DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch112

Boolean Network Robotics as an Intermediate Step in the Synthesis of Finite State Machines for Robot Control

Lorenzo Garattoni[1], Andrea Roli[2], Matteo Amaducci[2], Carlo Pinciroli[1] and Mauro Birattari[1]

[1]IRIDIA, Université Libre de Bruxelles, Belgium
[2]DISI-Cesena, *Alma Mater Studiorum* Università di Bologna, Italy
lorenzo.garattoni@ulb.ac.be

Abstract

We propose an approach to the automatic synthesis of robot control software based on the finite state machine (FSM) formalism. In our previous research, we have introduced Boolean network robotics as a novel approach to the automatic design of robot control software. In this paper, we show that it is possible to leverage automatically designed Boolean networks to synthesize FSMs for robot control. Boolean network robotics exhibits a number of interesting properties. Firstly, notwithstanding the large size of the state space of a Boolean network and its ability to display complex and rich dynamics, the automatic design is able to produce networks whose trajectories are confined in small volumes of the state space. Secondly, the automatic design produces networks in which one can identify clusters of states associated with functional behavioral units of the robots. It is our contention that the automatic design of a Boolean network controller can be a convenient intermediate step in the synthesis of a FSM, which offers the advantage of being a compact, readable, and modifiable representation. In this paper, we show that clusters of states traversed by network trajectories can be mapped to states of a FSM. We illustrate the viability of our proposal in two notable robotic tasks, namely collision avoidance and sequence recognition. The first task can be achieved by a memoryless control program, whilst in the second the robots need memory.

Introduction

The automated design of compact high-level representations of control software for robots is a challenge in artificial intelligence. Through methods of automatic design, a robot learns a behavior without the explicit intervention of the developer. Automatic design methods offer advantages with respect to manual methods in terms of robustness and generality of the design process. Moreover, the space of solutions explored by automatic techniques is larger and less constrained than that explored by methods of manual design (Koza et al., 2003; Lipson, 2005). However, the effectiveness of automatic methods depends on a number of aspects such as the definition of the search space and the existence of a predictive simulation of the system. Yet, such methods do not provide guarantees on the solution optimality. Automatic design techniques act iteratively on the robot control software in order to reach a configuration that fulfills the requirements. There exist several ways of representing control software of robots, but the finite state machine (FSM) formalism is the oldest and is broadly used.

Our intention is to propose a new way to automatically design FSMs representing control software for robots. The new approach exploits the work carried out in our previous research in the field of Boolean network robotics (Roli et al., 2011) as an intermediate step in the automatic design of FSMs. In the following, we first give an overview of the existing methods for the automatic design of robot control software and then we introduce the original contribution of our work.

Among the existing approaches to automatically obtain a FSM of a control software, evolutionary programming is one of the most notable (Fogel, 1962, 1993). EP is a paradigm used for the generation of programs, code, algorithms and structures in general, by means of variation and selection mechanisms inspired by natural evolution. Although EP was shown to produce interesting results in many important applications, several issues are still open about its employment (O'Neill et al., 2010). One of the main issues is the choice of the most appropriate representation for the programs to be evolved. In fact, the most suitable representation and the appropriate encoding of the programs into individuals in the evolution process are critical aspects for the performance of EP (Petrovic, 2007). Moreover, EP normally requires the definition of constraints to contain the size of the FSM.

Besides the evolution of FSMs, most of the effort in the field of automatic design of robot control systems has been concentrated around artificial neural networks (NNs) (Nolfi and Floreano, 2000). While NNs offer advantages such as high plasticity and adaptability, they are black-boxes and it is often very difficult to analyze their dynamics. The dynamical behavior of a NN can be modeled by a system of differential equations. There exists a number of works that show how the mathematical tools of dynamical systems theory can be used to gain significant insight into the dynamics of small continuous-time recurrent neural networks (Beer, 1995; Ya-

mauchi, 1993; Yamauchi and Beer, 1994). However, when the number of neurons is greater than a few units, the analysis becomes too complex to handle and only qualitative or approximate studies are possible.

