*Proceedings of the 2007 Winter Simulation Conference*
*S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.*

# REAL-TIME PREDICTION IN A STOCHASTIC DOMAIN VIA SIMILARITY-BASED DATA-MINING

Timo Steffens
Philipp Hügelmeyer

Schloss Birlinghoven
Fraunhofer Institute for Intelligent Analysis- and Information-Systems
53754 Sankt Augustin, GERMANY

## ABSTRACT

This paper introduces an application and a methodology to predict future states of a process under real-time requirements. The real-time functionality is achieved by creating a Bayesian tree via data-mining on agent-based simulations. The computationally expensive parts are handled in an offline phase, while the online phase is computationally cheap. In the offline phase the simulations are run and meaningful clusters of states are identified by use of virtual attributes. Then the transition probabilities between states of different clusters are organized in a Bayesian tree. Finally, in the online phase similarity measures are used again in order to classify query states into the clusters and to infer the probability of future states. The application domain is the support of military units during missions and maneuvers.

## 1 INTRODUCTION

In this paper we present an application for decision support in military missions and maneuvers. As research in opponent modelling has shown (Riley and Veloso 2002, Steffens 2005, Steffens 2006), anticipating future situations is helpful in order to counteract the enemy's actions. We describe a system for generating predictions that uses data mining and other artificial intelligence methods on data from simulation results. It is integrated in the LampSys simulation environment (Huegelmeyer, Steffens, and Zoeller 2006) which offers an intuitive graphical interface.

Decision Support Systems are intended to help a commander of a unit to decrease his cognitive load in situations that are difficult to assess due to dynamically changing information, complex interdependencies and time pressure. The system described in this paper aims at reducing the cognitive load by providing predictions of possible future situations at an appropriate level of abstraction. These predictions are clearly on a tactical rather than strategical level. While simulation is often used for stratecic decision making in a military context, it is far more seldom used for tactical simulation (Bosch and Rajab 2004), because the costs of collecting the neccessary data, model creation and validation, running the simulation and interpretation of the results can outbalance the benefit of the tactical question. In order to remedy this, LampSys comes with a set of pre-bundled szenario-kits, that make it easy to describe a simulation, run it and evaluate the results.

Conceptually, a digital double of the unit in the field is created via aerial reconnaissance and other data that the unit sends to the system. This way, the system keeps a world state which reflects the situation in the field. For this world state predictions about the probability of future situations are generated. In future work, these predictions will be sent to a personal digital assistant (PDA) of the unit.

In order to generate predictions on an abstract level that can be grasped intuitively, the prediction is based on meaningful clusters of states instead of single states. These clusters are obtained by simulation runs and are described using abstract virtual attributes (Richter 2003). The state of the digital double is mapped onto a cluster using a similarity-measure. The transition probabilities between this cluster and other clusters are represented by a Markov chain (Bremaud 1999). The complete process is depicted in figure 1.

The remainder of this paper is organized as follows. In the next section our approach is motivated by the demands of real-time prediction. Then our simulation environment LampSys is outlined shortly. In section 4 our approach to clustering is described. The computation of transition probabilities between clusters is covered in section 5. The generation and presentation of predictions is explained in section 6. Finally, the last section concludes and outlines future work.

## 2 OFFLINE ANALYSIS AND REAL-TIME PREDICTION

The traditional approach for using simulation for prediction is not adequate for the requirements of prediction in military

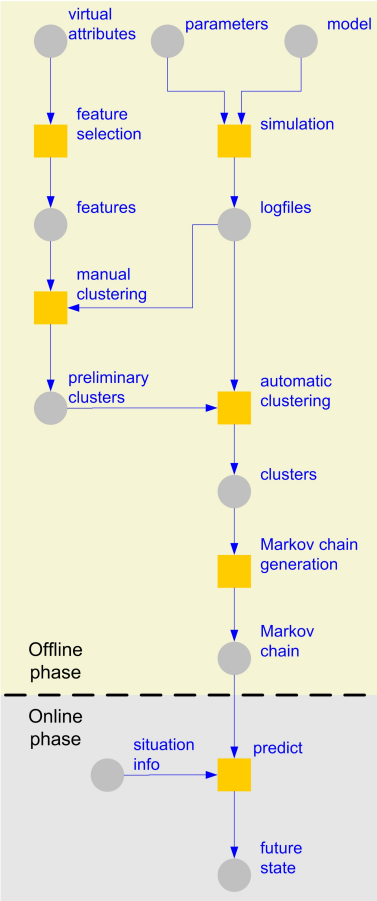


Figure 1: The complete process is divided into a computationally expensive offline phase and a computationally cheap online phase. The flow of data (circles) between actions (squares) is denoted by the arrows.

