# Iterated Mutual Observation with Genetic Programming

**Peter Dittrich**[(1)], **Thomas Kron**[(2)], **Christian Kuck**[(1)], **Wolfgang Banzhaf**[(1)]

[(1)]University of Dortmund
Department of Computer Science
D-44221 Dortmund
Germany
ls11-www.cs.uni-dortmund.de

[(2)]University of Hagen
Department of Sociology
D-58084 Hagen
Germany
www.fernuni-hagen.de/SOZ/SOZ2/Kron

**Abstract**

This paper introduces a simple model of interacting agents that learn to predict each other. For learning to predict the other's intended action we apply genetic programming. The strategy of an agent is rational and fixed. It does not change like in classical iterated prisoners dilemma models. Furthermore the number of actions an agent can choose from is infinite. Preliminary simulation results are presented. They show that by varying the population size of genetic programming, different learning characteristics can easily be achieved, which lead to quite different communication patterns.

## 1 Introduction

Mutual observation is - next to influence and negotiation - a mechanism to promote the formation of social structures (Schimank (2000): 207ff.). One may say that mutual observation is *the* primary factor of producing social order without given norms, rulers, or actors with rational intentions. By mutual observation the problem of "double contingency" (Parsons 1937; Parsons 1968; Parsons 1971; Luhmann 1984; Kron and Dittrich 2001) can be solved, that is to establish structures of mutual expectations, to which the actors orientate mandatory: social structures as results of unintended actions.

Mutual observation occurs if actors have "intention-interferences", that means that the actors are interdependent with regard to resources they are both interested in. In such a situation *before* they strive for cooperation they often observe each other, that is they tend to *co-orientation* and try to predict the other's action in order to select a most rational action.

This paper introduces a simple model of interacting agents that learn to predict each other. For learning to predict the other's intended action we apply genetic programming (GP) (Koza 1992; Banzhaf, Nordin, Keller, and Francone 1998). The strategy is fixed and does not change like in classical iterated prisoners dilemma models (Lindgren and Nordahl 1994; Hofbauer and Sigmund 1998). The game itself is also not changed over time (see, e.g., (Akiyama and Kaneko 2000) for a dynamically changing game). Our model is similar to a model introduced by Taiji and Ikegami (1999), but they use dynamical recognizers (Pollack 1991) instead of genetic programming.

We built the model to tackle the following questions:

1. **Evaluation of technology**
   Does GP provide a reasonable technique to model the ability of generalization of an actor? What kind of beneficial properties posses GP? How should the parameter be chosen? How fast does an actor learn? How do the evolved programs look like?

2. **Behavior of mutually observing agents**
   What kind of dynamic behaviors can appear when agents try to learn and predict each other behavior and use this prediction to act optimally?

We do not give answers to these questions, here, but define the model and present first simulation results.

## 2 Experimental Setup

The basic setting consists of two players (agents) and an environment (Fig. 1). A turn (action cycle) consists of the following steps:

1. **Update environment** $U$
   The environment is updated. We will consider two simple variants.
   (a) The environment is set to a constant value, $U = 0.5$.
   (b) The environment is set to a random value, $U = randReal(0, 1)$.

2. **Predict other player**
   Both players use their generalizer to predict the action of the other player. This is denoted by

