Designing a Growing Functional Modules “Artificial Brain”

Jérôme Leboeuf-Pasquier
University of Guadalajara, Mexico
jerome.lep@gmail.com

Eduardo José González Pacheco Oceguera
University of Guadalajara, Mexico
eduardogpo@hotmail.com

Abstract
The present paper illustrates the design process for the Growing Functional Modules (GFM) learning based controller. GFM controllers are elaborated interconnecting four kinds of components: Global Goals, Acting Modules, Sensations and Sensing Modules. Global Goals trigger intrinsic motivations, Acting and Sensing Modules develop specific functionalities and Sensations provide the controlled system's feedback. GFM controllers learn to satisfy some predefined goals while interacting with the environment and thus should be considered as artificial brains. An example of the design process of a simple controller is provided herein to explain the inherent methodology, to exhibit the components' interconnections and to demonstrate the control process.

Keywords: growing functional modules, learning based control, machine learning, connectionism, embodiment, artificial brain editor, embodied robotics, emergent behaviors.

1. Introduction

In computer science, controllers are designed to provide real time response to events sensed by the system they control. Controllers usually perform in real time and in a real world, such environment is characterized by its complexity, illegibility, dynamics among other confusing aspects. Hence, to face these adverse conditions, programmers drastically reduce the intricacy of the robot's environment; otherwise, the number of lines of code would increase drastically. “Classical” Artificial Intelligence does not offer much help because replacing sequential-imperative programming by another paradigm (functional, logical) still implies a huge coding effort. Biological entities, whose brains may be perceived as controllers, are able to deal with the complexity of the real world thanks to their learning abilities.

The Growing Functional Modules' project, inspired by machine learning, connectionism, and embodiment, is setting out to do the same: allowing the design of learning based controllers. Such controllers result from the interconnection of dynamic acting and sensing modules that do not integrate any previous knowledge but elaborate it gradually while interacting with the environment and hence, should be considered as “artificial brains”.

At this stage, the earliest experiments attest to the well founded of this approach [1]. Nevertheless, different tools have been required to pursue the project's development, among others robots' simulators, an embedded interface to implement robots and presently, a graphic editor to help designing the controllers [2]. As the controller results from the interconnection of different components, the editor is particularly convenient in order to create, update and display the resulting internal structures. A previous version of the editor [3] soon became obsolete because only one of the actual four kinds of components was implemented.

The purpose of the present paper is to explain how GFM controllers perform, illustrating their design, step by step of a simple learning based controller integrating its four components: Global Goals, Acting Modules, Sensations and Sensing Modules. The content has been organized in order to facilitate this achievement: section 2 describes the GFM basic control loop and section 3, the controller's components integrated in the editor. An illustration of an architecture's design is given in section 4 while its execution and graphical displays are described in section 5.