missions. Often, predictions are generated by simulating the current state into a defined number of time steps into the future. Since processes in the real world are usually non-deterministic, a single simulation run does not provide reliable predictions. Thus, many simulation runs are executed and their results are aggregated, for example via a Monte Carlo analysis (Bosch and Rajab 2004). Such approaches and the general class of Model Predictive Control approaches (Garcia, Prett, and Morari 1989) execute all of their computation in the time-critical online phase, that is, after the prediction task was issued. However, for real-time predictions during military missions, much of the computation should be done offline, before the request for predictions is issued. The approach presented in this paper aggregates the results of simulation runs beforehand.

Our work is more similar to the approach introduced in Sheikh-Bahaei and Hunt (2006) where simulation results are clustered and the prediction is achieved via assignment of a new compound to a cluster. We extend the approach from static classifications to a probabilistic process-oriented approach.

## 3 SIMULATION ARCHITECTURE

### 3.1 Scenario-based Approach

The utility of modeling and simulation tools for military applications highly depends on the quality of the utilized scenarios. Therefore, the careful design and precise description of scenarios as well as the ability to reliably communicate these assets is of prime importance. Moreover, considering non-deterministic wargaming, one must be able to specify and document alternative courses of action in order to enable a comprehensive analysis as well as comparisons with the optimal procedure. Therefore, a precise formal specification of complete scenarios constitutes an essential part of modern simulation environments. A complete description of a scenario has to include the environment (landscape, terrain, weather), forces, enemies, and non-combatants. The simulation language LAMPS (Huegelmeyer, Steffens, and Zoeller 2006) supports both the description of scenarios and the executable specification of agent behavior for compound agent groups down to individual agents. Like other modern simulation languages (L'Ecuyer and Buist 2005), LAMPS can be displayed both graphically and as a rule-set. We use it to specify the scenarios and situations in the military domain.

### 3.2 Simulation Environment

The language LAMPS can be executed by the LampSys simulation system. LampSys extends the Flip-Tick-Architecture (FTA) (Richter 1999), which has its roots in the JANUS project developed at the Gesellschaft für Mathematik und Datenverarbeitung (GMD) (Beyer and Smieja 1994). At its core, FTA is a design paradigm for scalable distributed systems that exhibit a priorily unknown dynamic characteristics as well as disturbances and inaccuracies which are difficult, if not impossible, to model in a closed-form mathematical approach.

LampSys is based on the concept of clans, which are groups of agents. It comprises three classes of entities: agents, clans, and tags. A clan $A$ is formed by a set of individual agents $a_j$.

$$A = \{a_1, \ldots, a_i, \ldots, a_k\}$$

Each actor $a_j$ is composed of a set of typed attributes $U_j$, whose value assignment determines the agent's state, together with an action function $f_j$, which entails all operations that can be performed by the agent.

$$a_j = (U_j = (u_{1,j}, \ldots, u_{m,j}), f_j)$$

Structural information concerning agents, i.e. names and types of attributes, is described via agent types. Formally, we have $type(a_j) = t$, if and only if agent $a_j$ is of type $t$. The system supports agent templates that can be used to store prototypical value assignments. Thus, an individual agent can be created either by instantiation of its type, or by copying from a pre-defined template.

The principle of autonomy of agents forbids the direct manipulation of internal data structures and behaviors of other agents. Consequently, all interactions between agents are handled via messages. Upon receiving a request, the agent is able to analyze its content and to decide whether it wants to comply. The basic unit of execution is called a cycle. During one cycle, the agent reads its messages and triggers the appropriate actions, which might consist of writing messages to other agent.

