes of states.

As a program can involve rules using random elements, newly produced facts cannot be predicted, This is only approximately true because in a real computer all the random numbers are determined by recursive functions. For pratical uses, it is therefore often necessary to repeat the program several times starting with different random seeds. In the following, we distinguish between *runs with repetition* and *simple runs* (or just *runs*). In a run with repetition, the input data are used several times. In other words, a simple run can contain random elements but the corresponding results are not analysed statistically. Normally, the statistical evaluations can be programmed by an 'outer' loop. We skip this aspect here for reasons of space and simplicity.

In one run we distinguish between new facts produced by the program, which can be deleted before the run stops (which we call *dynamic facts*), and all other new facts which we call *results* (relative to a run).

In this formulation a hypothesis corresponds precisely to a program, i.e. to a list of *program rules*, and the data of the theory correspond to the input data of the simulation system. The notion of deduction in a theory is important. In a theory, a sentence can be inferred from the hypotheses and the data. Finally, the successful applications of the theory corresponds to the successful runs of a simulation system. In a similar way a term (or a formula or a sentence) is deduced in a simulation system, if this term is generated by the program rules and the input data. We say that a run is successful if the output of the run corresponds to results, which are intended by the programmers. In this way, we can say that successful applications of the theory correspond to the successful runs of a simulation systems.

Unfortunately, the notion of a model has three different meanings. In logics, a model is a complex set, in computer sciences – including social simulations – a model is a representation or a 'picture' of a 'target' system (which often is a real system) and in normal speech, a model is a 'target' system which is represented by a picture or by other means. A further complication comes from 'micro', 'meso' and 'macro' levels, and the corresponding problem of scale. 'We' – the authors – follow the first characterization mentioned above. On the one hand, we prefer viewing models as set-theoretical entities, because they can be precisely be described in a human way (if a description is not super-formalized). On the other hand, we also favour the idea that the data of the facts corresponding to a target system should be stored separately. We use here the notion of model in the logical meaning, but *only if* the data are stored separately. We have no clear view about the problem of scale reduction in the case of a specific society; we are open here.

A theory (see Figure 2 at the left side) describes and explains a given society. The hypotheses and the data are interpreted in a (set-theoretic) model which represents or 'depicts' a real society. In our account the data of a theory are split into subsets. One subset of data 'belongs' to one real system, so that a group of scientists intend to investigate this system, and the corresponding data. Often a data item belongs to

two different systems at the same time.[3] For instance, in one period a person can live in two societies 'at the same time', so one data item is an element of several data sets.

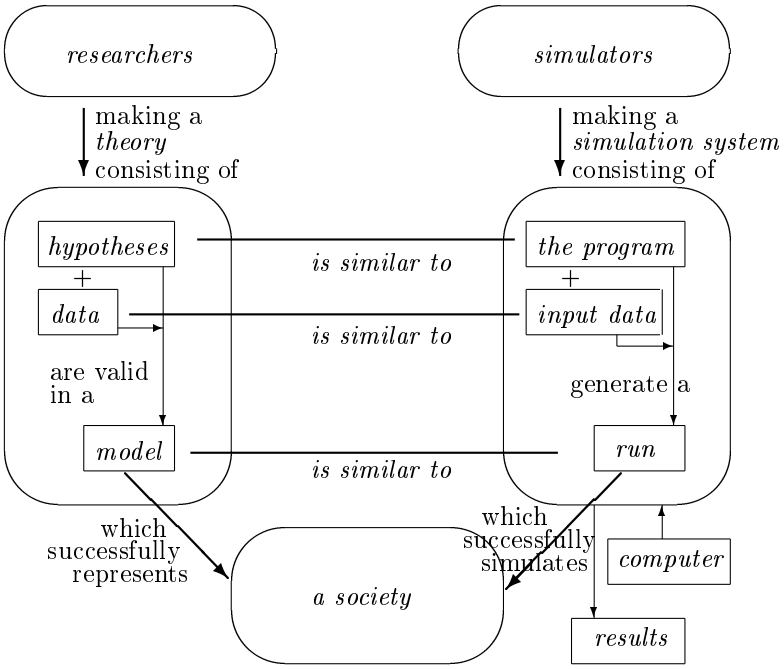

Fig. 2 Social theory vs. simulation: in detail

In a structurally similar way a simulation system (see Figure 2 at the right side) generates a run. The program and the input data produce many transitions which altogether form the run. After the program is started and the input data are loaded, new facts and results will be generated. During the computation, many results are produced and stored in newly generated files. The claim of describing or even explaining a real society is very weak. The content of such a claim amounts to the program, the input data and the newly generated results.