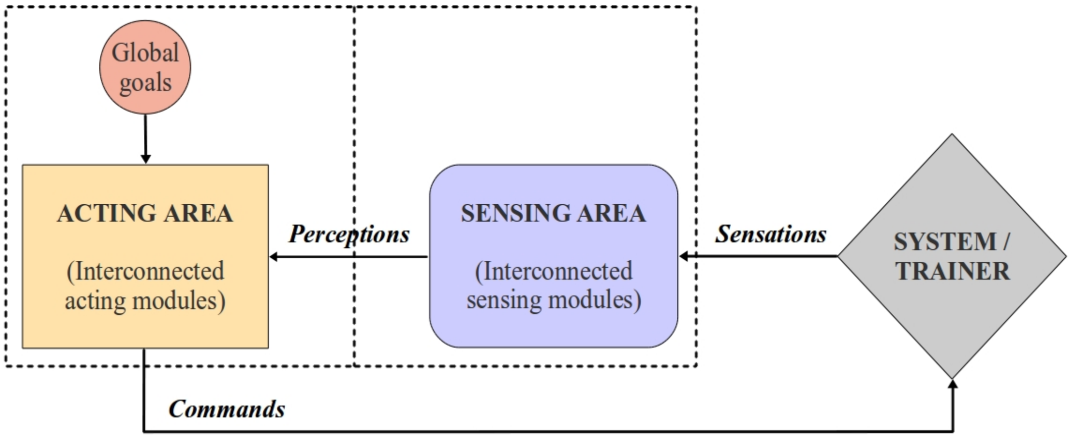


Figure 1. The GFM controller and its control loop

2. The control loop

The GFM control loop has many similarities to a standard one as shown in figure 1. The controller, delimited by a dotted line, sends during each cycle an output command in order to trigger some mechanical or virtual actuators and in return, receives a feedback composed of a sequence of sensors' values. However, the concept of reference value is replaced by “Global Goals” which are integrated to the controller. The Global Goals' concept refers to the set of motivations that induces the controller's activities and consequently, the system's behavior. An illustration of such “motivation” is given in section IV.

Moreover, the controller's architecture is divided in two areas, the “Sensing” and the “Acting” ones. The Sensing Area is in charge of interpreting the feedback sent by the system, the feedback is referred as “Sensations” (figure 1). Sensations are processed in input by a set of interconnected sensing modules that produce “Perceptions” in output. Perceptions are translated to the Acting Area where a set of Acting Modules processes these feedback values. According to these values and the input requests, the Acting Area triggers the next command to the system, initiating thus a new cycle of the control loop.

3. The controller's components

The controllers' components include Sensation, Global Goal, Acting Module and Sensing Module; there are available by clicking on the associated button of the editor's toolbar. The components functionalities and configurations are described in the following subsections.

3.1. Sensations

Each “Sensation” corresponds to an integer value corresponding to a specific system's sensor. Sensations are symbolized by a green rectangle on the editor's canvas (figure 2.a). Each newly created sensation is assigned an identifier previously incremented. Its unique field, initially filled with question-marks, allows it to associate a mnemonic in order to facilitate its interpretation.

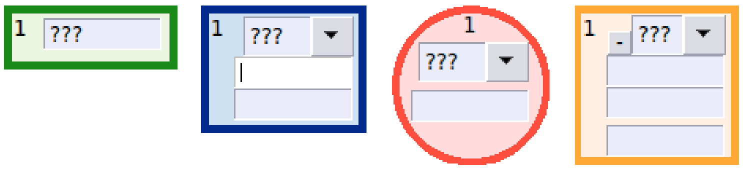


Figure 2. The editor's components from left to right:
a) Sensation, b) Sensing Module, c) Global Goal, d) Acting Module.

The defined sequence of sensations represents the system's feedback that is a sequence of sensors' values. Thus, both sequences must have the same length and their values must be in the same order. Sensations are employed as inputs for Sensing Modules and/or as feedback for Acting Modules.

3.2. Sensing modules

Sensing Modules are symbolized by a blue rectangle on the editor's canvas (see figure 2.b). Each newly created Sensing Module is assigned an identifier previously incremented. The upper field, initially filled with question-marks, indicates the type (or functionality) of the component. For example, this functionality may consist in applying a specific classification on sensations given in input. The component internal process will acquire this functionality meanwhile the system is interacting with its environment.

Connecting input Sensations is performed by right clicking a component in order to create a new connection specifying the output component. The middle field allows the specification of an Acting Module's identifier that confirms the correctness of the Sensing Modules output perception in order to allow learning. The lowest field contains some extra-parameters whose number and meaning are specific to each type of module. The input connections of a Sensing Module come from a subset of Sensations or Sensing Modules' outputs.

3.3. Global goals

Global Goals are symbolized by a red circle on the editor's canvas (figure 2.c). Each newly created Global Goal is assigned an identifier previously incremented. The upper field, initially filled with question-marks, allows the definition of the Global Goal's type. This type determines the request that will be sent to a unique Acting Module connected in output. The type specifies, among others, a constant, minimum, cyclic or user-specified bequest's value.