A scheduling clan is a set of agents sharing a common pace, i.e. all elements of a clan have the same time resolution $dt$. This in turn implies that their cycles are synchronized and that the clan switches from cycle to cycle as regularly as the tick of a clock. It is important to note that different agents do not necessarily share the same time resolution. Instead, the architecture supports individual running speeds for every agents. Moreover, time steps can vary from cycle to cycle. Thus, adaptive control of time increments can be realized. This is particularly valuable for increasing the time resolution in the computation of dynamics equations for fast moving objects.

The messages used for inter-agent communication are called tags. Instead of setting up a direct communication with other agents, agents register with one ore more interaction-clan. They send their messages to a designated clan-chief of that clan. While clan-chiefs can in principle implement other communication protocols such as publish-and-subscribe or blackboard, LampSys most often uses so-called tag-boards for discrete-time simulation. For event-based simulation other communication protocols are used. Tag-boards serve as the functional units for handling messages. In formal terms, a tag-board forms a medium $M_n$ for message exchange, while a LampSys system is capable of supporting multiple media:

$$M = \{M_1, \ldots, M_m\}$$

A tag-board consists of two sides. One is write-only and contains all tags sent to the board in time step $t$, whereas the other side is read-only and encompasses all tags written in time-step $t-1$. Analogously to agents, each tag-board has its own time resolution and thus its own cycle time. During a board cycle, the write-only side is flipped over. Thereby, the read-only part mirrors the tag content of the write part from the previous time-step. The write-only side is deleted after flipping. In this way, the lifetime of tags is effectively controlled by the time-scale of the pertaining board.

With this approach, fully synchronized (all agents and tag-boards share the same time resolution) as well as completely asynchronous systems (every agent and every board has its own time-scale) can be modeled in terms of the FTA.

The mathematical model of this agent architecture is a system of flexibly coupled inhomogeneous difference equations.

## 4 CLUSTERING OF STATES

The system introduced in this paper is required to present the user with explicit probabilities of future situations. The state space in complex situations is very large, thus the probability of single states becomes arbitrarily small. To address this issue, our prediction system predicts clusters of states instead of single states. However, this introduces the problem of acquiring meaningful clusters and to label them with human understandable names. This section describes how simulation states are clustered and labelled. First, we outline how simulation states are represented in LampSys. Then, the similarity measures and clustering techniques are described.

### 4.1 Simulation States in LampSys

LampSys is based on the concepts of ITSimBw (Huegelmeyer, Steffens, and Zoeller 2006). This means that all entities in the simulation is modelled as an agent. The behavior and the state of an agent is modelled as a set of attributes. Thus, the state of the simulation is specified by the values of all attributes of all agents.

Formally, an agent $a_i$ has a type $type(a_i) = tp$ which specifies the set of attributes. That is, the agent is specified by the set of attributes $U_i = \{u_{1,i}, \ldots, u_{n_{tp},i}\}$. An example for an agent type in the given domain would be `infantry`, which might have the attributes `position`, `physicalState`, `ammunition`.

According to the aforementioned FTA-paradigm, a real system is simulated by iterating the system states over the time $t$. Thus, the agents' attributes are variables of time. With $dt \to 0$ this iteration system is a system of $v$ coupled differential equations

$$D = (D_1, \ldots, D_v)$$

where the action function $f_j$ of an agent $a_j$ can be seen as the equation $D_j \in D$. The agent's attribute set is the variable vector of the equation. Thus

$$U_i^{t+dt} = f_i\left(U_i^t\right)$$

for $dt \to 0$ with

$$U_i^{t+dt} = U_i^t + dt * f_i\left(U_i^t\right).$$

In an implementation the condition $dt \to 0$ cannot be satisfied. Thus, the FTA-paradigm uses a set of difference equations. For each agent $A_i$ an individual $dt_i$ can be chosen for computing the steps of $D_i$.

While the simulation progresses, it steps through a series of states $A^t$. The simulation state at time $t$ is given by the set of all attribute values of all agents: $A^t = (U_1^t, U_2^t, \ldots, U_v^t)$. Thus, while the simulation progresses, there develops a series of simulation states $A = A^0, \ldots, A^p, A^q, \ldots, A^l$ with $t_q - t_p = Min_{h=1}^{v} dt_v$. As mentioned earlier, the intervals between time steps in $A$ are not necessarily equally long, since the interval is chosen at each time step as the minimum of the intervals that each agent proposes for its own action.

