Intelligent Techniques and Hybrid Systems Experiments Using the Acumen Modeling and Simulation Environment

Sotirios Tzamaras, Stavros Adam, Walid Taha

Intelligent Techniques and Hybrid Systems Experiments Using the Acumen Modeling and Simulation Environment*

Sotirios Tzamaras[1], Stavros Adam[1], and Walid Taha[2]

[1] Autonomous Systems Laboratory
Dept. of Informatics and Telecommunications
University of Ioannina, Ioannina, Greece
sotiris@tzamaras.com, adamsp@uoi.gr
[2] School of Information Technology
Halmstad University, Halmstad, Sweden,
walid.taha@hh.se

Abstract. Hybrid systems are dynamical systems of both continuous and discrete nature and constitute an important field of control systems theory and engineering. On the other hand, intelligent data processing has become one of the most critical devices of modern computer based systems as these systems operate in environments featuring increasing uncertainty and unpredictability. While these two approaches set completely different objectives, modern cyber-physical systems, taken as variants of hybrid systems, seem to constitute a field of increasing interest for applying intelligent techniques. Moreover, the examples of, not so recent, intelligent control systems are suggestive for considering a study on getting intelligent techniques close to hybrid systems. In this paper we present the experimental investigation we undertook in this direction. More specifically, we present and discuss the experiments carried out using Acumen a hybrid systems modeling and simulation environment. Without urging towards setting and solving questions of conceptual order we tried to figure out whether it is possible to represent intelligent behavior using a tool for modeling dynamical systems focusing on the study of its ability to permit the representation of both continuous and discrete intelligent techniques, namely, Reinforcement Learning and Hopfield neural networks. The results obtained are indicative of the problems related to the specific computational context and are useful in deriving conclusions concerning the functionality that needs to be provided by such modeling and simulation environments, in order to allow for the coexistence of hybrid systems and intelligent techniques.

1 Introduction

The increasing demand on systems theory and applications, especially regarding control, has caused a significant interest in the study of processes exhibiting both

* Part of this work was conducted by Sotirios Tzamaras, during his internship at the University of Halmstad, funded by the Erasmus+ program.

continuous and discrete dynamic behavior. These processes are studied by means of complex systems known as hybrid systems [1]which comprise two constituent elements, namely, the continuous-time models and the discrete-time components. Continuous-time models are governed by differential or difference equations while discrete-time components are driven by logic rules and are discrete-event systems using models, such as, automata, finite state machines, etc. In practical applications, it is commonplace to obtain a hybrid system whenever some continuous physical process is controlled by some embedded software system implementing a finite number of states, such as, on/off switches. Modern hybrid systems have received extensive interest due to a number of advances and developments in nonlinear control theory, intelligent control, adaptive control, computer science etc. Last but not least one should mention the increasing use of hybrid systems known as Cyber-Physical Systems (CPS). Typically, a CPS controls some physical process interacting with it through specialized computer and network interfaces. From the point of view of networked computer systems CPS's constitute entities residing in ecosystems commonly known as Internet of Things.

It is well known that modern systems operate in highly unpredictable and uncertain environments affecting the complexity of the physical processes and thereof the control systems. To a considerable extent, dealing with uncertainty has become possible thanks to a class of data driven techniques either of statistical nature originating from machine learning or based on nature inspired approaches from the field of computational intelligence. While the former seem to be more suitable for parameter or system identification tasks the latter have mostly been applied to control function design tasks. The ensemble of these data driven techniques and approaches constitute intelligent approaches and techniques which, recently, are considered to form a new field called machine learning control [5].