The lowest field allows the capture of some extra-parameters whose number and meaning are specific to each type of Global Goal. For example, if the type indicates that the request corresponds to the selection of a random value inside an interval then, the Global Goal requires two extra-parameters specifying both limits of this interval.

3.4. Acting modules

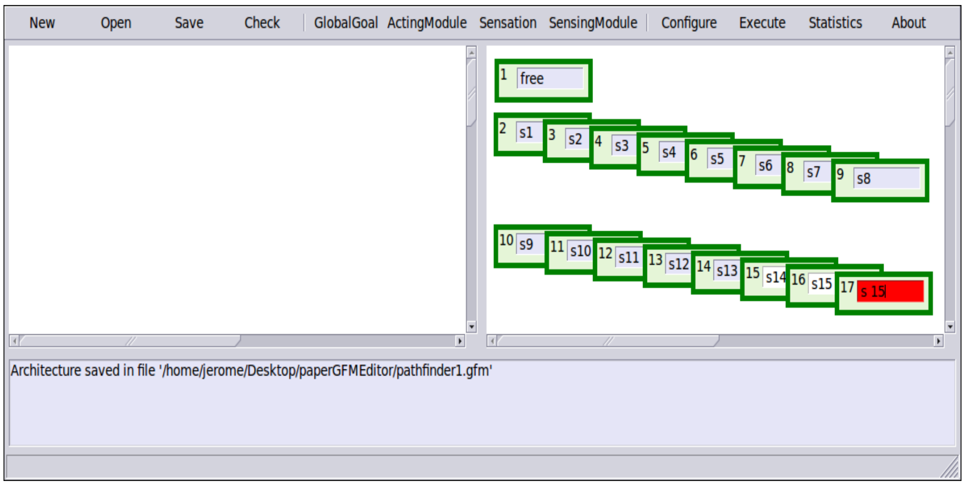
Acting Modules are symbolized by an orange rectangle on the editor's canvas (figure 2.d). Each newly created Acting Module is assigned an identifier, previously incremented. The upper field, initially filled with question-marks, indicates the type (or functionality) of the component. For example, Real Time Regulation in order to learn how to maintain one system's component inside a precise range [4] or Causal Inference in order to trigger states' transitions [5].

A single Global Goal or another Acting Module connected in input specifies at each cycle the request that should be reached. According to its type, the component's internal process is able to acquire a specific functionality while interacting with the environment. At each cycle, the result of this interaction is given by a subset of Sensations and/or Perceptions; this subset is specified in the upper middle field as a list of Sensations and Perceptions identifiers.

The lowest field contains some extra-parameters whose number and meaning are specific to each type of module. Right clicking allows the creation of output connections up to a maximum of four output Acting Modules. If no output connection is specified then, it is considered that the Acting Module sends commands to the system. In that case, the list of commands and their associated amplitudes should be declared in the lower middle field.

4. Designing a controller

The previous section offers a description of the different components and of the way to interconnect them. The present section will illustrate how to design a GFM architecture using the editor and interconnecting the described components.



Architecture saved in file '/home/jerome/Desktop/paperGFMEditor/pathfinder1.gfm'

Figure 3. Describing the sequence of input sensations sent by the system.

The control system's challenge consists of a basic simulation of an autonomous vehicle equipped with sixteen proximity sensors regularly disposed on its front part. Each sensor returns a positive value when an obstacle is present at a short distance in front of this sensor. The simulation generates a random row of obstacles presenting a large enough opening to allow the vehicle to pass through. The vehicle's controller should learn to change the vehicle's position in order to cross through the opening. Figure 10.a illustrates this behavior: the first lines of text shows a success, the vehicle's controller encounters a path free of obstacles at the first try while for the next row of obstacles, five attempts are necessary to succeed.

