A Simple Frame for Artificial Societies

Wolfgang Balzer and Karl R. Brendel and Solveig Hofmann

University of Munich
Munich Simulation Group
balzer@lrz.uni-muenchen.de,
karl.r.brendel@t-online.de,
hofmannsol@aol.com

Abstract. We present a framework for Artificial Societies (AS) in a Prolog programming environment and sketch four modules. The *language module* consists of a programming language, the *main program*, that simulates an AS, and the rules for each idealized *actor*, that are present within the main program. The *subsystems module* programs relations in special, social groups, the *basic module* programs basic non-linguistic entities and relations concerning all actors, and the *computer module* the structural processes of the actors. We describe in some detail, how we connect the programming language for an AS with the 'languages' of the actors.

The main goal of the longstanding project of our Munich Simulation Group is to develop the rudimentary structure of a computer program which describes and simulates real societies, like British, French, German, or Indian societies, in a very coarse-grained way. At present we work on smaller projects: social practices [9], parallel aspects [5], economic models, crises models [17], evaluation systems and state building, which can be embedded in our main frame. Other frames are on the one hand more general, they can be applied also to complex systems of other kinds. On the other hand they are more special in emphasizing modularity and speedy execution. See, for instance [12].

1 The Language Module

In our main program we use the type free programming language Prolog for simulating an AS. Additionally we use it as language for the actors of the AS. A Prolog program is primarily not designed for a maximum execution speed, but it can easily be written and read by people with a Latin-style language background.

The language of an AS contains *types* ([6], Chap. IV.), like $\langle 2, 2, \langle 6, 5, 1, 2\rangle\rangle$; *phrases*,[1] like *nick*[2] or '*the_war_1914-1918*'; *predicates*, like *go, perceive, discuss, produce, generate*; *sentences*, like '*sleep*(*nick, in_bed*)', '*discuss*(*rose,with_nick*)'; and in complex applications *variables, formulas* and *terms*. We call phrases and sentences *expressions*.

The structure of the main program is simple. It is written for sequential

[1] The set of phrases, particularly, contains words and names, too.

[2] In Prolog, all terms with a capital letter at the beginning are variables. See e.g. [6]. Phrases ('predicates') and sentences, in contrast, always start with a lowercase letter.

computers.(?) This program consists essentially of layers of loops, where the central loop handles the *ticks* (points in time, written here: $t, t_i, t+1, t+2, ...$).

A part of the programming language is used to represent the idealized *actor-languages*. Each actor of an AS uses an individual set of phrases and grammatical rules, and stores 'his' sentences in 'his own' memory. Thus, an actor-language can contain idiosyncratic expressions. In a first step, we use for an actor-language just the grammar of the programming language.

In more detail, we divide the set of expressions e of the main program in four groups. An expression-1 e is a phrase or a sentence, that is stored in the memory of an actor at a certain time, relative to an AS. An expression-2 results from an expression-1 e, by wrapping e in the form of $mess(t, a, \_)$. $mess(t, a, e)$ means, that the expression e is sent or received as a *message* by the actor a at time t. An expression-3 originates from an expression-1 e, by wrapping e in the form of $fact(t, a, \_)$, see e.g. [2], Appendix, and [9], pp. 95. $fact(t, a, e)$ denotes, that the expression e in the AS has the status of a *fact*, that is perceived or generated by the actor a at time t. The forth kind of expressions is only used by the programmers of the main program. We call expressions-2 *information elements* (*relative to an* AS) or *inels* for short. These *inels* are essential to our approach. We call expressions-1 *the internalized inels of an actor*, and expressions-2 *messages*. In Prolog, we therefore represent the distinctions of these four groups of expressions just syntactically.