As a result, bringing together hybrid systems and intelligent approaches seems to be unavoidable towards designing efficient CPS's which undertake more and more demanding tasks. In a broader sense this perspective does not seem to be new as the control systems community posed the question of the very nature of intelligent control systems since the early '90s [2]. Modern intelligent techniques are based on a wide range of approaches elaborated on a variety of concepts of either statistical or heuristic nature. This implies that bringing close hybrid systems and intelligent behavior is a task which needs important conceptual work to be done in order to bridge the gap between analytical techniques on one hand and statistical and/or heuristic on the other. Moreover, at a more practical level it is important to consider the problem of having suitable tools for designing and simulating hybrid systems together with intelligent behavior in the same operational context.

1.1 Contributions

The work presented in this paper deals with this latter point of view. By means of an experimental study of an environment used for designing hybrid systems this study intends to contribute on the following issues:

1. Investigate the ability of such environments to support implementation of discrete structures and events.
2. Study the possibility to implement control structures governed by intelligent agents.
3. Draw conclusions on the possible level of interaction between intelligent modules and hybrid systems.

Note that, the exact role of the intelligent function, as well as, its detailed study of interaction and/or integration with the control function of a hybrid system are important matters that need careful and exhaustive study and are beyond the scope of this paper.

The rest of the paper is organized as follows. In Section 2, we present the work related to the problem described in the beginning of this section 1 together with a literature review which highlights specific points of the problem set. Section 3 is devoted to a presentation of the Acumen environment pointing out the most interesting concepts and constructs allowing for this study. In Section 4 we give examples of implementing intelligent agents in Acumen and we discuss advantages and pitfalls detected in this work. Finally, Section 5 closes the paper with some concluding remarks.

2 Related Work

Though the hybrid systems community has not adopted intelligent techniques, recently, some interest on these techniques has been manifested by a number of researchers in the area of control systems. A number of recent publications on this topic can be found in the literature. In [12] Moe et al. give an overview and describe the state of the art concerning the use of machine learning techniques in control systems. Javadi-Moghaddam and Bagheri in [6] present an adaptive neuro-fuzzy genetic algorithm based on sliding-mode control system for a remotely operated vehicle with four degrees of freedom. A survey of Evolutionary Algorithms in Control Systems Engineering is presented by Fleming and Purshouse in [5]. Some older research includes the work of Lemmon and Anstaklis [10] on hybrid systems and their relation to intelligent control and the work of Lee and Kim in [9] describing the use of a neural network for designing an adaptive controller applied to turbulent channel flow for drag reduction.

However, the most significant part of this research deals with use of Reinforcement Learning (RL) a concept in Artificial Intelligence which corresponds to Neuro-Dynamic Programming. In Koch et al. [8] the authors study the use of RL for the control of UAV. More specifically, they investigate the performance and accuracy of the inner control loop providing attitude control when using intelligent flight control systems trained with state-of-the-art RL algorithms, Deep Deterministic Policy Gradient, Trust Region Policy Optimization, and Proximal Policy Optimization. In [20] Wang et al. present a novel approach to feedback control with deep RL. In a review paper [3] Bosoniu et al. present important advances in RL for control concerning performance, stability analysis, and deep

approximators. Recently, Quade et al. [15] archived their research on explainable machine learning on control issues such as robust control and stability analysis.

Moreover, intelligent approaches are more and more used in various applications of CPS's. For instance, one may cite the review paper [16] of Radanliev et al. on artificial intelligence in cyber physical systems and references therein. We may, also, cite the archived review paper [14] of Olowononi et al. on security issues of machine learning for networked CPSs. In addition to these, one may cite the most recent special issues by journals such as Computer Communication: Special Issue on AI-Driven Sensing and Computing for Cyber-Physical Systems and the special issue on Artificial Intelligence and Cyber-Physical Systems by the ACM Transactions on Cyber Physical Systems. This literature review highlights the potential interest in studying ways of bridging the gap between hybrid systems and intelligent techniques. Starting from the simple interaction, between these two approaches, one may envisage the implementation of intelligent techniques using hybrid systems in a unique environment. Hence, studying the infrastructure necessary to support both approaches is an important issue.