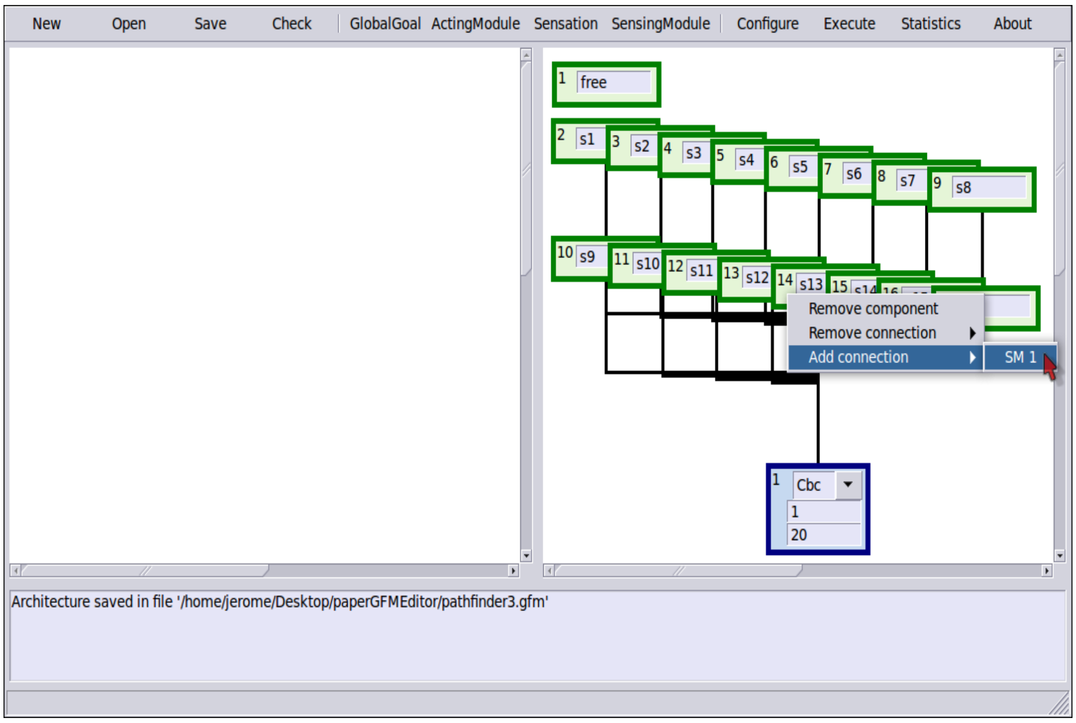
4.1. Design step 1

To design the controller, first of all, the user should click the button “new” on the toolbar to create an empty canvas and to assign it a controller's name, presently “pathfinder”. Then, the first convenient step is to describe the feedback given by the system as a sequence of sensations (see figure 3). The first sensation named “free”, indicates if the vehicle has been able or not to cross the range of obstacles. Each of the next sixteen values is associated to its corresponding proximity sensors that points out the presence of an obstacle. They are given a name beginning with “s” followed by a number. The red color of the last sensation's field indicates that the given name is invalid; presently, because it contains a blank character.

4.2. Design step 2

The next step consists in adding a Sensing Module able to classify the sixteen obstacles to produce a single output perception. In this case, the module's type is CBC meaning “Combination Based Classifier”. According to this type, some perceptions corresponding to obstacles' configurations are later discarded; others are gradually adapted to generate a successful pattern.