The interesting point here is the similarity between the theory and the simulation system. In the centre of Figure 2 we see the relation of similarity between the hypotheses and the program. This kind of similarity can be investigated by syntactic and semantic tools, like parsers and transation manuals, e.g. (CHOMSKY, 1965). Whether data and input data are similar to each other can be clarified by formal analysis, like set-theoretical tools or topology (BOURBAKI, 1961), (BOURBAKI, 2004), and semantic comparisons (R-Project). Here it is important to compare *sets* of data (TVERSKY, 1977). If we try, for instance, to compare only the facts 'Homer loves Jane' and 'Homer hates Bob' we will not make much progress. We should have sets of facts to go on, and even better, we should use some structural aspects for instance

[3] See (BALZER et al., 1987, Chap. 2).

a metric or some hypotheses, like 'X loves Y, then not (X hates Y)'. From similarity of sets and structures of data we can advance to similarity of sets and structures of numbers, to sets of sets of numbers (and so on), by using the many formal methods in mathematics, statistics and computer science (BALZER & ZOUBEK, 1995), (BOURBAKI, 1961), (R-Project). Additionally, similarity relations have components at the practical level. It is possible to bind hypotheses and program rules on social practices, see for instance (BALZER & TUOMELA, 2001).

Similarity between a model and a run is more difficult. A model is a set-theoretical entity; a run is a computer process. A model has no active component, whereas in a run often really new 'things' emerge or are created. However, such new entities also exist in a model. They are just not formulated explicitly. Derivable facts exist in a model, whether they are explicitly described or not. A set-theoretical model consists of *base sets* and *relations* so that a relation can be 'built' from the elements of the base sets (BOURBAKI, 2004, Chap. IV). A run can be seen as an ordered list of states, and each state as a collection of data. We can describe a given state as a database: a list of facts and/or rules. In a run, and relative to a given state of the run, new dynamic facts and results are generated – and sometimes deleted as well. In this way, we could define all the facts of a given state and use a one-to-one function to relate these facts to elements of the base sets of the relations of a model. Such elements can be described either by proper names or by atomic sentences 'belonging' to a modelled system.

Another problem is time. Is the linear order of the states of the run always isomorphic to a time structure which – in most cases – is a submodel of the model? We are not sure whether such a fit is always possible. Nevertheless, for a given simulation program this should be possible.

The most difficult problem of similarity comes from the non-constructive parts of a model. In a model it is not always possible to calculate a fact from other facts. Sometimes, the hypotheses are too strong: they 'escape' the deterministic methods. For example, in classical gravitation mechanics we cannot always solve the equations in a deterministic way. In a run, the computer deterministically calculates all the input data, the dynamic facts and the results – even if we use random numbers. The only solution that we can see is to use the notion of approximation of structures, see for instance (BALZER et al., 1987, Chap. 7). Granted the problem of fit of slices of time, we can assign the facts of one state bijectively to elements of a proper submodel of the time structure that is embedded in the model.

3 Data versus Input Data

The problem for a simulation system is data. In an empirical theory the data are measured by experiments, by surveys or by other scientific methods. Whether it is difficult or not, the researchers have to use those data which are obtained from a given real system. In social domains, some possible ways of getting such data are not allowed by operative moral system in force. If a method of determination is morally

possible, the method may still be too expensive.[4]

A further – and more 'normal' – problem is that often a study needs many data from one real system in order to verify even a weak claim. This can happen in many theories. In such cases a real system is only partially described and explained by a model. This is even more true for simulation systems. In our case, a simulation system can represent and describe only some few abstract and very small parts of a society.

For these reasons it makes sense to use the method of simulation. A 'pure' computer simulation uses only constructed or invented input data. Nevertheless, even in these pure cases this method makes sense. In other areas, simulation methods are used to enrich the existing empirical theories at hand, like in biology, in physics or in neuroscience. If a model seems to represent a real system, but some data are not available, it is often not difficult to produce a simulation program which supports the given model. The mixed, hybrid approaches in other more practical areas, like gaming and computer games, are of course very relevant. This mixed strategy is also used in social studies described, for instance, in the journal JASSS. We focus here on the pure case for two reasons. First, the constructive part of our paper would get complex, and of course lengthly, if we also describe mixed cases. Second, we have no means to get empirical data, see footnote 6.