In the context of this paper considering intelligent behavior and hybrid systems implies taking into account the environments and/or languages available for designing hybrid systems and how it is possible to implement intelligent behavior on these environments. In these terms it seems that the MATLAB© computing environment together with its Toolboxes such as Simulink, Control System, Neural Networks, etc stand above other available tools as it disposes numerous libraries both for designing hybrid systems as well as for implementing machine learning techniques. Probably, besides cost restrictions, the most serious inconvenience is that all these tools are proprietary and they are not open to user-made modifications.

Another well known environment for hybrid systems is Modelica; an equational modeling language implemented with different front-ends as a simulation computing engine which can be interfaced with other external libraries. As an example, Modelica has been used by Schaub et al. [17] in order to simulate autonomous vehicles using intelligent agents. Moreover, Lukianykhin et al. [11] proposed an approach which allows connecting models using a Functional Mock-up Interface to the OpenAI Gym toolkit. The objective is to exploit Modelica equation-based modeling and co-simulation together with RL algorithms. While such interface capabilities are not available in Acumen, to the best of our knowledge, Modelica constitutes a modeling environment displaying similar functionality as Acumen.

Finally, creating a modeling environment from scratch, in any of the well known programming languages (Python, Scala, etc.), constitutes an alternative possibility which proves to be an extremely time consuming option. Typically, programming environments offer the possibility to develop tools for simulating intelligent behavior which can be used in conjunction with integrated environments. This underlines the fact that integrated environments incorporating intelligent functionality are not commonplace.

3 Acumen: A Hybrid Systems Modeling and Simulation Environment

3.1 Description of the environment

Acumen [4] is an experimental Domain-Specific Language (DSL) written in Scala [13], a programming language that runs on top of the Java Virtual Machine (JVM). It is distributed as an Integrated Developing Environment (IDE) and provides a wide range of tools to the user, some of which include a code editor, error reporting and visualization of 3D animations.

The purpose of the language is to model and simulate hybrid systems, which are dynamical systems that can exhibit discrete, continuous or hybrid (discrete and continuous) behavior. Although small in size, the language provides a variety of expressions to the user allowing for simulation of larger systems, as well as the ability to model systems that contain multiple variables dynamically changing over time. Originally, Acumen was motivated by the absence of robust developing tools for modeling CPS. There is an obscure gap between verification tools and running simulators [21], and Acumen was built as an experimental bridge between the two. At the highest level, the leading goal of the language is to explore ways of designing a "rigorous-but-practical" tool for hybrid systems developers, as long as it encapsulates mathematical correctness, practicality and scalability [18].

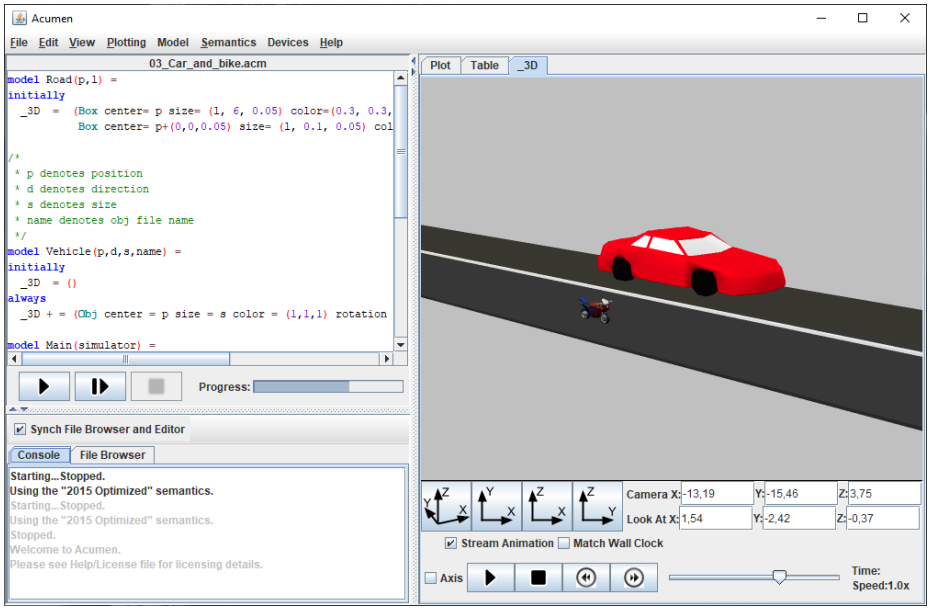

Fig. 1. Layout of the Acumen environment.