### 4.2 Clustering

Clusters are sets of states that are similar to states in the same cluster, and unsimilar to other states. Which states are similar is defined by a similarity measure. That is, a similarity measure maps a pair of simulation states into the interval $[0..1]$ by averaging the similarity of the agents:

$$sim(A^p, A^q) = \frac{\sum_{i=1}^{v} sim(a_i^p, a_i^q)}{v}$$

Agents that are compared to each other are always of the same type. The similarity of two agents is computed as the average of their attribute similarities.

$$sim(a_p, a_q) = \sqrt{\frac{\sum_{i=1}^{n_{type(a_p)}} sim(u_{i,p}, u_{i,q})}{n_{type(a_p)}}}$$

Since the agents' attributes can have non-numerical or numerical values, we use different similarity measures depending on the attribute types. For numerical attributes, the standard Euclidean similarity measure is used:

$$sim(x,y) = 1 - \left(\frac{x-y}{r}\right)^2$$

where $r$ is the range of the attribute. For non-numerical attributes, the identity comparison is used: $sim(x,y) = 1$ iff $x = y$, 0 else.

Attributes are not weighted differently. Instead, attributes that are not relevant for the task are not included at all. Additionally to the agents' attributes, it is possible to define virtual attributes (Richter 1999) if domain knowledge is available. Such virtual attributes are not explicitly represented in the simulation, but can be inferred or aggregated from the other attributes. A virtual attribute $M_i$ is defined formally as

$$U \times U \times \ldots \times U \to M_i,$$

where $U = \cup_{i=1}^{v} U_i$.

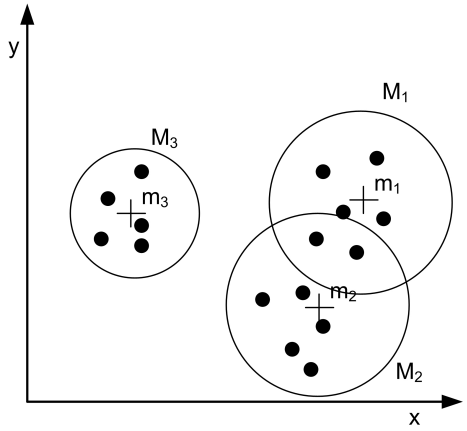


Figure 2: Example for ambiguous clustering. Solid circles depict simulation states, crosses depict center of the clusters. The clusters $M_2$ and $M_3$ overlap so that some simulation states cannot be mapped to a cluster in a non-ambiguous way.

An example for a virtual attribute is the attribute `center_of_the_infantry_units'_positions` which can be inferred from the `position`-attribute of the agents of type `infantry` that belong to the same unit. Our previous work showed that such virtual attributes often provide a good partitioning of the state space (Steffens 2006).

The LampSys environment provides an extentable set of virtual attributes which can be selected by the user for a clustering session. Thus, the user specifies a set $M = \{M_1, M_2, \ldots, M_3\}$ of virtual attributes $M_i$. The state space spanned by $M$ is considerably smaller than the simulation state space. Furthermore, its dimensions correspond to more abstract and thus more intuitive concepts than the rather technical original attributes.

Via the graphical interface of LampSys the user can assign simulation states to clusters. Whenever he encounters a state that does not fit into the existing clusters, he defines a cluster with a new label. This way, a set of clusters $C = \{C_1, C_2, \ldots, C_n\}$ is generated, where for each $C_i$ there exists a manually defined label $l_i = label(C_i)$.

Since $C$ is created manually, a subsequent process automatically checks whether the clusters are non-ambiguous. Furthermore, the center of the clusters is computed. This is achieved using the k-means-algorithms (MacQueen 1967). The center of each cluster is computed in an iteration of the algorithm until the centers do not change anymore. If the algorithm does not converge in a given amount of time, this is an indicator that the clusters are ambiguous (see figure 2).