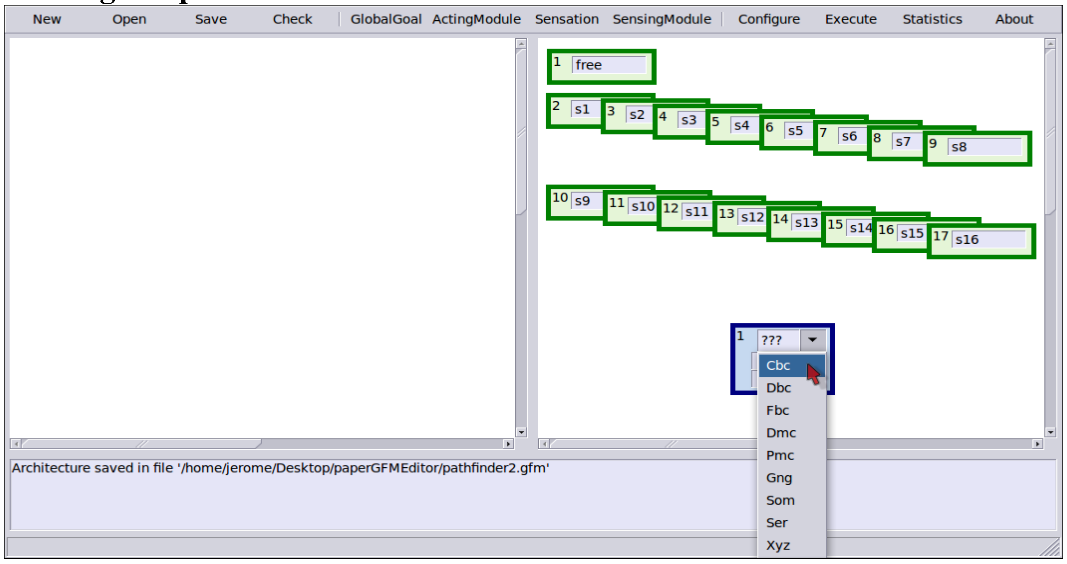
The next step consists of connecting the sixteen sensations in the input of the Sensing Module. To do this, the user must right-click on each Sensation, then on “new connection” and indicate the Sensing Module identifier. The resulting design is presented on figure 5. Finally, the Acting Module's field is set to “1” (indicating the number of the Acting Module that later will assess the correctness of the perception) and the unique extra-parameter set to “20” (related with the module's behavior).



Architecture saved in file '/home/jerome/Desktop/paperGFMEditor/pathfinder3.gfm'

Figure 4. Adding a Sensing Module and selecting its type

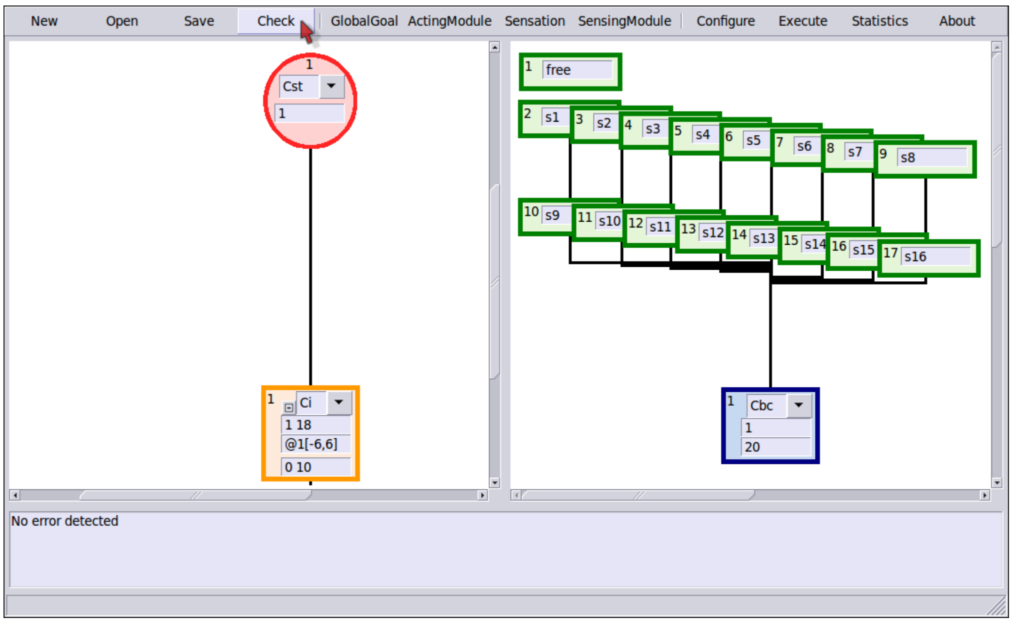
4.3. Design step 3



Architecture saved in file '/home/jerome/Desktop/paperGFMEditor/pathfinder2.gfm'