In our previous work (Roli et al., 2011), we have introduced Boolean network robotics as a novel approach to the automatic design of robot control software. Boolean networks (BNs) are a model of genetic regulatory networks (Kauffman, 1969). BNs are extremely interesting from an engineering perspective because of their ability to display complex and rich dynamics, despite the compactness of their description and the simplicity of their implementation. BN dynamics can be studied through traditional dynamical system methods (Bar-Yam, 1997; Serra and Zanarini, 1990). The use of concepts such as state space, trajectories and attractors, combined with the discrete nature of BNs, enables non-trivial analysis of the dynamical behavior. Such ease of analysis is one of the strengths of BN systems.

In this paper, we continue the line of research in Boolean network robotics showing that the automatic design of a Boolean network controller can represent an intermediate step in the synthesis of a FSM. The intuition stems from the analysis we carried out on the trained Boolean networks and two interesting properties it revealed. First, the automatic design shapes the network dynamics in very limited volumes of the state space. Second, such dynamics are structured in sets of clusters of states associated with functional behavioral units of the robots. On the basis of such properties, we propose a heuristic to map those clusters into states of a FSM, which offers a compact, readable, modifiable, and formally verifiable representation.

In this work, we propose a proof of concept of our proposal applying Boolean network robotics to two robotic tasks, i.e., corridor navigation and sequence recognition. The first task is a typical collision avoidance behavior and consists in moving along a corridor avoiding walls and objects; this task can be attained by a robot equipped by a memoryless control software. Conversely, the second tasks presents a sequence-recognition scenario (Sun and Giles, 2001). The complexity of the target task lies in the fact that it requires the robots to have memory of the past in order to choose the next actions to perform.

The analysis of the trained networks confirms the two properties mentioned and allows for simple mapping between clusters of states in the state space of a BN and states of a FSM.

Despite the simplicity of the two tasks, this work represents a first crucial step towards the definition of an automatic design method of FSMs that exploits Boolean network robotics as a convenient intermediate step.

Boolean network robotics

In this section we first introduce BNs and then we describe how they are employed and configured to let robots perform the desired tasks.

Boolean networks

A Boolean network is a discrete-state and discrete-time dynamical system. Its structure is defined by an oriented graph with N nodes each associated to a Boolean value x_i, $i = 1, ..., N$, and a Boolean function $f_i(x_{i_1}, ..., x_{i_{K_i}})$, where K_i is the number of inputs of node i. The arguments of function f_i are the Boolean values of the nodes whose outgoing arcs are connected to i. The state of the system at time t, with $t \in \mathbb{N}$, is defined as the vector of the N Boolean values at t. The state space size is 2^N. Several update schemes can be defined (Gershenson, 2004), but the most studied is characterized by synchronous and deterministic operations.

BN dynamics can be studied by means of the usual dynamical system methods (Bar-Yam, 1997; Serra and Zanarini, 1990), hence the usage of concepts such as state space, trajectories, attractors and basins of attraction. Recently, the attention of the scientific community has focused on the employment of efficient mathematical and experimental methods for analyzing network dynamics and thus have insight into the behavior of a BN system (Fretter and Drossel, 2008; Ribeiro et al., 2008; Serra et al., 2007).

BN-Robot coupling

To design a BN-based robot control system, we first need to couple the BN to the robot so as to let the BN dynamics guide the robot behavior. For this purpose, some nodes of the network are given special roles. More precisely, we define a set of input nodes and a set of output nodes. This choice characterizes our approach with respect to most of the work performed about BNs, in which they are considered as isolated systems, even though some notable exceptions exist (Ansaloni et al., 2009; Dorigo, 1994; Kauffman, 1991; Patarnello and Carnevali, 1986). The Boolean values of the input nodes are not determined by the network dynamics, but they are imposed according to the robot sensor readings. Similarly, the values of the network's output are used to encode the signals for maneuvering the robot's actuators. Several ways to define the mapping between sensor readings and network's input, and between network's output and actuators are possible. However, the most natural way is to define the mapping via a direct encoding. Figure 1 shows the coupling between BN and robot.

Automatic design methodology

Once a mapping between the BN and the robot is defined, the BN must be designed in order to control the robot's behavior. Our approach consists in treating BN design as a search problem. In fact, the design of a BN that satisfies given criteria can be modeled as a constrained combinatorial optimization problem by properly defining the set of decision variables, constraints and the objective function.