In the structuralistic theory of science it is crucial to look at *sets* of real systems (intended applications) and *sets* of models to understand the content of a theory. It is not sufficient to investigate just one system and the corresponding model because the theoretical content of a theory would be missed. The content is 'symbiotically' related to the boundaries of the set of *successful* applications. The same holds for the boundaries of successful models.

This theme becomes even more central in pure simulation studies. For a simulation project it is not sufficient to produce one run of the program – whether the program executes simple runs or runs with repetitions. If one run is done – with or without repetitions – we should go for a *new* run in which a *different set* of input data is used to execute this new, different run. In other words, we use sets of different input data so that each new input starts a new run. Many examples can be found, e.g. (FENT et at., 2010), (PICASCIA & PAOLUCCI, 2010), (TAKAHASHI et al., 2010). Realistically speaking, we can say that *one* set of input data 'is', or corresponds, to *one* 'concrete' – real or only possible – system. In our case of societies, we use many different sets of input data so that each set of input data generates a 'version' of a society, which can be more or less far away from a real, intended society. Formulated differently, one run corresponds to one model, and a model and a run both correspond to a possible world, and at the level of reality, both the model and the run can be similar to the *same* real system – here: similar to the same society.

In this approach variability is essential. One run arises from one set of data, and the program rules. In addition, one run corresponds to a version of a society in one way or another. Therefore, a set of input data corresponds to a version of a society as well. Now the point is that different input data and the arising runs can correspond to the *same* real society. As a society is a complex entity it was – at least to date – not

[4] Alas, the real societies do not spend much money for social sciences.

possible to gather and to construct just one set of input data for one society. Currently, we apply the strategy of using a whole set of input data and the corresponding runs which represents one real society.

In this situation, we want to distinguish between a set of runs that makes sense 'for us' and a set of runs which represents nonsense societies, or more bizarre systems. We want to clarify this distinction and the corresponding boundary. In other words, we want to come closer to the central aspect of a society, to describe 'it's closure'. For this goal, we use a notion of *paradigmatic application* introduced by (KUHN, 1970), which we describe in a structuralist way. This paradigmatic method has also been used in the domain of concept formation (GÄRDENFORS, 1990).

This method can be described in an abstract way. A set X has a layered set of surroundings such that all surroundings are supersets of X. In a application of the paradigmatic method, X is a small set of examples that forms the basis of a topology. The paradigmatic method distinguishes a special surrounding E of X, which is maximal to a certain extent. This surrounding represents the set of all elements that are distinguished in this way. We depict an oval-shaped surrounding E_1 in Figure 3a) and a star-shaped surrounding E_2 in Figure 3b). The bases of these two surroundoings consist of the few points depicted within the surroundings. In b) we can see that the boundary of E_2 stems from a *Voronoi* tesselation (OKABE, 1992).

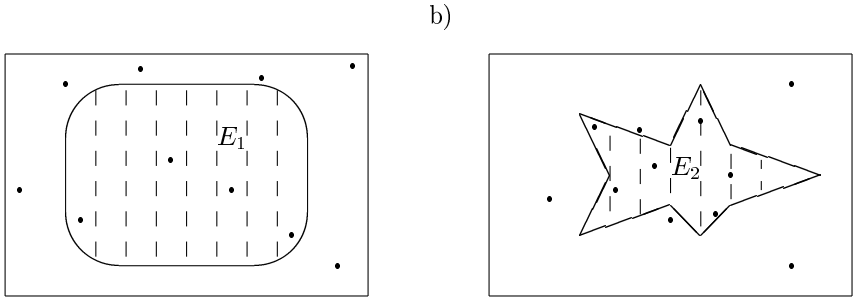

Fig. 3 Two forms of paradigmatic sets

In our applications is X a *set of* sets of input data. In Figure 3 we depict a set of possible sets of input data as a rectangle, some few, 'real' sets of input data (black points), and a surrounding in which points lying inside the surrounding. If we have a theory at hand, all the real sets of input data can be assigned to the intended applications of the theory. An intended application of a theory is successful if the hypotheses and the data fit into (or are approximately valid in) a corresponding model. In other words is an intended application successful if it can be embedded into a model. An unsuccessful application cannot be embedded in any model. If we link models to runs, and input data to intended applications, we can introduce successful runs and successful input data. A set of input data is successful if this set is similar to an intended application of the theory. We can explicate this similarity as follows. One can transform an application to a list of atomic sentences ('intended list'). The

same can happen for sets of results. Given a program, we can execute a run from a set of input data and obtain a set of results. If the input data are actually elements of the intended list then the set of input data is *successful*. In other words, we can say that a run is *successful* if both, the input data and the results, are found in an intended list stemming from a real intended application.