$$B' = f_A(U), \quad A' = f_B(U). \tag{1}$$

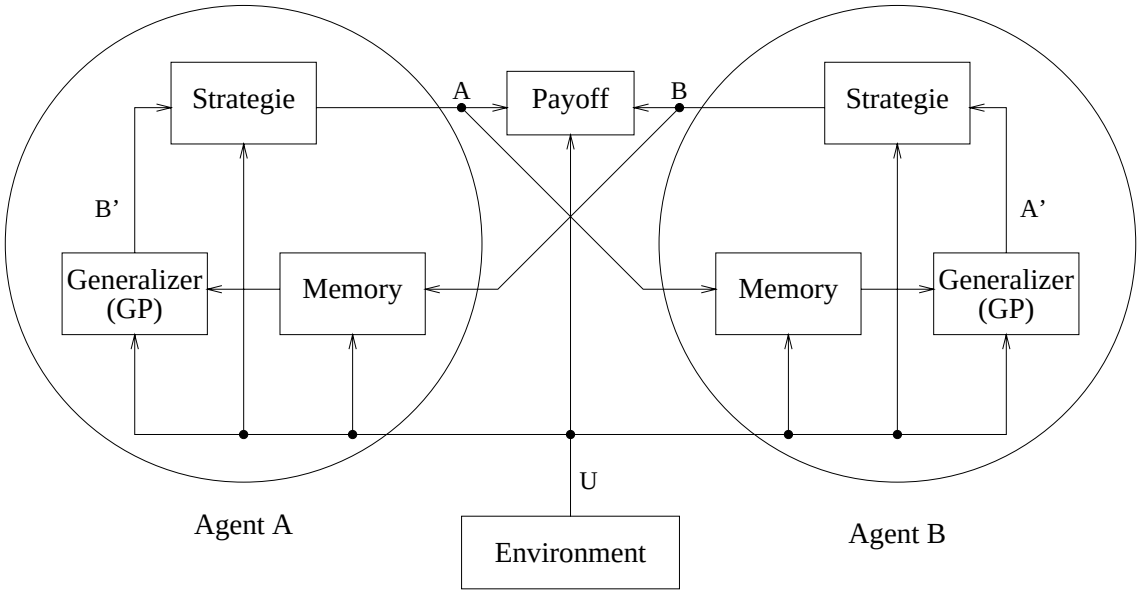


Figure 1: *Basic setup with two players. An arrow represents a scalar value between 0 and 1. The* memory *of player A or B stores environment-action pairs* $(U, B)$ *or* $(U, A)$*, respectively. A memory is implemented as a table.*

3. **Chose action (activate strategy)**
   Both players activate their strategy to compute their action. The strategy takes the current state of the environment and prediction of the other player as its inputs. The action is chosen such that the expected payoff would be optimal if the prediction of the other agent's behavior is correct. Because here we are using a simple environment, there is always only one such optimal action.

4. **Calculate payoff**
   The payoff is calculated and logged for analysis. There is no feedback of the payoff because the strategy should not be learned here. We assume an "optimal" strategy (see below).

5. **Update memory**
   The players store the action of the other player and the environment status in their memory. Note that in this simple model only the action of the other player is stored.

6. **Update generalizer (activate GP system)**
   Both player run their GP system to generate new prediction functions $f_A$ and $f_B$, respectively. The GP system uses the most recent $m$ memory entries as fitness cases. We call $m$ *memory size*.

The payoff function $h_A$ for agent $A$ is defined as

$$h_A(A, B, U) = \int_{A-r}^{A+r} \begin{cases} g(U, x)dx & \text{if } |x - B| > r, \\ \sigma g(U, x)dx & \text{otherwise,} \end{cases} \tag{2}$$

where $A \in [0, 1]$ is the action performed by Agent A, $B \in [0, 1]$ is the action performed by Agent B, and $U \in [0, 1]$ is the current state of the environment. There are two constant parameters: the **synergy factor** $\sigma$ and the **payoff radius** $r$. A synergy factor greater 1 means that both players get higher reward if they perform the same action. If $\sigma$ is smaller than 1, the same action of both players would cause a reduction of the payoff by a factor of $\sigma$, roughly speaking. The **kernal function** $g : [0, 1] \to \mathcal{R}$ is an auxiliary function and defined here as

$$g(U, x) = \frac{0.01}{(U - x)^2 - 0.01}. \tag{3}$$

Figure 2 shows a plot of the kernal function for environment $U = 0.5$.

To illustrate how the payoff is calculated an example is given: Assume both agents perform the same action $A = B$ and the environment is also $U = 0.5$ then the payoff for Agent A becomes:

$$h_A(A, B, U) = \int_{A-r}^{A+r} \sigma g(0.5, x)dx \tag{4}$$

This is exactly the area underneath the curve shown in Fig. 2 from $A - r$ to $A + r$ multiplied by $\sigma$. If both agents perform totally different actions such that $|A - B| > r$, their payoff is equal to the corresponding area without multiplication by $\sigma$. When the agents perform similar actions, only the "overlapping" area is multiplied by $\sigma$. For $\sigma = 0.5$ this results in sharing the payoff stemming from the "overlapping" area.

## 3 Results

In this section we show preliminary simulation results for the most simple case, namely a static environment.