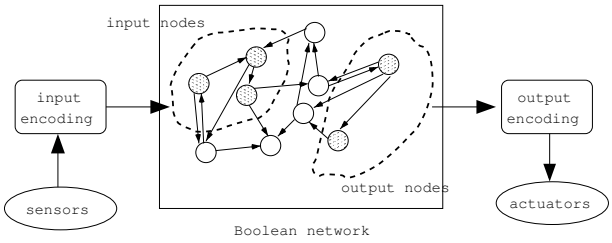

Figure 1: The coupling between BN and robot.

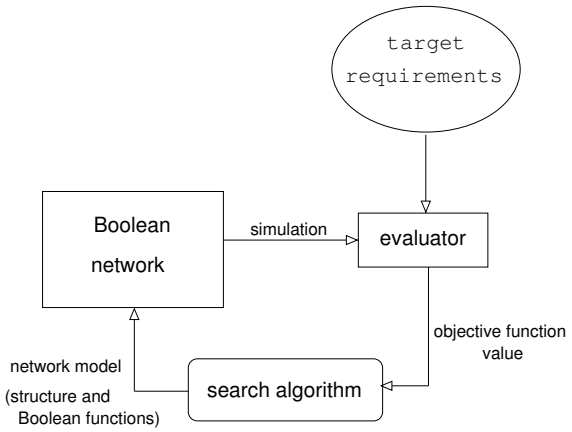

Figure 2: BN design process.

The search algorithm manipulates the decision variables which encode structure and Boolean functions of a BN. A complete assignment of these variables defines an instance of a BN. Then, we couple this network to the robot through the input-output mapping, and subsequently we execute the network. The evaluation of the network at each iteration of the search process is performed in a batch of simulated experiments. The performance of the robot in each experiment is assessed according to a user-defined objective function, which associates the robot behavior to a numeric evaluation. Finally, the search algorithm exploits this value of performance to proceed with the design process. In particular, the algorithm changes the configuration of the decision variables so as to find networks with better performances. This process is depicted in Figure 2.

Robot tasks

We addressed two test cases with different characteristics: the first, corridor navigation, is a memoryless task while the second, sequence recognition, requires memory. It is interesting to analyze how the nature of the task influences the organization of the state space in the trained networks. In particular, our analysis aims to determine whether the state space of the trained networks exhibit the same properties independently of the nature of the task. These properties, which are *(i)* the compression of the dynamics in limited regions of state space, and *(ii)* the organization in clusters of

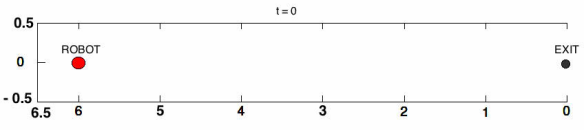

Figure 3: Corridor navigation environment.

states associated with behavioral units of the robots, enable the exploitation of the trained networks as intermediate step in the synthesis of FSMs.

In the remainder of this section we present the working environment and describe the two test cases.

Robot and Simulator

For both test cases, the robots are trained in simulation. The simulation framework we employed is the open source simulator ARGoS (Pinciroli et al., 2012). ARGoS is a discrete-time, physics-based simulation environment that provides a faithful simulation of the behavior of different robotics platforms.

The robot simulated in our test cases is the *e-puck* (Mondada et al., 2009). The e-puck is a small wheeled robot, designed for research and educational purposes. It has a cylindrical body of 7 cm of diameter, equipped with a variety of sensors. For our test cases, we use the 8 infra-red proximity sensors placed along the circular perimeter of the robot and the 3 infra-red sensors pointed directly at the ground in front of the robot. The 3 latter sensors can be used to detect the color of the ground, in greyscale. The actuators utilized, besides the motors of the two wheels, are the 8 red LEDs.

Corridor navigation

The first test case is designed to explore the features of networks able to perform a memoryless task. It consists of a robot that must navigate along a corridor avoiding any collision with the walls and finally reach the exit.

Environment: it consists of a straight corridor of 6.5 m in length and 1 m in width.

Task: at the beginning, the robot is placed within the corridor 6 m far from the exit. During the experiment, the robot must advance along the corridor, avoiding collisions and finally, within the given total execution time $T = 120$ s, reach the exit. See Figure 3 for a representation of the environment at the beginning of the experiment.

