Towards Adaptive, Interactive Virtual Humans in Sigma

Volkan Ustun[1] and Paul S. Rosenbloom[1,2]

[1] Institute for Creative Technologies, [2] Department of Computer Science
University of Southern California, Los Angeles, CA USA

Abstract. Sigma is a nascent cognitive architecture/system that combines concepts from graphical models with traditional symbolic architectures. Here an initial Sigma-based virtual human (VH) is introduced that combines probabilistic reasoning, rule-based decision-making, Theory of Mind, Simultaneous Localization and Mapping and reinforcement learning in a unified manner. This non-modular unification of diverse cognitive, robotic and VH capabilities provides an important first step towards fully adaptive and interactive VHs in Sigma.

1 Introduction

Virtual humans (VHs) are synthetic characters that can take the part of humans in a variety of contexts. The main goal for VHs is to look and behave as real people to the extent possible [17], including: (1) using their perceptual capabilities to observe their environment and other virtual/real humans in it; (2) acting autonomously in their environment based on what they know and perceive, e.g. reacting and appropriately responding to external events; (3) interacting in a natural way with both real people and other VHs using verbal and non-verbal communication; (4) possessing a *Theory of Mind* (*ToM*) to model their own mind and the minds of others; (5) understanding and exhibiting appropriate emotions and associated behaviors; and (6) adapting their behavior through experience. Most critically, this broad range of capabilities must be integrated together and work coherently. This integration can be quite hard, but if it can yield more than the sum of its parts, it can simplify a variety of other aspects.

Sigma [13] is being built as a computational model of general intelligence that is based on combining what has been learned from over three decades worth of independent work in cognitive architectures [7] and graphical models [3]. The long-term goal is to understand and replicate the *architecture of the mind*; i.e., the fixed structure underlying intelligent behavior in both natural and artificial systems. This ambitious goal strives for the complete control of VHs (and robots in the future) that behave as closely as possible to humans, primarily by achieving and integrating the list above in a manner that should ultimately yield *plug compatibility* between humans and artificial systems. Such compatibility potentially creates very flexible VHs and simulation environments.

Sigma's development is guided by a trio of desiderata: (1) *grand unification*, uniting the requisite cognitive and non-cognitive aspects of embodied intelligent behavior in complex real worlds; (2) *functional elegance*, yielding broad cognitive (and sub-cognitive) functionality from the interactions among a small general set of mechanisms; and (3) *sufficient efficiency*, executing rapidly enough for anticipated applications. The first and last desiderata are directly relevant to the construction of broadly capable, real-time, VHs, while the middle one implies a rather unique path towards them, where instead of a disparate assembly of modules, all of the required capabilities are constructed and integrated together on a simple elegant base. Most of the work to date on Sigma has individually explored particular capabilities for learning, memory and knowledge, decision making and problem solving, perception, speech, Theory of Mind, and emotions. These individual capabilities are important in building human-like intelligence but getting them to work together is also quite challenging [17]. *Sigma's* non-modular, *hybrid* (discrete + continuous) *mixed* (symbolic + probabilistic) character supports attempting a deep integration across these required VH capabilities, straddling the traditional boundary between symbolic cognitive processing and numeric sub-cognitive processing.

The work here combines a subset of these capabilities within Sigma to construct an adaptive, interactive VH in a virtual environment. The VH is adaptive not only in terms of dynamically deciding what to do, but also in terms of embodying two distinct forms of relevant learning: (1) the automated acquisition of maps of the environment from experience with it, in the context of the classic robotic capability of Simultaneous Localization and Mapping (SLAM); and (2) reinforcement learning (RL), to improve decision making based on experience with the outcomes of earlier decisions. The VH is interactive both in terms of its (virtual) physical environment – through high-level perception and action – and other participants, although the latter is still quite limited. Although speech and language are being investigated in Sigma, neither is deployed in this VH, so social interaction is limited to constructing – actually, learning, with the help of RL – models of the self and others.[1] These forms of adaptivity and interaction are combined together within Sigma and, for this initial VH, with a basic rule-based decision framework.

Sigma provides the ability to exhibit this combination of capabilities in a unified manner because of its grounding in a *graphical architecture* that is built from graphical models [3] (in particular, factor graphs and the summary product algorithm [5]), n-dimensional piecewise linear functions [10], and gradient descent learning [14]. The required VH capabilities emerge in a functionally elegant manner from the interactions among this small but general set of mechanisms, plus knowledge. For example, RL arises from the interactions between gradient descent learning and particular forms of both domain-specific and domain-independent knowledge [11].