### 3.1 Experiments with Static Environment ($U = 0.5$)

For these experiments the environment does not change. Furthermore, if both agents perform the same (or a similar) action, the payoff will be narrowed. This is achieved by setting the synergy parameter $\sigma$ to 0.05. The parameters are set as:

| name | value | note |
|---|---|---|
| $U$ | 0.5 | environment |
| $r$ | 0.05 | payoff radius |
| $\sigma$ | 0.05 | synergy factor |
| $m$ | 10 | memory size |

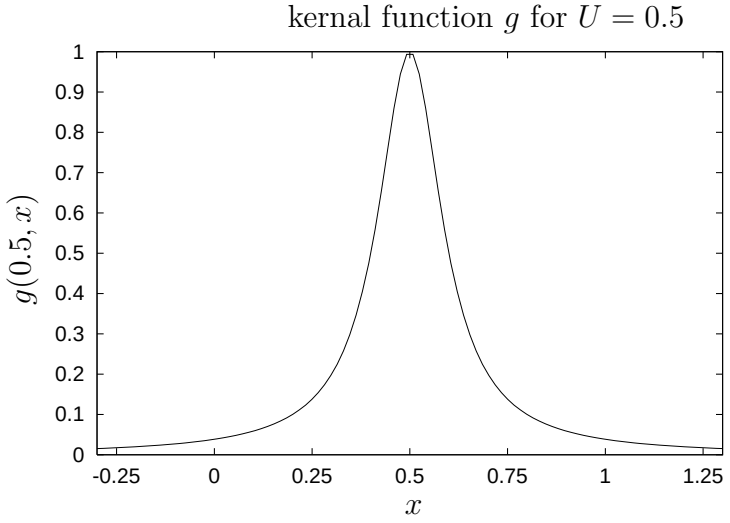


Figure 2: *Kernal function* $g(U, x) = 0.01/((U-x)^2-0.01)$ *for environment* $U = 0.5$.

Figure 3 shows the action sequence of three experiments where the population size of the GP system is set to $M = 20, 40, 100$, respectively. Note that the population size $M$ is a parameter of the GP system. $M$ specifies how many individuals (programs representing $f_A$ or $f_B$, respectively) are stored at once during the evolutionary search process in order to find a good prediction function. The population size $M$ should not be confused with the number of agents.

The population size influences the prediction ability of the agents. A small population size does not allow to evolve a good prediction function. For $M = 20$ (bad prediction ability) the actions are fluctuating strongly. For $M = 100$ (good prediction ability) the situation stabilizes. For an intermediate population size $M = 40$ (fair prediction ability) we observe an intermittent behavior. Phases of "stable cooperation" are disrupted by strong fluctuations.

In the simulation shown in Fig. 3 for $M = 40$ the unstable phase starting at Turn 56 is caused by a wrong prediction of Agent A. The prediction in addition to the action sequence is shown in Fig. 4.

Figure 5 shows the time evolution of the obtained payoff of one agent for different population sizes of the GP system. We can see that the average payoff increases with increasing population size. So, better learning and generalization ability leads to higher payoff. This observation is also supported by Fig. 6. The figure compares the payoff obtained by Agent A and B. During a stable phase of cooperation both agents receive high payoff; but not necessarily the same quantity.

### 3.2 Visualization of the Generalizer

The current state of agent $A$ is represented by its memory and the generalization function $f_A$. To visualize this state we can just plot function $f_A$ over the environment