During the execution, if a collision between the robot and the walls of the corridor occurs, the experiment is immediately stopped.

Performance measure: the performance assigned to the robot is simply its final distance from the exit (normalized). The smaller is this distance, the better is the performance of the robot.

BN-robot setup: for successful navigation, the robot needs the 8 proximity sensors to detect the walls and avoid them.

At each time step, the readings of the 8 sensors are encoded into the values of the BN input nodes. We use 4 input nodes to encode the readings of the proximity sensors. Thus, the 8 proximity readings are gathered in pairs. If at least one of the two sensors of the pair exceeds a chosen threshold, the corresponding input node value is set to 1. The pairs are formed to allow the robots to detect walls in the four directions north-east, south-east, south-west and north-west.

Once the readings of the sensors are encoded in the input nodes, the network's state is updated and finally the values of the output nodes are read, decoded and utilized to set the actuators. We use two output nodes to set the wheel speeds either to zero or to a predefined, constant value.

For this test case, we set the network size to 20 nodes. We leave the analysis on how this value affects performance for future investigation.

BN design: the initial topology of the networks, i.e., the connections among the nodes, is randomly generated with $K = 3$ (i.e., each node has 3 incoming arcs) and no self-connections, and it is kept constant during the training. The initial Boolean functions are generated by setting the 0/1 values in the f_i uniformly at random. Our search process, which is a stochastic descent, works only on the Boolean functions. In particular, at each iteration, the search algorithm changes the configuration of the network by flipping one bit of the Boolean functions. The flip is performed by changing a random entry in the f_i, where i is a randomly chosen node. The new configuration is accepted if the corresponding BN-robot system has a performance at least equal to the current one. The evaluation of each network is performed on a set of initial conditions, that form the training set. For this test case, the training set is composed of six different initial orientations of the robot. The six angles are chosen so as to have six equally spaced orientations in the range between $\frac{\pi}{3}$ and $-\frac{\pi}{3}$ (with 0 that is the straight direction of the robot towards the exit). In this manner, the robot must be able to cope with a wide range of different situations and avoid the walls it detects in any direction. The final evaluation assigned to the robot is computed as the average of the performance across the 6 trials. We executed 100 independent experiments, each corresponding to a different initial network. In each experiment, we run the local search for 1000 iterations.

Sequence recognition

The second test case aims to explore the properties of networks able to perform a task that requires memory. The task is sequence recognition (Sun and Giles, 2001). In particular, the robot must learn to recognize a sequence of colors by performing certain actions. This kind of task is more complex than the previous one, because the robot needs a form of memory to be able to choose the next action depending on the past.

Environment: it consists of a straight corridor of 7 m in length and 1 m in width. Along the corridor, the ground is painted to form a striped pattern with three different colors: white (W) represents the background, while black (B) and gray (G) are the symbols of a sequence to be recognized.

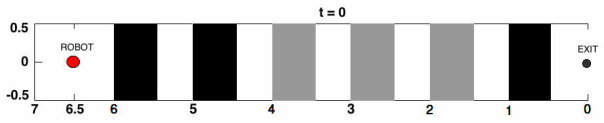

Figure 4: Sequence recognition corridor environment.

Task: at the beginning of the experiment, the robot is placed within the corridor 6.5 m far from the exit. During the experiment, the robot must move along the corridor and reach the exit. Every time the robot encounters a black or gray area in the right sequence, it must turn its LEDs on. Conversely, when the robot encounters the background color or other colors in the wrong order, it must keep its LEDs off. The sequence to be recognized is a cyclic repetition of black followed by gray. By performing the right sequence of actions while moving along the corridor, the robot must be able to reach the exit within the given total execution time, fixed in T=130 s. Figure 4 represents an example of the environment at the beginning of the experiment.

In the environment depicted in Figure 4, the robot must perform the following sequence of actions to achieve the goal (omitting the background color (W) whose corresponding LED correct status is always OFF):

```
Colors along the corridor
B    B    G    G    G    B
ON   OFF  ON   OFF  OFF  ON
Robot's LEDs correct status
```

If the robot, at any instant in time during the execution, performs the wrong action, the experiment is immediately stopped.

Performance measure: The performance assigned to the robot is the final distance from the exit of the corridor (normalized between 0 and 6.5). The value must be minimized.