Chen et al. [1] discussed the fusion of symbolic and probabilistic reasoning at an earlier stage of the development of Sigma. In that study initial steps towards grand unification were demonstrated when perception, localization and decision-making were implemented within a single graphical model, with interaction among these capabilities modulated through shared variables. The work here greatly expands on this approach to yield a more significant combination of capabilities, plus a deployment in a VH that is embodied in the SmartBody character animation system [15] and

[1] As explained later, what the VH actually does is to model itself as if it were a different VH.

that operates within a 3D virtual environment rather than a toy one-dimensional space. In the work here, SmartBody's internal movement, path-finding and collision detection algorithms are used in animating the VH's actions, although eventually much of this is to be moved within Sigma. Sigma has no direct access to the virtual environment, but can perceive and act on it through a (deliberately) noisy interface.

In addition to the contributions of this work to the creation of adaptive, interactive VHs, it is also thus an important step in the maturation of Sigma. The hope is that this will serve as a foundation towards developing even more complete and functional VHs. It also should help better understand how to handle the interactions between VHs and their environments in a robust manner that is similar to, but still somewhat simpler than, interactions between robots and the real world.

The conceptual model(s) for the VHs, and their interactions with the virtual environment, are described in Section 2. A basic introduction to Sigma is provided in Section 3. Section 4 provides a discussion of the Sigma models that control the VH(s). Conclusions and possible extensions to this work are discussed in Section 5.

2 Conceptual Model and Environment

Physical security systems are comprised of structures, sensors, protocols, and policies that aim to protect fixed-site facilities against intrusions by external threats, as well as unauthorized acts by insiders. Physical security systems are generally easy to understand but they also allow complex interactions to emerge among the agents. These properties make physical-security-systems simulation a natural candidate as a testbed for developing cognitive models of synthetic characters [18]. Similar to the discussion in [18], a physical-security-system scenario in a retail store has been selected as a platform to develop and test Sigma VH models.

In a typical retail-store shoplifting plot, offenders first pick up merchandise in a retail store and then try to leave without getting caught by any of the store's security measures. A simple grab-and-run scenario is considered in this paper, but a large number of different scenarios are possible. In this scenario, the intruder needs to locate the desired item in the store, grab it, and then leave the store. The role of security is to detain the intruder before s/he leaves the store. A basic assumption is that it is not possible to tell what the intruder will do until s/he picks up an item and starts running. The security can immediately detect the activity and start pursuing the intruder once the item is picked up (assuming CCTV). If the intruder makes it to the door, it is considered a success for the intruder.

For the basic setup, it is assumed that the intruder does not know the layout of the store and hence it has to learn a map and be able to use it to localize itself in the store. When the intruder locates the item of interest, it grabs the item and leaves the store via one of the exits. In the hypothetical retail store used here (Fig. 1), there are shelves (gray rectangles), the item of interest (the blue circle) and two entry/exit doors (red rectangles). The intruder leaves the store via either (1) the door it used to enter or (2) the door closest to the item of interest. The main task for security is to learn about the exit strategies of intruders and use this to effectively detain them.