2 The Society-Wide Entities and Relations

From the sequential ticks we define the relations of *posteriority* $\prec$, and of *periods*, relative to action-types. Physical space is represented by means of 3-dimensional

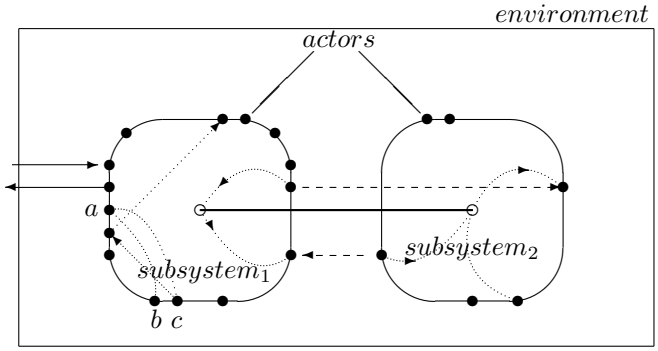

Figure 1: Artificial Society with two subsystems.

coordinates. The *events* $(e, e_i, ...)$ are used in Prolog as phrases in a multitude of variants such as 'sunrise143' or 'a_war' but also as sentences like 'dies(nick)'. We also subsume the 'normal' objects, like 'tree' or 'men' as 'borderline cases' under the events. In a puristic, philosophical sense an action is a special kind of event. At the programming level, however, we certainly treat actions as an own kind

of entity. Depending of the kind of action we synchronize actions in different ways. We are aware of the problem of real synchronous actions in a sequential computer. Usually we program action loops asynchron(ous?), relative to a loop (or several loops). For example, if an action consists of haggling about a costly good the time-point for the beginning of the action is defined, so that the action can go on for a certain period. During the action of haggling an actor can begin to perform a second action, for instance, to react to an order independently of the ongoing haggling. The main program decides in a rather crude way which of two actions has priority.

The main entities in an AS are the *actors* $(a, a_i, ...)$ and their 'humanoid' characteristics and relations, that are required for the reproduction of the AS, such as an 'existence predicate' ($exi(t, a, 1)$: a exists at time t), components for aging, birth, death, sex and the generation of new 'children'.

An important component of an AS contains the information elements called *inels*. On the language level, *inels* can be words, phrases, sentences, variables, formulas and terms.

In an AS, we describe two common relations between the actors and the environment inherent to the AS that can be specialized in different ways. The first relation expresses – in a very general sense – the perception of an actor (*perceive*). For instance, an actor can perceive a tree, a sunrise, a discussion or a war. This relation transforms objects, events or actions into *inels*. On the other hand, the relation of generation (*generate*) is an important relation, by which an *inel*, that belongs to the internalized expressions of an actor, is 'changed' or transformed into an entity, which is part of the actor's environment. An actor can transform an *inel* into an action or into an event. If the generated result is an action, the *inel* i has to be transformed in a two step process. First i becomes an *intention*, and subsequently it becomes an action. An actor can e.g. 'drop a stone from a tower' (that is a simple event), 'start a discussion with another actor' (this is an action), but also — in a borderline case — 'construct a table'. Additionally, we subsume the automatic 'reflexes' of an actor under the generation relation.

In the basic module we differentiate between four types r^τ of individual relations that get filled with content in the subsystems of the AS. For an individual relation r^τ, we highlight the *main actor* a of relation r^τ. If the main actor a in r^τ is related to an *event*, we write r^{ev}, if a is related to an *action* we write r^{ac}, if a has an *internal* relation in a given subsystem we write r^{int}, and if a 'lives' in a given subsystem and if he has an *external* relation to another subsystem we write r^{ext}. With regard to contents, we list here some examples of different individual relations belonging to the subsystem-module: 'The main actor a listens to music' is an event-relation r^{ev}; 'a marries a'' is an action-relation r^{ac}; 'a marries a', where a and a' both are diplomats' is an internal-relation r^{int} in the political subsystem; 'a marries a', whereby a is a politician and a' is an economist' is an external-relation r^{ext} (both actors stem from different subsystems). Needless to say, these different relation types can overlap. For instance, a relation r^{int} can simultaneously be a type r^{ac} relation.