BN-robot setup: for this task, the robot needs the ground sensor to detect the color of the ground. For our simple application we use only the central sensor. Since we encode three values (W, B, G), at each time step, the reading of the sensor is encoded into the values of two BN input nodes. We use four nodes to encode the proximity sensors that, even though not strictly needed for the task, can be still useful for the navigation along the corridor.

After the network's state update, we decode and use the values of the output nodes to set the actuators. Besides the two nodes used to control the wheel speeds, an additional output node is utilized to set the state of the LEDs either to ON or OFF.

For this test case we increased the network size to 30 nodes.

BN design: initial topology and Boolean functions are randomly generated with $K = 3$. In our experiments, the search strategy is a stochastic descent and works only on the Boolean functions, leaving the topology unchanged. The evaluation of each network is performed on a set of initial conditions. More precisely, the training set is composed of 10 different randomly generated sequences of colors on the ground. Differently from the corridor-navigation case, the robot starts always pointing towards the exit. In this way, the navigation task is simplified so as to focus the complexity on the sequence recognition. The final evaluation of a robot is the average value of the performance across the 10 trials. Due to the high computational cost required by each experiment, we executed only 30 independent experiments with 30 different initial networks for 100000 iterations of the search algorithm.

Analysis of the results

The analysis of the results obtained in both test cases reveals two properties. First, the dynamics of the automatically designed networks spans across a very limited region of the whole potential state space. This means that the search algorithm moves towards networks whose dynamics are compact. This relationship between the design process and the dynamical features of the networks is notable: the search algorithm acts directly only on the network structures, searching for a good behavior of the BN-robot systems while ignoring the dynamics property of the networks. Nevertheless, the analysis shows that the algorithm shapes and compresses indirectly the dynamics of the networks.

The second property observed is the organization of the state space traversed by the final networks in a set of clusters of states, each devoted to perform a specific series of actions.

In the remainder of this section we present the analysis and the results obtained for both test cases.

Corridor navigation analysis

Once the design process is completed, the focus of the study is on the dynamical features of the resulting networks. The first aspect we analyzed is the measure of the fraction of state space utilized by the trained networks. In order to carry out this analysis, we collected a large number of trajectories, corresponding to different initial conditions, for each BN obtained. Then, we counted the number of different states that each network traversed across all the trajectories and we reported the empirical cumulative distribution of the resulting values. Figure 5 shows the distribution for the corridor-navigation test case.

The plot shows that the final network dynamics traverse limited volumes of the state space. In fact, the median usage of state space in the 100 trained networks is located around 150 states. This is a very tiny fraction of the whole potential space, whose dimension is 2^N (2^{20} in this case). This first property enables an analysis of the network dynamics that allows to gain significant insight into the behavior of the BN-robot systems.

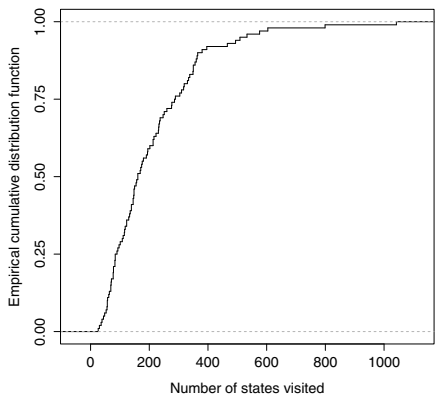

Figure 5: Empirical cumulative distribution function of the number of visited states in final networks. Corridor-navigation test case.

To analyze the organization of the dynamics of a BN controlling a robot, we collected its trajectories by simulating the experiment. Then, we gathered the trajectories and we generated the graph of the observed state transitions. For lack of space, the graphs can be found as on-line supplementary material (Garattoni et al., 2013).

The state space of the robot performing corridor navigation can be decomposed in three macro areas. One is responsible of the behavioral units that react to walls detected on the east side of the robot. Likewise, another cluster of states is devoted to avoid the obstacles on the west side of the robot. Besides, the two areas are both connected to a third cluster, responsible of moving the robot straight ahead as long as no obstacle is detected. Furthermore, it is possible to observe that each cluster of states contains few topical states, visited many times, and a series of other nodes gradually increasing in number and decreasing in visits. To verify this property, we performed the analysis of the graph for all the final networks of the corridor-navigation test case. We report in a plot the cumulative distribution of the fraction of states visited at least v times, where v is the number of visits on the x-axis. The results, showed in Figure 6 for a typical case, suggest that the dynamical behavior of a BN is built around few, prominent states that correspond to the main traits of the robot behavior.