3.2 Conceptual Approach

Acumen uses the notion of *models* as its core data structure. Hereafter, we will denote an Acumen model by A-model. Each A-model may consist of two sections, which are the *initially* and the *always* section. The *initially* section takes care of the initialization of all the variables used by the A-model. On the other hand, the *always* section describes the dynamic behavior of the model over time by means of simple or conditional statements. We need to note, here, that statements in Acumen are executed in parallel, i.e. at the same time, and so *the order of such statements is irrelevant*. In this sense Acumen statements do not follow the semantics of statements met in classical programming languages and other computing environments. In order to depict the conceptual structure of Acumen, highlighting the constructs which permit to model and simulate hybrid systems, let us consider a concrete example of modeling and simulating a bouncing ball.

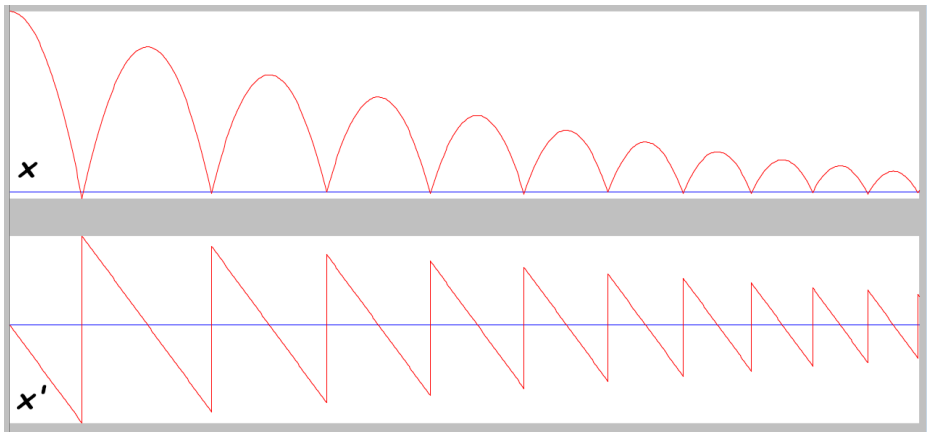

Fig. 2. Bouncing ball model. Evolution of the position and its derivative.

In Figure 2, we see the Acumen result while executing the A-model configuration for a bouncing ball. This A-model is rather simple and can be described by an Acumen 3-D object, such as a sphere, and a position variable x with its respective first and second derivatives, $\dot{x}$ for the rate of change of x and $\ddot{x}$ for the gravity acceleration. Such expressions can be easily modeled in Acumen by writing the code as shown in Figure 3. This Acumen example highlights the way one may model a hybrid system such as a bouncing ball. Continuous behavior concerning change in the position of the ball over time is described by the two differential equations while the discrete event corresponding to the ball touching the ground is formulated by the *if* statement.

Another notable example is a hybrid dynamical system in intelligent control [10], that is a system consisting of a continuous-state plant interfaced to a discrete event supervisor. The supervisor of the hybrid system is the component deriving

```
model Main(simulator) =
initially
  x = 3 , x' = 0 , x'' = -9.8,
  _3D  =  ()
always
  x''=-9.8,
  if x<0 && x'<0 then
    x' += -0.9*x'
  noelse,
  _3D =(Sphere center=(0,0,x) size=0.2 color=(1,0,0) rotation=(0,0,0))
```

Fig. 3. Acumen algorithm for the bouncing ball model.

and setting provably correct goals for the control function. These goals are the result of decisions taken by a subsystem or a function implementing some form of intelligent decision support. Studying the feasibility to represent such intelligent behavior in a modeling and simulation environment for hybrid systems such as Acumen [18] results in investigating a) the expressive power of the environment and the type of intelligent behavior that it is possible to represent, b) the level of interaction that is possible to achieve between continuous and discrete time components and c) the changes and upgrades needed for the language constructs in order to fully support the previous points.