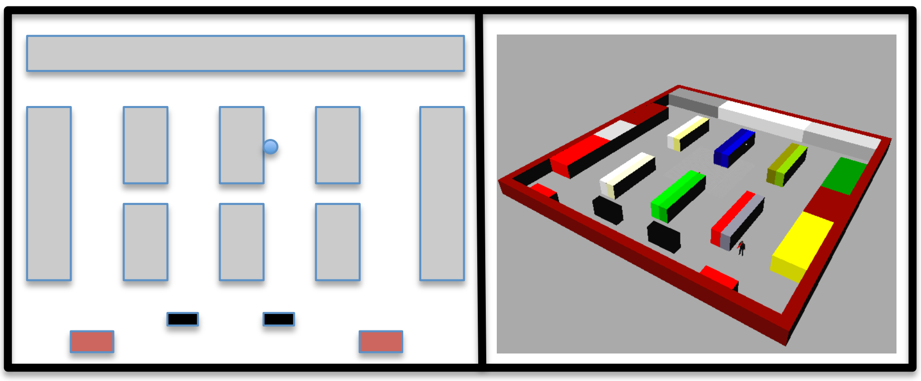

Fig. 1. Layout of the store and its SmartBody representation

SmartBody [15], a Behavior Markup Language (BML) [4] realization engine, is used as the character animation platform for this study, with communication between the Sigma VH model and SmartBody handled via BML messages. In the current set-up, locomotion and path finding are delegated to the SmartBody engine. Sigma sends commands and queries to SmartBody to perform these tasks and to return perceptual information. Two basic types of perception are utilized by the Sigma VH model: (1) information about the current location of the agent, mimicking the combination of direction, speed and odometry measurements for a robot; and (2) objects that are in the visual field of the agent and their relative distances, mimicking the perception of the environment for a robot (currently the agents have x-ray vision, but a more realistic visual system is in the works). Location information is conveyed to the Sigma VH model with noise added; perfect location information is not available to the model.

3 Sigma

The Sigma cognitive architecture is built on *factor graphs* [5] – undirected graphical models [3] with variable and factor nodes, and functions that are stored in the factor nodes. Graphical models provide a general computational technique for efficient computation with complex multivariate functions – implemented via hybrid mixed *piecewise-linear functions* [10] in Sigma – by leveraging forms of independence to: decompose them into products of simpler functions; map these products onto graphs; and solve the graphs via message passing or sampling methods. The *summary product algorithm* [5] is the general inference algorithm in Sigma (Fig. 2). Graphical models are particularly attractive as a basis for broadly functional, yet simple and theoretically elegant, cognitive architectures because they provide a single general representation and inference algorithm for processing symbols, probabilities and signals.

The Sigma architecture defines a high-level language of *predicates* and *conditionals* that compiles down into factor graphs. Predicates specify relations over continuous, discrete and/or symbolic arguments. They are defined via a name and a set of typed arguments, with *working memory* (WM) containing predicate instantiations as

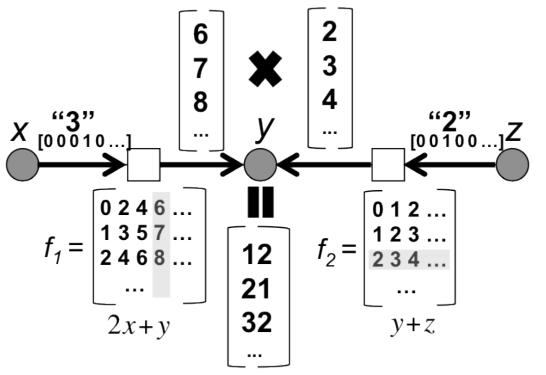

Fig. 2. Summary product computation over the factor graph for $f(x,y,z) = y^2+yz+2yx+2xz = (2x+y)(y+z) = f_1(x,y)f_2(y,z)$ of the marginal on y given evidence concerning x and z.

functions within a WM subgraph. Predicates may also have perception and/or long-term memory (LTM) functions. For perceptual predicates, factor nodes for *perceptual buffers* are connected to the corresponding WM subgraphs. For example, `Observed (object:object visible:boolean)` is a perceptual predicate with two arguments: (1) `object` of type `object`; and (2) `visible` of type `boolean`. This predicate specifies which objects are visible to the agent at any particular time. For memorial predicates, *function factor nodes* (FFNs) are likewise connected to the corresponding WM subgraphs. Messages into FFNs provide the gradient for learning the nodes' functions.

Conditionals structure LTM and basic reasoning, compiling into more extended subgraphs that also connect to the appropriate WM subgraphs. Conditionals are defined via a set of *predicate patterns* – in which type specifications are replaced by constants and variables – and an optional *function* over pattern variables. *Conditions* and *actions* are predicate patterns that behave like the respective parts of rules, pushing information in one direction from the conditions to the actions. The example conditional in Fig. 3 updates the information about which objects have been seen so far, based on the information in the `Observed` predicate. *Condacts* are predicate patterns that support the bidirectional processing that is key to probabilistic reasoning, partial matching, constraint satisfaction and signal processing. Overall, conditionals provide a deep combination of rule systems and probabilistic networks.

```
CONDITIONAL Seen
     Conditions: Observed(object:o visible:true)
     Actions: Seen-Objects(object:o)
```

Fig. 3. Conditional for context information.