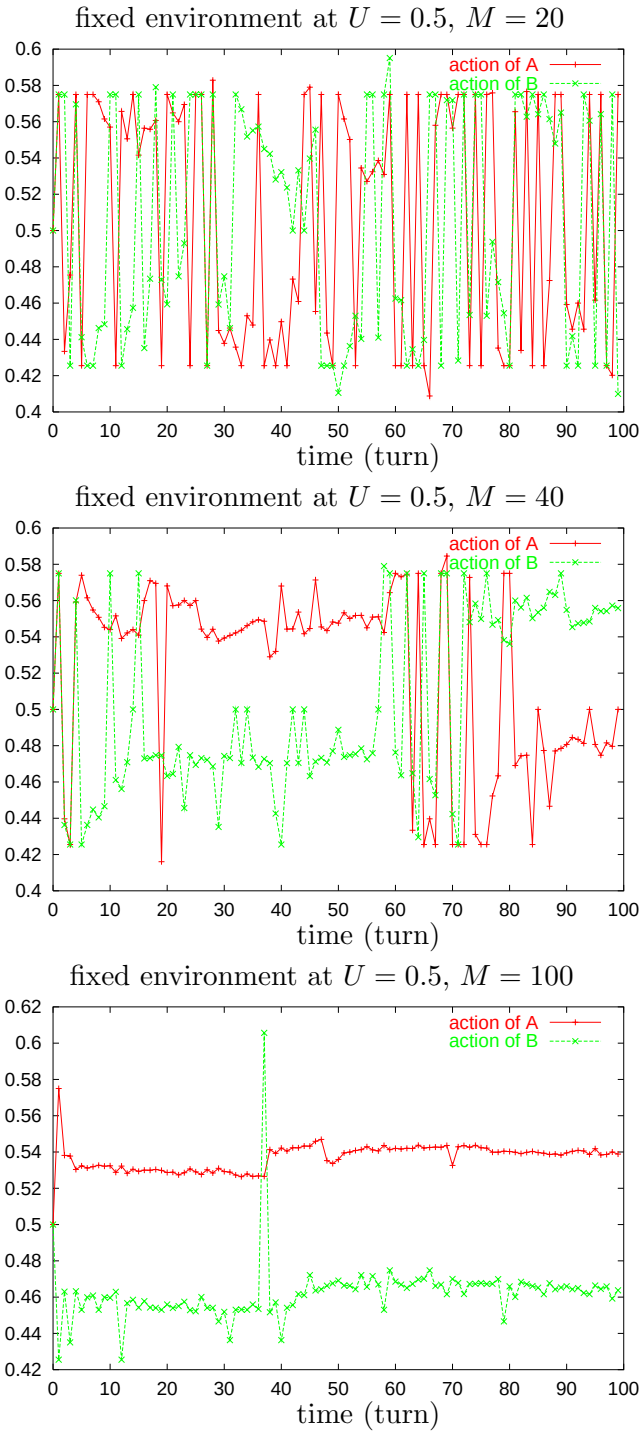


Figure 3: *Action of player A and B over time. GP system population size* $M = 20, 40, 100$. *Parameters: Environment* $U = 0.5$, *memory size* $m = 10$, *synergy factor* $\sigma = 0.05$, *payoff radius* $r = 0.05$.

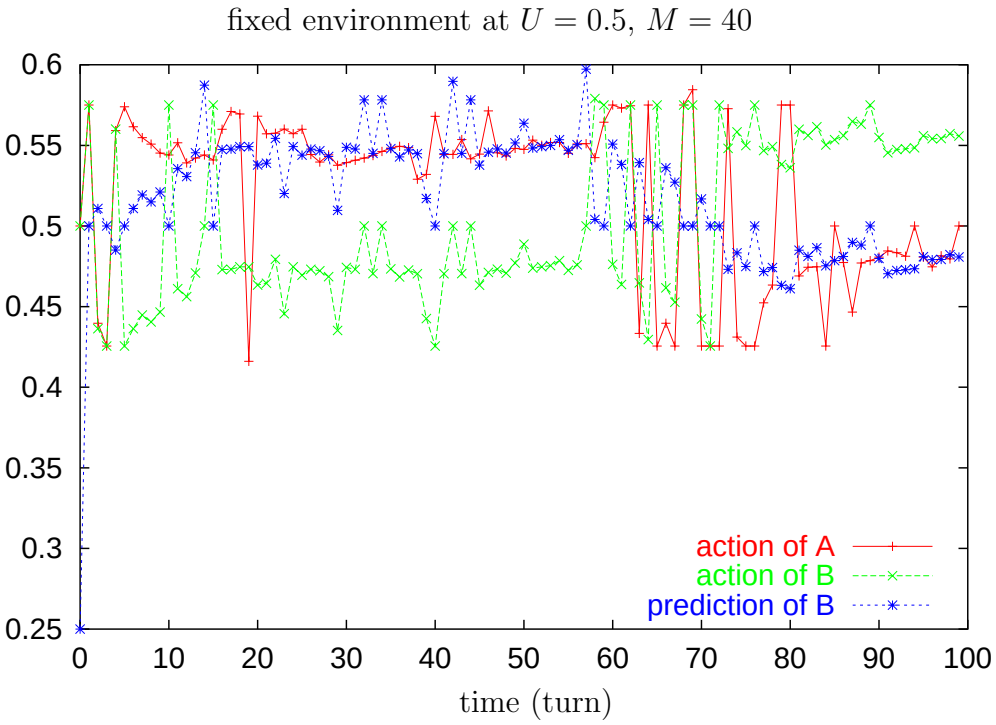


Figure 4: *Prediction of Agent B is shown in addition to the action sequence of Agent A and B. Same run as in Fig. 3.*