4 Experiments and Results

In order to achieve the goals set by the previous items we choose to implement two specific intelligent techniques, namely, RL and Hopfield neural networks. The reason for these choices is that these techniques have both a continuous and a discrete time version. Moreover, the problem chosen for RL is the exploration of a 2-dimensional maze which requires its representation by means of discrete constructs. It is thus possible to investigate the performance of Acumen in both continuous and discrete time together with some framework triggering some discrete time events.

4.1 Q-learning Intelligent Agent

Let us recall that RL is a machine learning technique used to solve complex optimization problems in uncertain/unknown environments. The approach uses agents to explore the unknown environment by taking actions [7] and giving them rewards based on the progress of the optimization task. So, during this process agents are able to adjust and optimize their behavior through a feedback mechanism based on the rewards strategy [19]. The Q-learning algorithm used in this experiment is described by Equation (1).

$$Q^{new}(s_t, a_t) \leftarrow Q^{old}(s_t, a_t) + \alpha * (r_t + \gamma * maxQ(s_{t+1}, \alpha) - Q^{old}(s_t, a_t)). \quad (1)$$

In our implementation an intelligent agent traverses and learns its way to a goal position. This agent is an abstract entity implemented using two A-models. The first A-model, which is called the "Main Model", simulates the maze environment and runs the Q-learning algorithm, while being in charge of reacting to the different actions and providing the appropriate rewards. The second A-model is the "Moving Model" which navigates in the maze environment. By default operation of the agent for these two functions is continuous in time but this specific implementation uses discrete time steps. The reason for this choice is justified by the fact that in a continuous time setup for Q-learning we would have to compute the terms of Equation (1) in continuous time. Yet, it is known that typical implementations of this strategy use a neural network. The following Figure 4 outlines the maze used for this experiment.

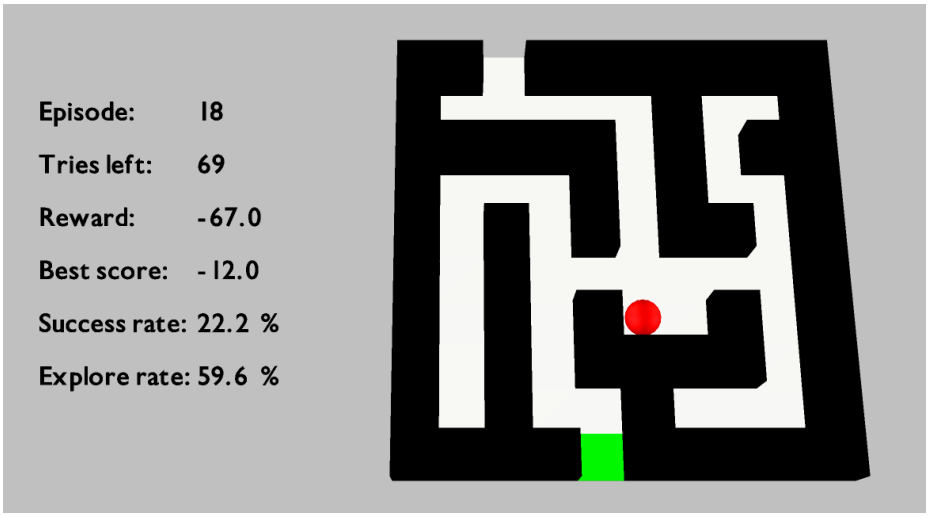

Fig. 4. Simulation of a maze environment in Acumen.