Processing in Sigma is driven by a *cognitive cycle* that comprises input, graph solution, decisions (selection of best elements from distributions), learning, and output. Graph solution occurs via the summary product algorithm. Most of perception and action is to occur within graph solution in Sigma, rather than within external modules [3]. Decisions in Sigma, in the classical sense of choosing one among the best operators to execute next, are based on an architecturally distinguished selection predicate: `Selected(state:state operator:operator)`. Typically the operator associated with the highest value, or utility, in the distribution is selected. Once an operator is selected, it can be applied by conditionals with actions that modify the state in working memory. As in Soar [6], a single cognitive cycle yields *reactive* processing; a sequence of cognitive cycles yields *deliberative* processing; and if no operator can be selected, or a selected operator can't be applied, a Soar-like impasse [6] occurs, leading to *reflective* processing.

4 The Sigma Virtual Human (VH) Model

In a typical physical security system setting there are intruders (shoplifters) and security personnel. There may also be neutrals, but they are not modeled in this work. In this initial implementation, only an intruder is modeled as a VH, although part of the capability of a security agent – of learning a model of the intruder based on witnessing the intruder's actions – has been grafted onto the implementation of the intruder, as if it were observing itself from the outside (although the intent is eventually to move this into a distinct VH). In the current scenario, the intruder does not know the layout of the store in advance, and so it must learn a map of the store while simultaneously localizing itself in the learned map (Section 4.1). The intruder also needs a decision framework to dynamically decide what to do based on its immediate circumstances (Section 4.2). Learning the strategy of the intruder – i.e., whether it exits through the entry door or the closest door – occurs by learning a policy for the agent via RL and then using this policy and the agent's actions to determine on each trial the relative likelihoods of the two strategies being used (Section 4.3).

4.1 Simultaneous Localization and Mapping (SLAM)

The intruder has no a priori knowledge about the layout of the retail store (which is as shown in Fig. 1). Therefore, it has to learn a map of the store while simultaneously using the map to localize itself within the store. A 31×31 grid is imposed on the store for map learning. A VH only occupies a single grid cell, whereas objects in the environment – such as shelves – may span multiple cells.

The Sigma VH model defines two perceptual predicates – `Location-X(x:location) and Location-Z(z:location)` – to represent the location of the VH on the grid (the `location` type is discrete numeric, with a span of 31). Together these two predicates span the space of 2D cells in the grid, with perception of *x* and *z* involving a noise model that assumes any neighboring cell of the correct cell may be perceived as the agent's current location. The objects in the visual field are also perceived, along with the relative distances of the center of these objects to the agent's location.

To perform SLAM, a Dynamic Bayesian Network (DBN) [2] is defined via two almost identical conditionals, one for *x* (Fig. 4) and a similar one for *z*. These conditionals convert relative locations of objects given the agent to absolute locations in the map, using the affine transform *translation* to offset the agent's current location by the distance to the object. In Fig. 4, `Object-Location-X` is a memorial predicate that has a function that represents the map that is learned via gradient descent. Since both the `Location-X` and `Object-Location-X` patterns are condacts, processing is bidirectional between them; both perception of the VH's location and perception of the objects' locations have an impact on the posterior for the VH's location. This bidirectional processing forms the basis for SLAM, where the map is learned while it is simultaneously used for localization.

```
CONDITIONAL SLAM-X
    Conditions:  Observed(object:o visible:true)
                 Object-Distance-X(dist-x:dx object:o)
    Condacts:    Location-X(x:lx)
                 Object-Location-X(object:o x:(lx-dx))
```

Fig.4. SLAM conditional for the *x* coordinate

4.2 Behavior Rules

The objective of the intruder is to grab the item of interest (Fig. 1) and then to leave the store through one of the exits without being detained. In the current implementation, four basic behaviors, with corresponding Sigma operators, are available to the intruder: (1) *walk towards target object*; (2) *run towards target object*; (3) *pick up target object*; and (4) *walk towards random object.* Target objects play a role in the intruder achieving its goals, while random objects drive exploration of the store.

The intruder is initialized with a sequence of target objects that it needs either to walk towards or to pick-up. Given that the intruder does not have a priori knowledge of the store, it may need to explore the store, mapping it in the process, to locate the target objects. The basic operator used for exploration is *walk towards random object.* The intent is that doing this will help the agent eventually discover the target object.