These notions can be extended to all sets ('potential' sets) lying in a given surrounding. A run executed by a set of potential input data is a *potentially successful* run. In other words, a boundary is introduced, and in this sense 'given', so that all potentially successful runs (and sets of input data) lies inside this boundary, and all other runs are found 'at the other side' of the boundary. In this way we can link successful run to successful models and successful sets of input data to successful intended applications. If we replace empirical data by constructed or generated input data, we move from a theory to a simulation system. The process of approximation does not change the picture.

Finally, it is important to work with complete systems of input data so that all possibilities can be found. We think this can only be done by using the strategy of starting with an idealized approach and gradually specializing and enlarging a simulation programm.[5]

4 The Methodology of Variation

We describe in an abstract way a systematic strategy of executing many different runs for different input data.[6] Very abstractly speaking this method can be seen a as variant of sensitivity analysis used in sociology (DEIF, 1986), (CHATTOE et al., 2000). One can vary a parameter or a hypothesis and see whether and how this change leads to different results. The method we describe here only alter parameters. Some examples are: (MAKAROV & BAKHTIZIN, 2010), (KAMINSKI, 2010), (KLINGERT & MEYER, 2010). If we change a hypothesis we do this only by modifying a parameter that is a part of the description of the hypothesis.

We assume that the *main program* for a simulation system is given. In Figure 4 we depict one run of the main program.

Starting the *main file* the essential constants for one run are loaded from the *file for constants*.[7] We call such a list a *system of world constants* (for one society). This main file uses the list of world constants to create a new set of input data with the help of the *file for creating data* – which is also loaded. The new input are stored in a newly generated file *stored data*, and are then loaded into the main file. The input data are now used. The essential part of the programm then executes these rules – in our case the rules for simulation a society. In this part, many results are stored in different files *results*1, ..., *results*N.

When the main program uses probability elements another loop is added in which the program depicted in Figure 3 is repeated several times. In each repetition the

[5] See (BALZER et al., 1987, Chap. IV and VIII.

[6] See www.munich-simulation-group.org for an example program.

[7] In Netlogo, for instance, these constants must be entered in a formated, standardized list, see the website http://ccl.northwestern.edu/netlogo/ .

same input data are used.

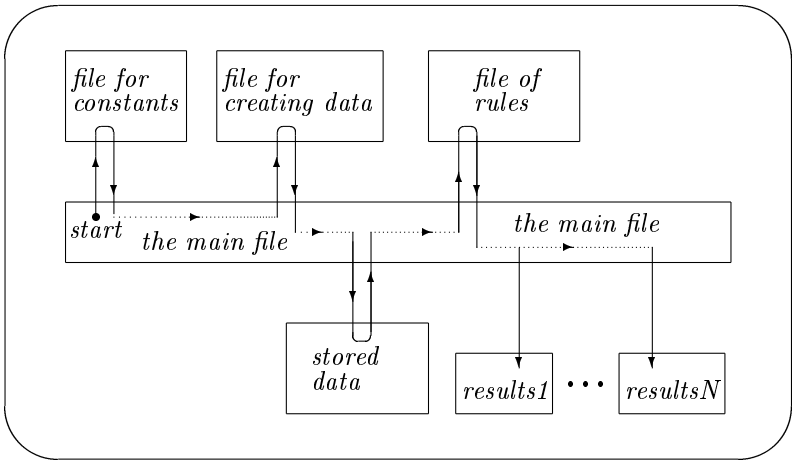

Fig. 4 Workflow of the main program

Now, the new feature is to add an uppermost *world-loop* in which different artificial societies – *worlds* – are executed. In Figure 4, a world-loop is depicted which starts at the left side of the workflow. Starting the world-loop, a file *autogen* is loaded, by which a series of runs is *auto*matically *gen*erated. A data item in *autogen* is a kind of prescription to generate a series of other data (see below). Each world has a number i, $i = 1, ..., k$, a 'name' $w(i)$, and a system of world constants stored in the *file of constants* enabling a main run to start (see Figure 3 and 4). The code for the world-loop generates a rather long *list* of systems of world constants. We store this list in a file *universal list for worlds*. One item, one system of constants for one world $w(i)$, is picked from this list, is 'unpacked', and the data are written in a newly generated file *stored constants for* $w(i)$ which we have seen already in Figure 3 (but there without relativization of a special world).