In addition to the above an A-Model is needed for setting up and updating the 2-dimensional maze, represented in Acumen using 3-dimensional objects. This is done by initializing an A-model called the Environment. This is a static A-model responsible for creating all the 3-dimensional objects needed to visualize the maze layout for the simulation. The A-model configuration corresponding to the Moving Agent instantiates parameters tied to the agent's behavior such as the current and the reset positions, the moves executed and the accumulated reward. It also creates a 3D object, i.e. the red sphere, in order to visualize the agent. On the other hand the A-model related to the Main Model comprises the Acumen code for executing the Q-learning algorithm while communicating with the Environment and the Moving Model for retrieving its position and moving behavior.

In conclusion, this set up comprises two concurrent A-Models responsible for carrying out two different functions of the intelligent agent, as well as a static A-model for the objects forming the maze. The Moving Model handles the motion of the red sphere while the Main Model is responsible for updating the Q-table with the appropriate values guiding the motion of the Moving Agent. This experiment permits to exploit Acumen language constructs mainly for implementing intelligent agents by investigating alternatives on how to implement the Q-learning algorithm. Another option would be to define two different agents one represented by the Main Model and a second one represented by the the Moving Model. This architecture assigns a specific task to each agent thus breaking the intelligent agent into two communicating agents. Despite the fact that interaction between A-Models is supported by Acumen by means of shared memory access and concurrent execution of the A-Models, we believe that a supplementary level of abstraction in the definition of agents would add to the complexity of the representation without, essentially, solving any problem.

Finally, implementation of Q-learning requires the creation of a Q-Table which describes the quality of the different actions of the agent in any state. Hence, implementation of Q-Table in Acumen requires a suitable data structure while the actions described in Q-Table need to be implemented by a set of discrete states, e.g. a form of a matrix of states. So, Acumen should be able to allow for the creation of discrete structures, such as matrices, while permitting direct or indexed access to the elements of these structures. However, it was found that such constructs were not part of the specifications of Acumen and so we had to undertake a number of significant modifications to endow Acumen with this functionality.

4.2 Simple Hopfield Networks

In this experiment we studied the simulation of two Hopfield network models in order to obtain hands-on experience with the implementation of intelligent models with dynamic behavior. It is well known that Hopfield networks are recurrent artificial neural networks which are able to simulate associative memory systems with binary output neurons. The dynamics of the Hopfield network can be studied either in discrete time steps or in continuous time flow.

For this case study we implemented both discrete and continuous time models pre-trained as it was not within our goals to verify how the Hebbian update rule modifies the networks' weights. Simulation of both models gave the expected results with the continuous model exhibiting for each neuron smooth transition between the extreme values of the bipolar function *tanh* used for the activation function of the neurons. Figure 5 displays the output of each model after convergence.

$$C_t \frac{dx_i(t)}{dt} = -\frac{x_i(t)}{R_i} + \sum_{j=1}^{N} w_{ij}\phi_j(x_j(t)) + I_i. \tag{2}$$

The differential equation for the continuous model is given by Equation (2) and the corresponding Acumen code is shown in Figure 6. This model configuration

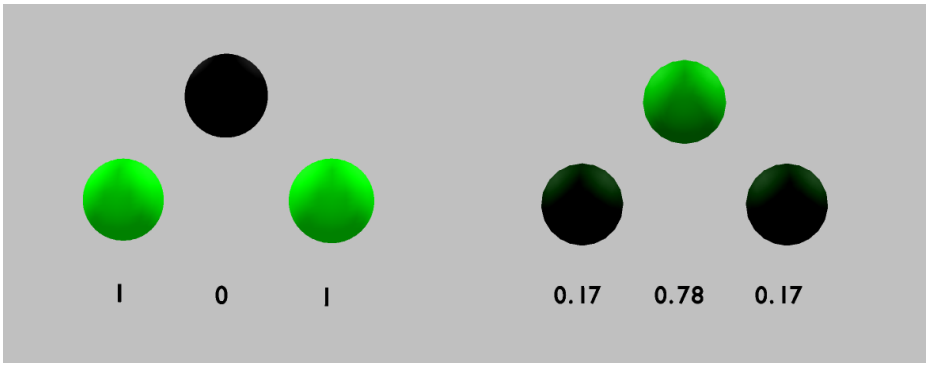

Fig. 5. Hopfield models in Acumen (Left: Discrete, Right: Continuous)