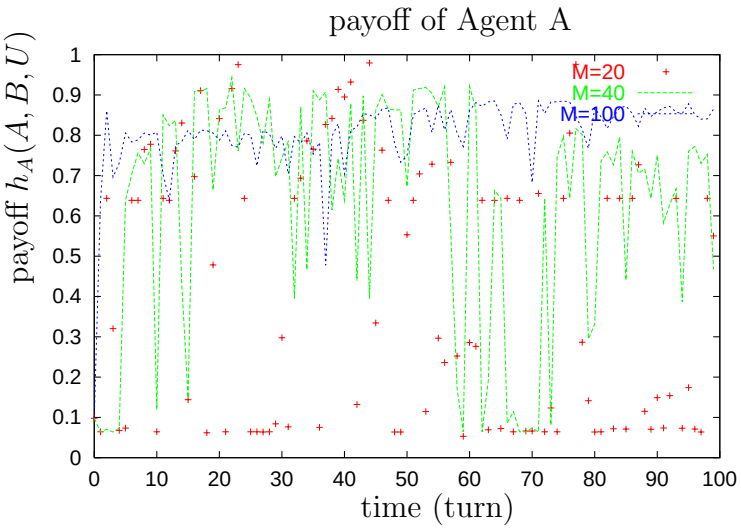


Figure 5: *Evolution of the payoff for population size $M = 20, 40, 100$ of the genetic programming system. Same runs as in Fig. 3.*

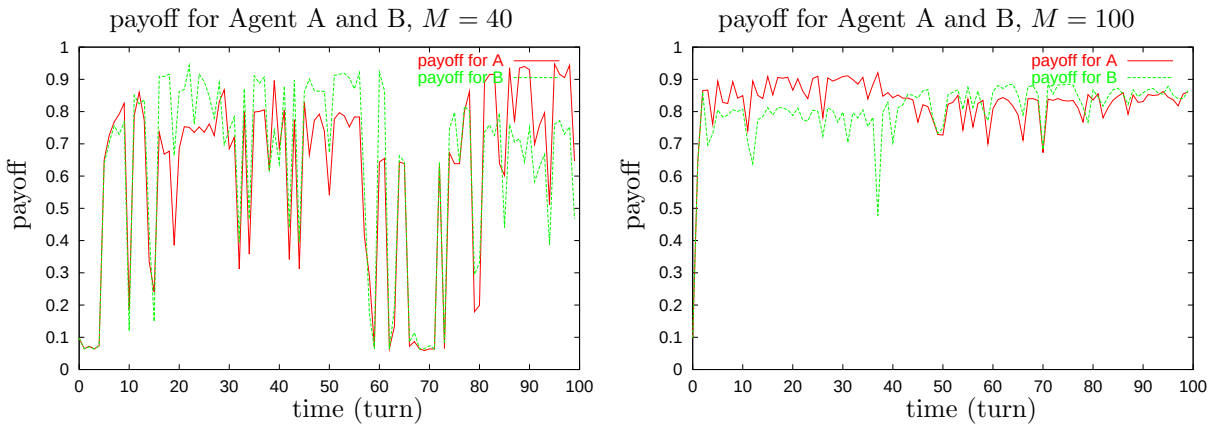


Figure 6: *Evolution of the payoff for Agent A and B. Same runs as in Fig. 3*

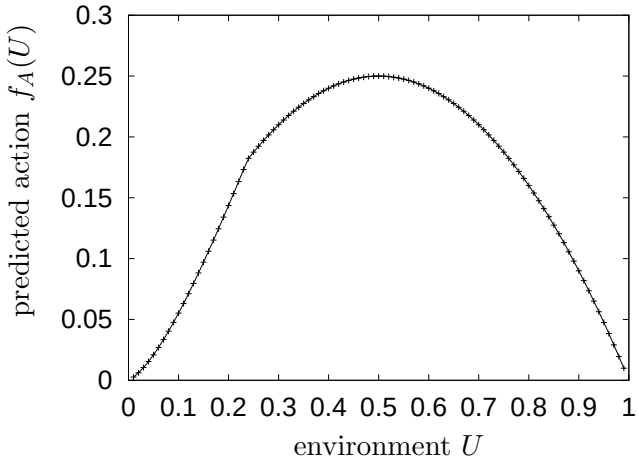


Figure 7: *Example for the current state of the prediction function of an agent. The function predicts the action of B for a given environment state $U$. Example take from Agent A after the first turn in run "fullpred80".*

$U$. Figure 7 shows an example. Note that in the simulations in this section the environment does not change. So, only the function value for $U = 0.5$ is used in the simulations.