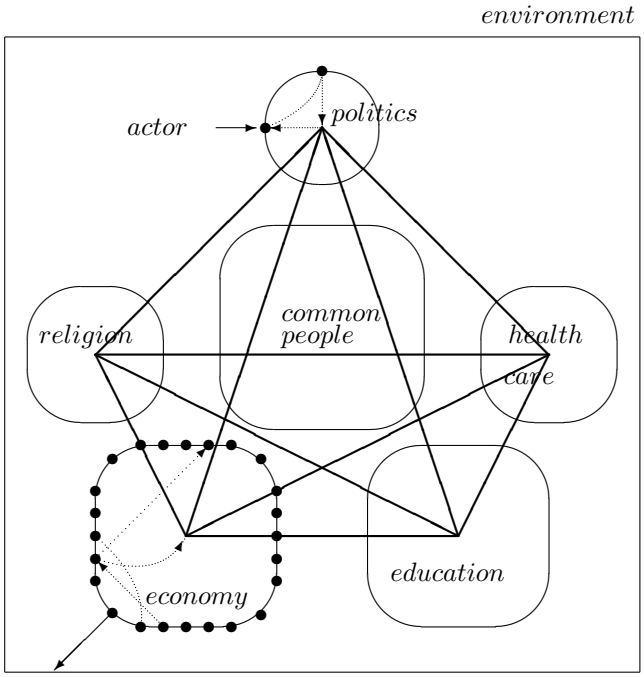

Abbildung 1: The Subsystems of an Artificial Society.

Two subsystems are depicted (Fig. 1) in the 'small world style',[3] and the filled circles are some actors in these subsystems. A dotted line represents an individual relation. An arrow shows the direction of the relation; a line without an arrowhead indicates the lack of direction. There are additionally relations that require more than two actors, like a, b, c. There are relations ('normal arrows') from an actor to the environment, and *vice versa*. The external relations are shown as a bundle. Thus, an external relation runs along the bold faced line.

3 The Subsystems of an AS

Because we are not affiliated to any specific sociological field,[4] we introduce some empirically explorable subsystems, which we don't want to fill with sociological or other content. Additionally to physical space we use several social spaces relative to subsystems in the form of graphs and cellular orderings.

We assume five particular subsystems (Fig. 2) based upon a large main population which we call '*common people*' due to lacking a more appealing name. The biggest particular subsystem of an AS is the *economy*, that in turn consists of employers and the related enterprises (like firms, organizations, or

[3] See [16], and the corresponding basic approach of NetLogo [14].

[4] We abstain from referencing the sociological 'classics' like Bourdieu, Coleman, Durkheim, Giddens, Luhman and Parsons in detail.

holdings), of farmers (and farms, kolchozes, or industrial plantations), and of sellers (and shops, warehouses, or eBay).

Another subsystem, *education*, is on the actor-level mainly built of a) children and parents, b) pupils and teachers, c) students and professors, and consists on the institutional level of the corresponding facilities, like kindergarten, schools and universities. *Religion* forms the next subsystem. Here the actors are the people of faith and the priests. The associated institutions embrace rituals for the different phases of an actor's life such as initiation rite, marriage ceremony, seasonal celebrations and funeral service. The *health care* subsystem contains sick persons and doctors or healers with the appertaining institutions, like hospitals or medical practices.

Finally, the mightiest subsystem – currently in most of the real societies – is *politics*. Here we can find the government (including the opposition) together with the often existing subsystems of the legislature, the executive authority and the law, as well as the public members that are frequently engaged as voters.[5]

4 The Computer Module

The parallel computer systems that we were already using 15 year ago, the Transputer systems, see [5], [3], are unfortunately no longer available. Currently we are forced to work with single processor computers in which the actors are processed in a sequential way. Hence, a spatio-temporal separation of subprocesses that should be associated to a specific actor is hardly possible. In addition to the main program, some generated and some downloaded tools are concurrently used. In our mind, the main computer of an AS is a system of independent actor-computers connected by wires. Currently, we can represent this architecture only within the program code.

So we write the code in such a way, that every actor a is structured according to (Fig. 3). An actor a has a *processor* with a *memory*, a *mailbox*, and four *ports*. He receives 'direct' messages from other actors through the port p_1 and sends 'direct' messages to some or all actors through the port p_2. Through the port p_3 a receives items of perception, that are no direct messages from other actors. These perceptions get converted into *inels* by the *transformer*.[6] The incoming *inels* from the mailbox and from the transformer are sequentially ordered by the *sequencer* and forwarded to the processor. Every processor output – an *inel* – is passed through the sequencer. The *inel* can directly be sent to some or all other agents via the port p_2, or can be converted through the transformer, and the port p_4 into an 'event' or into an 'action', or can be forwarded to a 'training loop'. Using the *generate*-port p_4 we see processes, that goes to the outside world of the actor and are partially generated by internalized *inels* of

[5] In this arrangement, the army of an AS is part of the executive authority. In a different conception the army could be made up in an own subsystem. Religion can be combined with health care and education into a single subsystem named *knowledge system*.