In this case the user is asked to recheck his assignments of states to clusters or to expand the set of virtual attributes. Increasing $M$ with an additional virtual attribute is necessary if the given dimensions are not sufficient to partition the clusters.

If a set of clusters is generated, each simulation state can be assigned to its most similar cluster center via the function $c(A^i) = max_{j=1}^{k}(sim(A, m_i)$ where $m_i$ is the center of cluster $M_i$.

### 4.3 Score

In order to support the user in choosing from the alternative states in a choice situation, we assign a score to each individual state. The score is based on the distance to the aquired goal, the time left to reach the goal, the own lost resources to reach this state and the propability to reach the goal at all. Additionaly individual other factors can be added.

$$S_C = S_{gdist_C} * S_{time_C} * S_{res_C} * S_{pgoal_C}$$

A fitness value of 0 denotes that the mission goal is not achieveable anymore (e. g. all units are lost). A value of 1000 means that the mission goal is optimally achieved with the optimal use of ressources and in the minimal amount of time.

## 5 TRANSITION PROBABILITIES

In this section we describe how we compute the probability of future states.

A simulation steps through a series of states which can be assigned to clusters. Thus, a series of clusters can be generated. The transition between clusters is non-deterministic and can even be cyclic, for example

$$C^1 \rightarrow C^2 \rightarrow C^1 \rightarrow C^3 \rightarrow C^2 \rightarrow C^4 \rightarrow C^5 \rightarrow \ldots$$

The series of clusters can be viewed as a stochastic process in discrete time. A time step in this series depends solely on its predecessor. We use markov chains (Bremaud 1999) to model such processes. For a number of simulation runs, the frequency of transitions between each pair of clusters is computed. Based on these frequencies, the transition probability between clusters is inferred. The result is a directed graph (see figure 3) where clusters are nodes and where the arcs are formed by the transition probabilities

$$P\left(C_{t+1} = c\left(A^{t+1}\right) | C_t = c\left(A^t\right)\right)$$

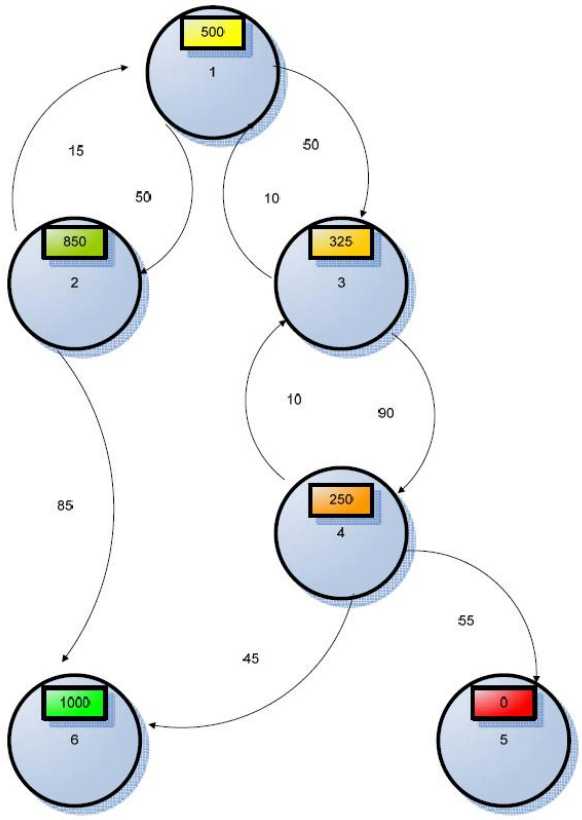


Figure 3: A directed graph depicting the transition probabilities (arcs) between clusters (nodes). The fitness value of clusters is depicted as a rectangle in the nodes.

## 6 PREDICTION

In the application discussed in this paper, the system keeps a state which reflects the situation in the field via aerial reconnaissance and communication of data. For this current state the system predicts possible future states.

Technically, using the function $c(A)$ a state can be mapped into a cluster $M_i$. If the similarity between the state and the center of $M_i$ is below a certain threshold, the mapping is considered inappropriate. This can be the case if the simulation runs did not cover this particular situation or if the clusters do not cover the state space well.

In any case, the system tells the user a confidence factor about the mapping of the state into a cluster.