The observations and the analysis presented so far suggest a procedure for deriving a representation of the robot's dynamics in the form of a FSM. We determine the states of the FSM by starting the observation from the topical states and gradually moving to the less important ones. The result is that a state in the FSM takes the place of a clusters of connected states in the state space in which the BN remains until

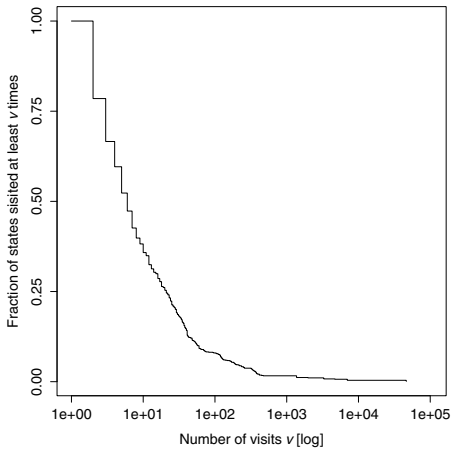

Figure 6: Distribution of the fraction of states visited at least v times. The fraction is computed with respect to the total number of states visited in 200 runs of 120000 time steps each. Corridor-navigation test case.

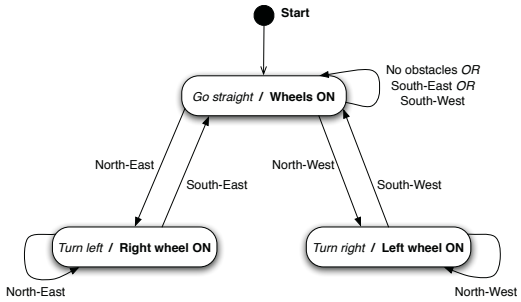

Figure 7: Finite state machine of the state space graph. Corridor navigation.

a specific input is received. By following this simple heuristic for a typical case of corridor navigation, we derived the FSM in Figure 7. We can observe that the automaton corresponds to a very simple yet effective behavior: the robot goes straight until an obstacle is detected on one side; in that case, the robot turns to the other side.

Sequence recognition analysis

The analysis carried out for the second test case is similar to that presented for the corridor-navigation test case. Due to the higher complexity of the sequence-recognition task with respect to the corridor navigation and the important computational cost required by each run, the number of successful networks to analyze is much lower than in the first test case. However, the analysis confirms the same properties.

The number of states visited by the trained networks is very low: the state space usage is on average 200 states out of 2^{30} potential states. By collecting the trajectories of the successful networks and generating the corresponding graphs of the observed state transitions, we observe that the limited region of state space utilized is again organized in sets of clusters of states. The graph derived in a typical case of sequence recognition can be found as supplementary material (Garattoni et al., 2013). At the top of the graph, a set of nodes allows the robot to navigate on the background with its LEDs off until the first colored stripe is found. Then, two clusters of nodes are responsible of the next action, depending on the detected color (turn LEDs on if black, turn LEDs off if gray). Once the first color has been recognized, the BN goes into a new region, dual to the first. Here, we find another area for the background color and two clusters of nodes for the black and gray with actions swapped with respect to the first region. When also the second color is recognized, the dynamics return back to the first area, reusing the same states to recognize a sequence of any length. This analysis shows that the memory, in our case the last color recognized, is stored in the state space in which the BN operates.

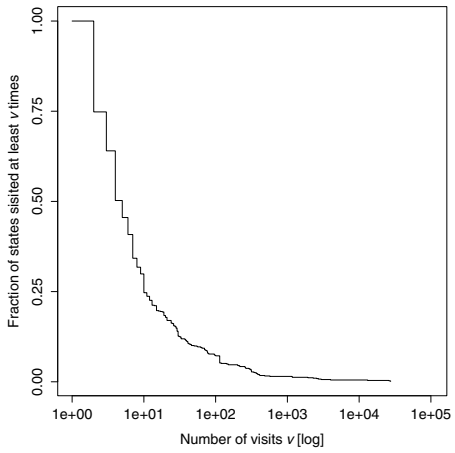

Figure 8: Distribution of the fraction of states visited at least v times. The fraction is computed with respect to the total number of states visited in 200 runs of 130000 timesteps each. Sequence-recognition test case.