[6] Viewed realistically, some incoming sensory data get converted to brain signals. Of course, a computer lacks this somatic level.

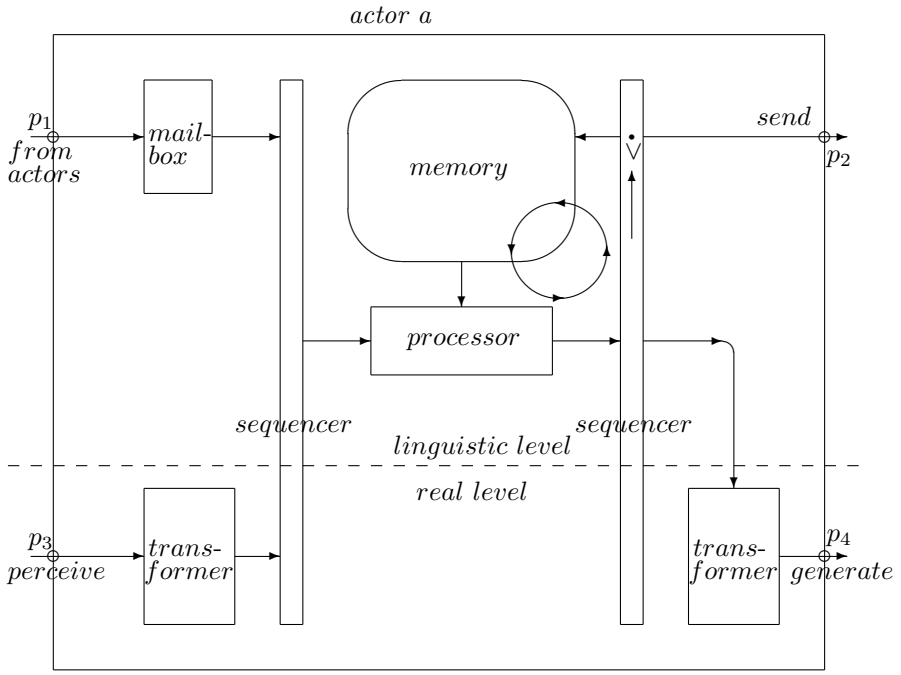

Abbildung 2: The Computer Module.

the actor. The transformer at first generates an *intention* which exists internal to the actor.

Using the port concept, we distinguish two different levels. On the upper level, *inels* are directly sent to other actors and received by the other actors. Viewed realistically, we call this the *linguistic* level. On the lower level, information is forwarded through events and actions to the outside world of the actors. The events and actions have to be perceived by the actor and have to be transformed into *inels*. Complementary to that, *inels* are transformed into actions that get transferred to the outside of an actor. In a simulation, this two levels can only be distinguished in a syntactical and computational way.[7]

5 Future Prospects

We mention at least three main topics for the future. Within the actor's memory it is permitted to use names which are related to one and the same actor (or to parts of that actor) in an reflexive way. E.g. the actor *nick* currently memorizes the sentence *thinks*(*nick*). With regards to content, this leads to reflexivity or circularity that can also be found in real life. We try to limit this circularity in

[7]See e.g. the SONATA model in [8] as a complex example of this functional separation of an agent's perceptions, actions and cognitions.