In each step i of the world-loop, i starts a run of one world, one society. In the main program the contents of the *file for constants* is replaced. The system of world constants used in step $i - 1$ is deleted and the 'next' item from the universal list is picked, transformed into a system of world constants for $w(i)$ and written to the *file for constants*. Additionally, the corresponding input data generated in step $i - 1$ are deleted, this must be done carefully. Then the 'new' main program starts in the same way as in Figure 3. Some files used in the main program of a world-loop must be indexed by the name $w(i)$ of the number i for the world which is just executed.

Finally we describe the code (see note 10) which generates the set of systems of world constants which is to put in the *universal list for worlds*. In the simplest case, a system of world constants is a list of components d_k, $k = 1, ..., s$, where the number s of constants from a system of world constants is the same in all systems of world constants. One component d_k can be a number α or a pair $[b_{below}, b_{upper}]$ of numbers

which represents an interval, i.e. b_{below}, b_{upper} are the boundaries, $b_{below} \leq b_{upper}$. A system of world constants is then a list $[d_1, ..., d_s]$. We transform this nested list into a flat list by deleting the brackets in pairs of the form $[b_{below}, b_{upper}]$. Such a flat list $[c_1, ..., c_r]$, $s \leq r$, can be generated as follows.

For each index j, $j \leq r$, we find a $domain_j$ in a pre-built *autogen. domain*. $domain_j$ is just a list of ordered numbers. For instance, if we use a system of world constants of the form $[\alpha_1, [\alpha_2, \alpha_3], \alpha_4, [\alpha_5, \alpha_6]]$ we use 6 $domains_j$, $j = 1, ..., 6$. $domain_3$, for instant, has the form $[\alpha_3^1, ..., \alpha_3^{u_3}]$. Mathematically formulated, we transform the lists $domain_j$ into sets $domain_j^*$, we take the product $domain_1^* \times ... \times domain_r^*$, and transform this product into a list of elements of this product. To generate such a list by a computer program we need a r-fold recursion.[8] For example, if we have 3 domains for constants: $[\alpha_1, \alpha_2, \alpha_3], [\beta_1, \beta_2, \beta_3, \beta_4], [\gamma_1]$, a list of all lists has the form $[[\alpha_1, \beta_1, \gamma_1], [\alpha_1, \beta_2, \gamma_1], [\alpha_1, \beta_3, \gamma_1], [\alpha_1, \beta_4, \gamma_1], [\alpha_2, \beta_1, \gamma_1], ...]$. One should roughly estimate the number of the systems of world constants and consider the specifications of the computer used for the simulation runs to ensure that the computation does not end with a stack overflow.

After this uppermost loop has finished, we can analyse the results in the usual way. Additionally, we can use statistical methods to compare different simulated societies. In this way, we can produce a tesselation, and pick one class of runs (or a few such classes) which we think are successful. 'For us', such runs correspond to societies. We can state boundaries so that the constants lie in areas where the results represent successful runs.

5 Conclusion

We look at simulation systems in the same way as we look at empirical theories. The main difference of these two entities is that in a pure simulation system the data can only be generated artificially whereas the data for a theory – in the pure case – exist always before the theory is applied. In this paper, we describe only the 'extreme', idealized cases. The extreme case could be embedded into 'normal' cases, where mixed, hybrid data are used in simulations and/or in empirical theories. However, this takes some additional effort. Our approach emphasizes the distinction between sets of data and sets of sets of data. In a simulation that uses purely generated data, the input data in one run are only used as a kind of variables.

We found four aspects of similarity between theories and simulation systems: similarity of 1) hypotheses and program rules, 2) data and input data, 3) a model and a run, and 4) the success of an intended application (and the corresponding models) and the success of a run. We claimed that a run is successful, if the three first aspects or criteria of similarity are met, and if furthermore the simulation results of the run were intended by the programmers.

[8] In our group a simple rule, the 'Urban-rule', is programmed under PROLOG, see www.munich-simlation-group.org.