## 4 Discussion

In this paper we have introduced a simple model of mutually observing agents and presented first simulation results. Form these preliminary results we can conclude that quite complex interaction dynamics can appear depending on the characteristics of the cognitive component. This means that the cognitive model of an agent has a strong bearing on the resulting dynamics. In our simulation the learning behavior is conditional upon the population size of the GP system. The better the learning of the future actions, the more easier social order will emerge.

GP has been found useful as a learning component because of its flexibility and its conceptual structure. The output of GP is machine code or a computer program that can be compiled. Thus execution of the learned function is highly efficient. The price for flexibility is that the learning process requires relatively[1] large computational resources. At any time step the GP system resumes the learning of the situation, that is from a technical perspective that the population of the GP system is initialized randomly each time in Step 6 of an action cycle. So, an agent interprets his memory contents each time independently from how he has interpreted the memory contents before. This can lead to a totally different interpretation (and thus prediction) from one turn to the next and destabilizes the order. We can avoid this problem by keeping the population of the GP system, instead of reinitializing it in each turn, and evolving the population in Step 6 only for some generations using the updated memory.

The complex behavior for intermediate prediction ability ($M = 40$, Fig. 3, middle) might be related to the *punctuated equilibrium* phenomenon as introduced by Gould and Eldredge (1977). It would be interesting for future investigations to look at the distribution of the length of ordered (cooperative) and disordered phases and analyze their scaling laws (Bak and Sneppen 1993). In future work we would also like to compare our model with the results by Taiji and Ikegami (1999) especially by adopting their *context space plots*.

**Acknowledgement**

We are grateful to Gudrun Hilles, Uwe Schimank, and Andre Skusa for helpful comments. The project is funded by the German Research Foundation (DFG), grant Ba 1042/7-1 and Schi 553/1-1.

## References

Akiyama, E. and R. Kaneko (2000). Dynamical systems game theory and dynamics of games. *Physica D 147*(3-4), 221–258.

Bak, P. and K. Sneppen (1993). Punctuated equilibrium and criticality in a simple model of evolution. *Phys. Rev. Lett. 71*, 4083–6.

[1] Compared to artificial neural networks or classical statistical methods.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone (1998). *Genetic Programming - An Introduction*. San Francisco and Heidelberg: Morgan Kaufmann and dpunkt Verlag.

Gould, S. J. and N. Eldredge (1977). Punctuated equilibria: the tempo and mode of evolution reconsidered. *Paleobiology 3*, 115–151.

Hofbauer, J. and K. Sigmund (1998). *Evolutionary Games and Population Dynamics*. Cambridge, UK: Cambridge University Press.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Natural Selection*. Cambridge, MA, USA: MIT Press.

Kron, T. and P. Dittrich (2001). A framework for building agents for social simulation. *Zeitschrift für Soziologie* submitted, in German.

Lindgren, K. and M. G. Nordahl (1994). Evolutionary dynamics of spatial games. *Physica D 75*(1-3), 292–309.

Luhmann, N. (1984). *Soziale Systeme*. Frankfurt a.M.: Suhrkamp.

Parsons, T. (1937). *The structure of social action*. New York, NY.

Parsons, T. (1968). Interaction. In D. L. Sills (Ed.), *International Encyclopedia of the Social Sciences*, Volume 7, London, New York, pp. 429–441.

Parsons, T. (1971). *The system of modern society*. Englewood Cliffs.

Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning 7*, 227–252.

Schimank, U. (2000). *Handeln und Strukturen. Einführung in die akteurtheoretische Soziologie*. Weinheim: Juventa.

Taiji, M. and T. Ikegami (1999). Dynamics of internal models in game players. *Physica D 134*(2), 253–266.