Similarly to the corridor-navigation test case, each cluster of states is devoted to the execution of a particular functional behavioral unit of the robot. To support the observation and show that each cluster unfolds around few topical states and a series of other nodes gradually less important, we report in a plot the distribution of the fraction of states visited at least v times. The results for a typical case of sequence recognition are depicted in Figure 8.

The properties discussed so far allow the employment of the same heuristic used for the first test case to obtain a compact FSM representation of networks performing sequence recognition. From the graph described, we derived the FSM shown in Figure 9.

From the comparison of the FSM in Figure 9 and the one

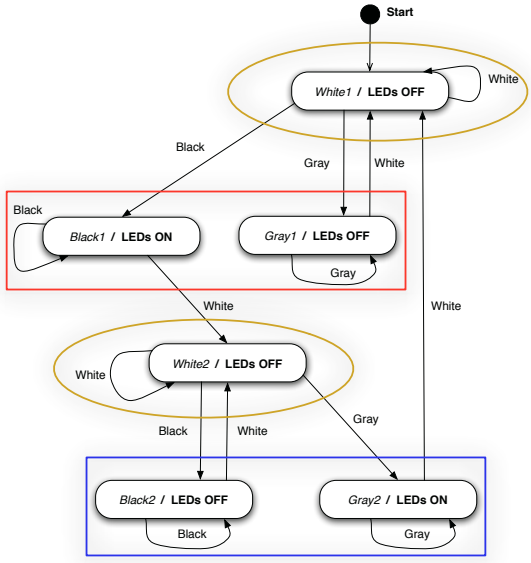

Figure 9: Finite state machine of the state space graph. Sequence recognition.

in Figure 7, it is possible to notice the influence of the different nature of the two tasks. The interesting aspect to highlight is the representation of the memory required by the sequence-recognition test case. In the corridor-navigation FSM, the action executed by the robot at each instant in time is determined only by the current observation of the world. This is due to the fact that corridor navigation is a memoryless task in which the robot is not required to keep memory of the past but it can simply react to the current stimuli of the environment. On the contrary, the FSM performing sequence recognition can activate different actions, e.g. LEDs on or LEDs off, for the same observation, e.g. the detection of the black color, depending on the previous state. Therefore, the memory that the robot needs to keep track of the last recognized color is stored in the phase space in which the BN operates. More precisely, the memory of the past is represented by the area of state space utilized by the network at a certain time, which is function of all the previous robot-environment interactions.

Conclusion

In this paper, we have exploited the properties of the automatic design of BN-robot control software to synthesize FSM representations of the robot program. This result has been made possible by an analysis performed on the state space of the best networks obtained at the end of the design process. In particular, the exploration revealed two crucial properties: *(i)* the trajectories of the BNs controlling the robots are confined in very small areas of the state space and *(ii)* the dynamics are organized in clusters of states occupying different areas of the state space, each corresponding to a different set of actions to perform. These results allowed us to outline a procedure to derive a compact view of the best performing network behaviors in terms of FSMs.

A major advantage of this method over current automatic design of FSM controllers for robots is that it does not require any assumption on the number of states nor conditions on the transitions between states. This implies that the behavior of the robot is automatically segmented, i.e., the actions composing the robot's behavior do not need to be specified *a priori*. We would like to emphasize that the use of BNs makes it possible to exploit the properties of both NNs and high level representations like FSMs. In fact, most NN-based robot programs define a mapping between sensor readings and actions on the actuators and thus operate as low-level, fine grained programs which are particularly effective in reactive systems. Conversely, FSM control software is usually based on high-level actions and it is suitable for modular control programs which can be also formally verified. With BN robot programs we can combine both characteristics, as BNs can indeed operate low-level and, at the same time, enable the designer to manipulate a FSM description of the robot control software.

The work carried out for this paper is only a first necessary step towards the application of the proposed approach to more complex and demanding tasks. Future work will focus on improving the performance of the design process and defining an automatic method for synthesizing FSMs starting from Boolean networks. These steps are required for a fair comparison of the proposed approach with existing and well refined design methods.