References

BALZER, W., C. U. MOULINES, J. D. SNEED, 1987: An Architectonic for Science, Dordrecht.

BALZER, W., C. U. MOULINES, J. D. SNEED, eds. 2000: Structuralist Knowledge Representation, Amsterdam - Atlanta.

BALZER, W. & R. TUOMELA, 2001: Social Institutions, Norms and Practices. In: R. Conte, C. Dellarocas, eds.: Social Order in Multiagent Systems, Boston, pp. 161-180.

BALZER, W. & G. ZOUBEK, 1995: On the Comparison of Approximate Empirical Claims. In: W: E. Herfel et al., eds., Theories and Models in Scientific Processes, Poznan Studies 44, pp. 327-344.

BOURBAKI, N. 1961: Topology, Paris.

BOURBAKI, N. 2004: Theory of Sets, Berlin etc. (1. edition 1968).

BRENDEL, K. R. 2010: Parallele oder sequentielle Simulationsmethode? Implementierung und Vergleich anhand eines Multi-Agenten-Modells der Sozialwissenschaft, Verlag Utz, München.

CHATTOE, E., N. J. SAAM & M. MÖHRING, 2000: Sensitivity Analysis in the Social Sciences: Problems and Prospects. In: R. Suleiman, K.G. Troitzsch, N. Gilbert, eds.: Tools and Techniques for Social Science Simulation, pp. 243 -273.

CHOMSKY, N., 1965: Aspects of Theory of Syntax. MIT Press. Cambridge Mass.

DEIF, A. S. 1986: Sensitivity Analysis in Linear Systems, Springer, Berlin.

DIEDERICH, W., A. IBARRA, T. MORMANN, 1989: Bibliography of Structuralism. In Erkenntnis 30, pp. 387-407.

DIEDERICH, W., A. IBARRA, T. MORMANN, 1994: Bibliography of Structuralism. In Erkenntnis 41, pp. 403-418.

ERNST, A. & S. KUHN, eds. 2010: Proceedings of the 3rd World Congress on Social Simulation WCSS2010 (CD-ROM), Kassel, Germany: Center for Environmental Systems Research, University Kassel.

FENT, T., B. DIAZ & A. PRSKAWETZ, 2010: Family Policies and low Fertility: How does the Social Network influence the Impact of Policies. In: Ernst A. & S. Kuhn, eds.

GÄRDENFORS, P. 1990: Induction, Conceptual Spaces and AI, Philosophy of Science 57, pp.78 - 95.

GILBERT, G.N. & R. CONTE 1995: Artificial Societies: The Computer Simulation of Social Life, London.

HOFMANN, S. 2009: Dynamik sozialer Praktiken, Wiesbaden, 2009.

KAMINSKI, B., M. LATEK, M. Jakubczyk, 2010: Measuring the Impact of Workforce Sickness on Economic Output Controlling for Technology and Epidemiology. In: Ernst A. & S. Kuhn, eds.

KLINGERT, F. & M. Meyer, 2010: Multi-Agent-Simulation of Prediction Markets:

Does Manipulation Matter using Zero-Intelligence Traders? In: Ernst A. & S. Kuhn, eds.

KUHN, T. 1970: The Structure of Scientific Revolutions, Chicago.

MAKAROV, V. L. & A. R. BAKHTIZIN, 2010: Agent-Based model for simulation of terrorism in Russia's Causasus. In: Ernst A. & S. Kuhn, eds.

OKABE, A., B. BOOTS, K. SUGIHARA 1992: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, New York.

PICASCIA, S. & M. PAOLUCCI, 2010: Diffusion of culture and the "PageRage effect". In: Ernst A. & S. Kuh, eds.

PITZ, T. 2000: Anwendung Genetischer Algorithmen auf Handlungsbäume in Multiagentensysteme zur Simulation sozialen Handelns, Verlag P. Lang, Frankfurt.

R-PROJECT: www.R-project.org.

STEGMÜLLER, W. 1976: The Structure and Dynamics of Theories, Springer, New York.

TAKAHASHI, H., S. Takahashi, T. Terano, 2010: Analyzing the influence of fundamental indexation of financial markets through agent-based modeling. In: Ernst A. & S. Kuhn, eds.

Tversky, A. 1977: Features of Similarity. In Psychological Review 84, pp. 327 - 52.

WILENSKY, U. 1999: NetLogo. http://ccl.northwestern.edu/netlogo/ Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.