displays how Acumen handles randomness and the reaction based on the different random numbers which leads to the appropriate use of differential equation for the specific neuron. In this experiment we investigated the possibility to

```
always
  n0.output = N(0),
  n1.output = N(1),
  n2.output = N(2),
  match rnd with [
    0 -> N'(0) = (-(N(0)/R) * (W(0,1)*tanh(N(1)) + W(0,2)*tanh(N(2))) + theta)/C,
         N'(1) = N'(1),
         N'(2) = N'(2)
  | 1 -> N'(1) = (-(N(1)/R) * (W(1,0)*tanh(N(0)) + W(1,2)*tanh(N(2))) + theta)/C,
         N'(0) = N'(0),
         N'(2) = N'(2)
  | 2 -> N'(2) = (-(N(2)/R) * (W(2,0)*tanh(N(0)) + W(2,1)*tanh(N(1))) + theta)/C,
         N'(0) = N'(0),
         N'(1) = N'(1)
  ],
  rnd += argrand((0,1,2))
```

Fig. 6. Acumen algorithm for the continuous Hopfield model.

implement both a continuous and a discrete time intelligent model such as the Hopfield network. In both cases Acumen provided the necessary constructs to successfully model and simulate both models thus providing the expressive power suitable for implementing both discrete and continuous behavior. However, we need to note that Acumen is not a computational environment giving the possibility to define and execute vector equations of any form. This constitutes a minor disadvantage which, again, is related to the lack of suitable data structures with direct or indexed access in Acumen.

The main result of the experiments performed is that Acumen proves to have the constructs necessary to implement intelligent behavior both for continuous and discrete time models. This is achieved using the *model* construct which encapsulates both the dynamics of an agent and the static elements related to its outlook or the environment. *Models* in Acumen are executed concurrently thus simulating parallelism. Information is exchanged between *models* by means of accessing shared memory and so there is no need of some specific communication or message passing mechanism. Different behaviors of the same agent can be implemented using specific Acumen *models* which co-exist and act as whole for simulating the agent as a single entity.

5 Concluding Remarks

In this paper we, experimentally, evaluated the potential of a hybrid systems design and simulation environment such as Acumen, to support implementation of intelligent behavior by means of independent interacting models. However, while Acumen has the necessary expressive power for such implementations it lacks some elementary but necessary data structures of direct or indexed access to their elements. This was the main obstacle we faced. The add-ons built to remedy this disadvantage proved to be insufficient as these constructs were not within the requirements set for Acumen. We need to note that either Acumen or any other similar environment should offer such data structures in order to permit building more complex intelligent behavior such as fully interacting agents. The level of interaction between intelligent modules and hybrid systems is defined by the interaction set by Acumen for its models.

The experiments carried out permit to draw conclusions on the possibility to represent intelligent techniques using hybrid systems modeling tools, thus bringing closer these two different concepts. Though these partial conclusions are hopeful we consider that effective cooperation of intelligent techniques and hybrid systems is feasible for specific tasks and problems under the essential requirement that intelligent approaches do not set serious problems on issues such as stability, convergence and optimality of the control systems. Finally, in general terms we may conclude that this study can be applied to draw conclusion for a wider spectrum of modeling and simulation applications such as intelligent robotic systems, intelligent drones etc.

Acknowledgments

The authors would like to thank the anonymous reviewers for their suggestions and comments on earlier version of the manuscript, that helped to improve the paper at hand.