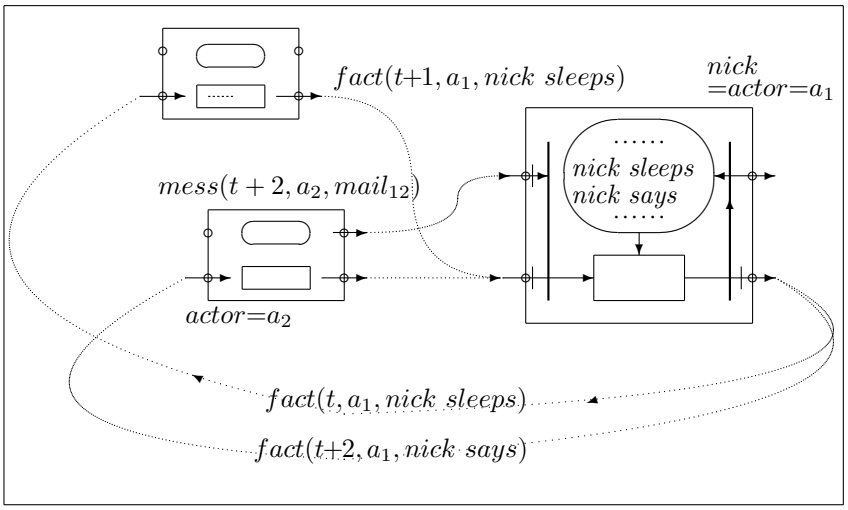

Abbildung 3: The Four Levels of an Artificial Society.

our approach by syntactically differentiating between the internalized *inel*s, the facts and the messages.

Furthermore, we currently neglect the fact that the actor's environment in an AS imposes many additional influences upon an actor. Many aspects, already existing in computer games, could be adapted to our frame.[8]

Finally, we have to postpone the study of temporal and/or local parallel actions, that are reaching their physical – and other – limits, until a future generation of computers are available. However, some of the aspects can also be implemented in a sequential way. A good example is [7].

Literatur

[1] Balzer, W.: A Basic Model for Social Institutions. Journal of Mathematical Sociology. 16, 1–29 (1990)

[2] Balzer, W.: SMASS: A Sequential Multi-Agent System for Social Simulation. In: Suleiman, R., Troitzsch, R. G., Gilbert, N. (eds.) Tools and Techniques for Social Science Simulation, pp. 65–82, Heidelberg (2000)

[3] Balzer, W., Brendel, K. R.: DMASS: A Distributed Multi-Agent System for Simulation in Social Systems (1996), www.lrz-muenchen.de/ W.Balzer/BALZER.html

[8] See e.g. the very popular computer game 'The Sims' or 'Haunt 2' described in [10].

[4] Balzer, W., Brendel, K. R., Hofmann, S.: Künstliche Gesellschaften. Facta Philosophica. 10, 3–24 (2008)

[5] Brendel, K. R.: Parallele versus sequentielle Multi-Agenten-Simulation als Methode der Sozialwissenschaft. Ein Vergleich anhand eines Solidaritätsmodells, Dissertation, München (2008)

[6] Clocksin, W. F., Mellish, C. S.: Programming in Prolog, Berlin (1987)

[7] Deguchi, H., Tanuma, H., Shimizu, T.: SOARS: Spot Oriented Agent Role Simulator – Design and Agent Based Dynamical System. In Proceedings of the Third International Workshop on Agent-Based Approaches in Economics and Social Complex Systems (AESCS04), pp. 49–56, (2004)

[8] Ernst, A., Krebs, F., Zehnpfund, C.: Dynamics of Task Oriented Agent Behavior in Multiple Layer Social Networks. In Takahashi, S., Sallach, D., Rouchier, J. (eds.) Advancing Social Simulation: The First World Congress, pp. 319–330, Springer, Berlin (2007)

[9] Hofmann, S.: Dynamik sozialer Praktiken, Wiesbaden (2009)

[10] Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., and Stokes, D.: AI Characters and Directors for Interactive Computer Games. In Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference, San Jose, California, pp. 877–883, (2004)

[11] Plikynas D.: A Social Field Model: Premises for an Artificial Intelligence Based Simulation of Human Society. In [13] pp. 135–142

[12] Polhill, J. G., Nicholas M., Gotts, N. M.: A new approach to modelling frameworks. In [13] pp. 215–222

[13] Terano, T., Takahashi, S., Sallach, D., Rouchier, J. (eds.) Proceedings of the 1st World Conference on Social Simulation, Vol. 1, Kyoto (2006)

[14] Tsvetovat, M., Carley, K. M.: Simulating Social Systems Requires Multiple Levels of Complexity. In [13] pp. 231–238

[15] Wilensky, U.: NetLogo. http://ccl.northwestern.edu/netlogo/ Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL (1999)

[16] Wilensky, U.: NetLogo Small Worlds model. http://ccl.northwestern.edu/netlogo/models/SmallWorlds Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. (2005)

[17] Will, D.: Krisensimulation mit abstrakten Handlungstypen: Ein neuer, methodischer Ansatz, Dissertation LMU München. (2000)