This exploratory operator is always available; however, if other more task-relevant behaviors are available, they take precedence. For example, Fig. 5 shows a conditional in which the operator *walk towards target object* is suggested for selection with a utility of 0.5 when the VH has seen the target object (and hence, there is an estimate of its location). Exploration has a lower utility, so walking to a target object takes precedence if both operators are available. When the VH is within a threshold distance of the target object, a new operator – *pick up target object* – is then selected. The model terminates when the intruder reaches its preferred exit door, which acts as the target object for the *run towards target object* operator.

```
CONDITIONAL WALK-TOWARDS-TARGET
    Conditions:     Target-Object(object:o)
                 Seen-Objects(object:o visible:true)
    Actions:   Selected(operator:walk-target)
    Function:  0.5
```

Fig.5. Conditional suggesting the *walk towards target object* operator

4.3 Leveraging Reinforcement Learning (RL) to Model Others

One basic assumption made in this paper is that it is easy to recognize that a grab-and-go scenario has been initiated, by observing the pick-up behavior of the intruder.[2] However, even though security can easily recognize when such a scenario has been initiated, it still needs to intercept the intruder before it leaves the retail store. As there are two exit doors, early anticipation of the intruder's choice increases the chances of a successful detention.

Theory of Mind (ToM) involves formation of models of others and generation of expectations about their behavior based on these models to enable effective decisions in social settings [19]. In decision-theoretic approaches to ToM, such models can be represented as reward functions. For the intruder in our scenario there are two possible models, distinguished by whether a reward is received when the agent returns to its door of entry or when it reaches the nearest door from the item of interest. This corresponds to Bayesian approaches to multi-agent modeling that use a distribution over a set of policies to specify the beliefs that one agent has about another [8].

In general, RL enables agents to learn effective policies for task performance based on rewards received over a sequence of trials [16]. In Sigma, RL is not a separate architectural learning algorithm, but occurs through a combination of gradient-descent learning and the appropriate knowledge expressed in predicates and conditionals. Here, as in [9], RL is leveraged in selecting among models of other agents; in particular it is used to emulate the process by which a separate security agent would learn a model of the intruder. First a form of multiagent RL is used to learn a distinct *policy*, or *Q function*, for the intruder under each possible model, and then these policies are used in combination with the perception of the intruder's actions to yield a gradient over the two models that is proportional to the models' Q values for the performed actions. In particular, the model for which the observed action has higher Q values will have an increased likelihood in the posterior distribution.

The conditional that compares the Q values and generates a posterior distribution for the models is shown in Fig. 6. It multiplies the Q values for the observed action – specified by the location of the intruder and the direction of movement from that location – in each policy by 0.1, to scale down utilities in [0,10] to values for selection in [0,1], and then projects these values onto the model predicate to generate a posterior distribution on the model of the intruder.

```
CONDITIONAL PREDICT-MODEL
    Conditions:  Previous-RL-Loc(location:loc)
                 RL-Direction(direction:d)
                 Q(model:m location:loc direction:d
                   value:[0.1*q])
    Actions:     Model(model:m)
```

Fig.6. Model prediction conditional

[2] The details of this recognition process are beyond the scope of this paper but there are a variety of behavioral cues (e.g. posture changes while concealing an item, gait changes under stress etc.) that could be revealing. Exhibiting and detecting such cues is one of a number of intriguing future directions for this work.

Ultimately, this approach to model learning needs to be evaluated in terms of how well it helps security catch the intruder, but for now we will only consider how quickly it helps determine through which door the intruder is attempting to escape. Here the VH was first run for 20 trials for each exit-door scenario to learn the policies on a low fidelity 4×4 grid, which was reduced for efficiency purposes, but with the VH still actually moving on the full 31×31 grid.. Then each model was run for 5 trials for testing. The average number of cognitive cycles required to complete the scenario after picking up the item of interest is 31.2 for door 1 and 17.2 for door 2. The Sigma model correctly selects the exit strategy – P(Correct Exit Strategy) > 0.65 – after 12.4 cognitive cycles for door 1 (39.7% of the time to needed to reach the door) and 11.8 cognitive cycles for door 2 (68.6% of the time needed to reach the door). Thus, at least in a preliminary form, this demonstrates how model learning can help identify the correct door considerably before it is reached.