References

1. Alur, R., Courcoubetis, C., Halbwachs, S., Henzinger, T., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical computer science **138**(1), 3–34 (1995)

2. Antsaklis, P.J.: On Intelligent Control: Report of the IEEE CSS Task Force on Intelligent Control. Tech. rep., University of Notre Dame (January 1994)
3. Busoniu, L., de Bruin, T., Tolić, D., Kober, J., Palunko, I.: Reinforcement learning for control: Performance, stability, and deep approximators. Annual Reviews in Control **46**, 8–28 (2018)
4. Effective Modeling Group: Acumen modeling language. http://acumen-language.org
5. Fleming, P., Purshouse, R.: Evolutionary algorithms in control systems engineering: a survey. Control Engineering Practice **10**(11), 1223–1241 (2002)
6. Javadi-Moghaddam, J., Bagheri, A.: An adaptive neuro-fuzzy sliding mode based genetic algorithm control system for under water remotely operated vehicle. Expert Systems with Applications **37**(1), 647–660 (2010)
7. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research **4**, 237–285 (1996)
8. Koch, W., Mancuso, R., West, R., Bestavros, A.: Reinforcement Learning for UAV Attitude Control. ACM Transactions on Cyber-Physical Systems **3**(2) (feb 2019)
9. Lee, C., Kim, J., Babcock, D., Goodman, R.: Application of neural networks to turbulence control for drag reduction. Physics of Fluids **9**(6), 1740–1747 (1997)
10. Lemmon, M., Antsaklis, P.: Hybrid systems and intelligent control. In: Proceedings of 8th IEEE International Symposium on Intelligent Control. pp. 174–179 (1993)
11. Lukianykhin, O., Bogodorova, T.: ModelicaGym: applying reinforcement learning to Modelica models. In: Proceedings of the 9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools. pp. 27–36 (2019)
12. Moe, S., Rustad, A.M., Hanssen, K.G.: Machine learning in control systems: An overview of the state of the art. In: Bramer, M., Petridis, M. (eds.) Artificial Intelligence XXXV. pp. 250–265. Springer International Publishing, Cham (2018)
13. Odersky, M., Spoon, L., Venners, B.: Programming in scala. Artima Inc (2008)
14. Olowononi, F.O., Rawat, D.B., Liu, C.: Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS. IEEE Communications Surveys & Tutorials **23**(1), 524–552 (2021)
15. Quade, M., Isele, T., Abel, M.: Explainable Machine Learning Control – robust control and stability analysis (2020)
16. Radanliev, P., De Roure, D., Van Kleek, M., Santos, O., Ani, U.: Artificial intelligence in cyber physical systems. AI & society pp. 1–14 (2020)
17. Schaub, A., Hellerer, M., Bodenmüller, T.: Simulation of artificial intelligence agents using modelica and the dlr visualization library. In: Proceedings of the 9th International MODELICA Conference. pp. 339–346 (2012)
18. Taha, W., Duracz, A., Zeng, Y., Atkinson, K., Bartha, F.A., Brauner, P., Duracz, J., Xu, F., Cartwright, R., Konečnỳ, M., et al.: Acumen: An open-source testbed for cyber-physical systems research. In: International Internet of Things Summit. pp. 118–130. Springer (2015)
19. Van Otterlo, M., Wiering, M.: Reinforcement learning and markov decision processes. In: Reinforcement Learning, pp. 3–42. Springer (2012)
20. Wang, Y., Velswamy, K., Huang, B.: A Novel Approach to Feedback Control with Deep Reinforcement Learning. IFAC-PapersOnLine **51**(18), 31–36 (2018), 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018
21. Zhu, Y., Westbrook, E., Inoue, J., Chapoutot, A., Salama, C., Peralta, M., Martin, T., Taha, W., O'Malley, M., Cartwright, R., et al.: Mathematical equations as executable models of mechanical systems. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems. pp. 1–11 (2010)