Using the Markov graph, the system presents the probabilities of future situations and graphically depicts the fitness values of these situations.

The prediction phase is deliberately simple, since the requirements of the whole system are that the computational effort has been done beforehand.

## 7 CONCLUSION

We presented an approach for supporting tactical decisions in military situations. From a number of simulations, scenario-specific predictions are generated for given situations in real-time. This real-time functionality is achieved by partitioning the process into a computationally expensive offline phase and a computationally cheap online phase. By using virtual attributes the predicted situations provide an appropriate level of abstraction in order to support time-critical decisions.

Future work includes the automation of clustering by means of associating states to behavior specifications of the simulated agents.

## ACKNOWLEDGMENTS

This work is done under contract for the department A5 of the IT-office of the German armed forces (IT-AmtBw). At this institution, the project is overseen by Major T. Doll, whose valuable contributions are gratefully acknowledged.

## REFERENCES

Beyer, U., and F. Smieja. 1994. Janus: A society of agents. Technical report, GMD, Sankt Augustin, Germany. GMD Report No. 840.

Bosch, P. C., and M. Rajab. 2004. Autonomous predictive-adaptive simulation for operations support. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 2018–2024: Winter Simulation Conference.

Bremaud, P. 1999. *Markov chains*. Berlin: Springer.

Garcia, C. E., D. M. Prett, and M. Morari. 1989. Model predictive control: theory and practice a survey. *Automatica* 25 (3): 335–348.

Huegelmeyer, P., T. Steffens, and T. Zoeller. 2006. Specifying and simulating modern warfare scenarios with ITSimBw. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. L. D. M. Nicol, and R. M. Fujimoto: ACM.

L'Ecuyer, P., and E. Buist. 2005. Simulation in java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*, 611–620: ACM.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, ed. L. M. LeCam and J. Neyman, 281–297. Berkeley, CA: University of California Press.

Richter, G. 1999. Flip-tick architecture: A cycle-oriented architecture for distributed problem solving. Technical report, GMD, Sankt Augustin, Germany. GMD Report No. 19.

Richter, M. M. 2003. Fallbasiertes Schliessen. *Informatik Spektrum* 3 (26): 180–190.

Riley, P., and M. Veloso. 2002. Planning for distributed execution through use of probabilistic opponent models. In *Proceedings of the Sixth International Conference on AI Planning and Scheduling (AIPS-2002)*, ed. M. Ghallab, J. Hertzberg, and P. Traverso, 72–81. Menlo Park, CA: AAAI Press.

Sheikh-Bahaei, S., and C. A. Hunt. 2006. Prediction of in vitro hepatic biliary excretion using stochastic agent-based modeling and fuzzy clustering. In *WSC '06: Proceedings of the 37th conference on Winter simulation*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. L. D. M. Nicol, and R. M. Fujimoto, 1617–1624: Winter Simulation Conference.

Steffens, T. 2005. Similarity-based opponent modelling using imperfect domain theories. In *IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, ed. G. Kendall and S. Lucas, 285–291. Colchester, UK: Essex University.

Steffens, T. 2006. *Enhancing similarity measures with imperfect rule-based background knowledge*. Ph. D. thesis, University of Osnabrueck.

## AUTHOR BIOGRAPHIES

**TIMO STEFFENS** received his Master in Artificial Intelligence and Computational Linguistics from the University of Osnabrueck, Germany, in 2002. He received his doctoral degree in 2006 from the Institute of Cognitive Science. At the Fraunhofer Institute for Intelligent Analysis- and Information-Systems he works in projects with the industry and academia. His main research activities are in the area of agent-based simulation, multi-agent systems and machine learning. He can be contacted via <timo.steffens@iais.fraunhofer.de>.

**PHILIPP HÜGELMEYER** graduated in Artificial Intelligence and Computational Linguistics at the University of Osnabrueck, Germany, in 2002. He worked as a researcher at the same university until 2005. Now he is a researcher at the Fraunhofer Institute for Intelligent Analysis- and Information-Systems in Sankt Augustin. His research interests include Multi-agent systems, agent based simulation and social choice theory. His email-address is <philipp.huegelmeyer@iais.fraunhofer.de>.