5 Conclusion

A first adaptive, interactive VH based on Sigma has been created that combines short-term rule-based adaptivity in decision making with two forms of long-term adaptivity (i.e., learning) – *mapping* in the context of SLAM, and *modeling* of others via RL – plus both interaction with a virtual environment and social interaction (in terms of ToM reasoning and learning). This VH, even as initial and limited as it is, provides an initial indicator of the potential for grand unification. To expand further on this potential, we are prioritizing the development of multiple VHs, plus the ability for them to interact via speech and language.

There are different types of participants – intruders, security, and neutrals – in a typical physical security system. Such variety makes the physical-security-system setting very flexible for the generation of scenarios that encompass many different interactions among VHs, and between VHs and humans. Consequently, an extended version of the current retail store security setting should yield a useful testbed for further exploration of this potential. As the scenarios get more complex, we expect the forms of cognition exhibited to become comparably sophisticated, going beyond simple rule-based reasoning to more involved combinations of reactive, deliberative and reflective processing.

In addition to grand unification, the VH here leverages Sigma's functionally elegant approach to providing and combining capabilities such as rule-based reasoning, SLAM, ToM, and RL. It also stretches, but does not quite break the desideratum of sufficient efficiency. Although the decision cycles were longer than the desired 50 ms [12] – around 250 ms – the impact on the performance of the VH was minimal because those activities requiring faster decisions were delegated to SmartBody's inner algorithms. Further analysis and optimizations are clearly required, but this should not detract significantly from how the VH here exhibits simple interactive, adaptive behavior by combining different types of reasoning and learning mechanisms under a unified model in a functionally elegant manner.

Acknowledgments. This effort has been sponsored by the Office of Naval Research and the U.S. Army. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. We would also like to thank Ari Shapiro for his overall support with SmartBody.

References

1. Chen, J., Demski, A., Han, T., Morency, L. P., Pynadath, D. V., Rafidi, N., & Rosenbloom, P. S. Fusing Symbolic and Decision-Theoretic Problem Solving+ Perception in a Graphical Cognitive Architecture. In *Biologically Inspired Cognitive Architectures* (2011).
2. Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, *2*(4), 31-43. (2010).
3. Koller, D., & Friedman, N. Probabilistic Graphical Models: Principles and Techniques. MIT press. (2009).
4. Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., & Vilhjálmsson, H. Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent virtual agents*. (2006).
5. Kschischang, F. R., Frey, B. J., & Loeliger, H. A. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498-519. (2001).
6. Laird, J. E.: The Soar Cognitive Architecture. MIT Press, Cambridge, MA (2012)
7. Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. *Cognitive Systems Research 10, 141-160.* (2009).
8. Pynadath, D. V., & Marsella, S. C. PsychSim: Modeling theory of mind with decision-theoretic agents. In *IJCAI*. (2005).
9. Pynadath, D. V., Rosenbloom, P. S., & Marsella, S. C. Reinforcement Learning for Adaptive Theory of Mind in the Sigma Cognitive Architecture. In *Artificial General Intelligence*. (2014).
10. Rosenbloom, P. S. Bridging dichotomies in cognitive architectures for virtual humans. In *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*. (2011).
11. Rosenbloom, P. S. Deconstructing reinforcement learning in Sigma. In *Artificial General Intelligence*. (2012).
12. Rosenbloom, P. S. Towards a 50 msec cognitive cycle in a graphical architecture. In *Proceedings of the 11th international conference on cognitive modeling*. (2012).
13. Rosenbloom, P. S. The Sigma cognitive architecture and system. *AISB Quarterly*, *136*. (2013).
14. Rosenbloom, P. S., Demski, A., Han, T., & Ustun, V. Learning via gradient descent in Sigma. In *Proceedings of the 12th International Conference on Cognitive Modeling*. (2013).
15. Shapiro, A. Building a character animation system. In *Motion in Games*. (2011).
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press. (1998)
17. Swartout, W. Lessons learned from virtual humans. *AI Magazine*, *31*(1), 9-20. (2010)
18. Ustun, V., Yilmaz, L., & Smith, J. S. A conceptual model for agent-based simulation of physical security systems. In *Proceedings of the 44th annual Southeast regional conference, ACM*. (2006).
19. Whiten, A., ed.: Natural Theories of Mind. Basil Blackwell, Oxford, UK. (1991)