Of course, the approach has also some limitations. First of all, it requires to deal with Boolean inputs and outputs, which could be sometimes problematic. In addition, the FSM is derived by collecting samples of BN trajectories and a trade-off between precision and computational complexity has to be found.

Acknowledgements

This work was partially supported by the EU project ASCENS (grant 257414). Mauro Birattari acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate.

References

Ansaloni, L., Villani, M., and Serra, R. (2009). Dynamical critical systems for information processing: a preliminary study. In Villani, M. and Cagnoni, S., editors, *Proceedings of the Satellite Workshops of the International Conference of the Italian Association for Artificial Intelligence (AIIA09)*, pages 210–218. Reggio-Emilia, Italy.

Bar-Yam, Y. (1997). *Dynamics of complex systems. Studies in nonlinearity*. Addison-Wesley, Reading, MA.

Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):471–511.

Dorigo, M. (1994). Learning by probabilistic boolean networks. In Ruck, D., Wada, M., and Bounds, D., editors, *1994 IEEE International Conference on Neural Networks: IEEE World Congress on Computational Intelligence*, pages 887–891. IEEE Press, Piscataway, NJ.

Fogel, D. B. (1993). Evolving behaviors in the iterated prisoner's dilemma. *Evolutionary Computation*, 1(1):77–97.

Fogel, L. J. (1962). Autonomous automata. *Industrial Research Magazine*, 4(2):14–19.

Fretter, C. and Drossel, B. (2008). Response of boolean networks to perturbations. *The European Physical Journal B - Condensed Matter and Complex Systems*, 62(3):365–371.

Garattoni, L., Roli, A., Amaducci, M., Pinciroli, C., and Birattari, M. (2013). Additional material to the paper "Boolean network robotics as an intermediate step in the synthesis of finite state machines for robot control". Available as `http://iridia.ulb.ac.be/supp/IridiaSupp2013-004/`.

Gershenson, C. (2004). Introduction to random boolean networks. In Bedau, M., Husbands, P., Hutton, T., Kumar, S., and Suzuki, H., editors, *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, pages 160–173. MIT Press, Boston, MA.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467.

Kauffman, S. A. (1991). Antichaos and Adaptation. *Scientific American*, 265:78–84.

Koza, J., Keane, M., and Streeter, M. (2003). Genetic programming's human-competitive results. *IEEE Intelligent Systems*, pages 25–31.

Lipson, H. (2005). Evolutionary robotics and open-ended design automation. In Cohen, B., editor, *Biomimetics*, pages 129–155. CRC Press.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In Gon calves, P., Torres, P., and Alves, C., editors, *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB, Castelo Branco, Portugal.

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics*. The MIT Press.

O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363.

Patarnello, S. and Carnevali, P. (1986). Learning networks of neurons with boolean logic. *Europhysics Letters*, 4(4):503–508.

Petrovic, P. (2007). Strengths and weaknesses of FSA representation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 723–725. ACM, New York, NY, USA.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L. M., and Dorigo, M. (2012). ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

Ribeiro, A. S., Kauffman, S. A., Lloyd-Price, J., Samuelsson, B., and Socolar, J. E. S. (2008). Mutual information in random boolean models of regulatory networks. *Physical Review E*, 77(1):011901.

Roli, A., Manfroni, M., Pinciroli, C., and Birattari, M. (2011). On the design of boolean network robots. In *Proceedings of EVOApplications 2011*, Lecture Notes in Computer Science, pages 43–52. Springer, Berlin, Germany.

Serra, R., Villani, M., Graudenzi, A., and Kauffman, S. A. (2007). Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. *Journal of Theoretical Biology*, 246(3):449–460.

Serra, R. and Zanarini, G. (1990). *Complex Systems and Cognitive Processes*. Springer-Verlag, Secaucus, NJ.

Sun, R. and Giles, L. C. (2001). Sequence learning: From recognition and prediction to sequential decision making. *IEEE Intelligent Systems*, 16(4):67–70.

Yamauchi, B. (1993). Dynamical neural networks for mobile robot control. Technical report, NRL Memorandum Report AIC-033-93 (Naval Research Laboratory).

Yamauchi, B. and Beer, R. D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246.