Figure 5. Adding connections from Sensations 2-17 to the Sensing Module 1

The next step consists of adding a Global Goal expressing a motivation required by the controller. The goal is to keep the vehicle's front free of obstacles, thus the Sensation “free” should stay equal to “1”. After adding a new Global Goal, its assigned type should be “Cst” corresponding to a constant output request (see figure 6). In the parameter field, its specified value is “1”.

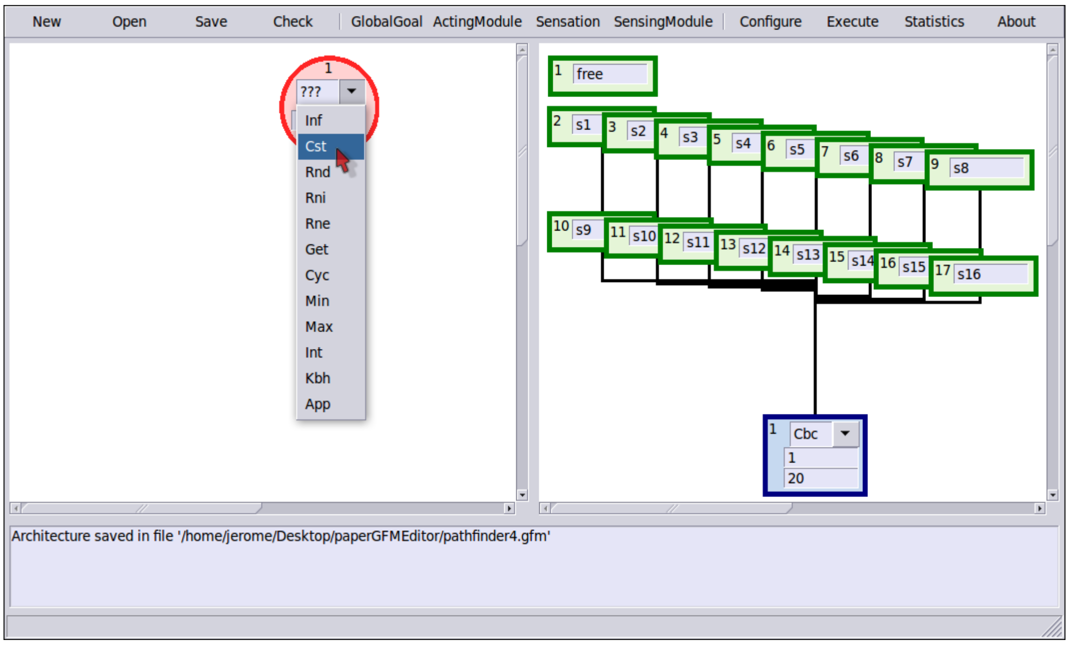


No error detected

Figure 6. Adding a Global Goal with its required type

4.4. Design step 4

The fourth step consists of adding an Acting Module, its configuration values and input connection as shown in figure 7. A type “CI” is assigned because it functionality will consist of triggering a steering command in accordance with the perception from the Sensing Module and in order to satisfy the input request from the Global Goal. Consequently, the feedback is set to “1 18” where “1” is the reference to the Sensation “free” and “18” to the perception in output of the Sensing Module. The identifier “18” for this perception is computed as at the total number of Sensation plus one (first sensing module). Identifiers and their references are automatically updated when a Sensation is added or deleted.



Architecture saved in file '/home/jerome/Desktop/paperGFMEditor/pathfinder4.gfm'

Figure 7. Adding an Acting Module, its configuration values
and its input connections

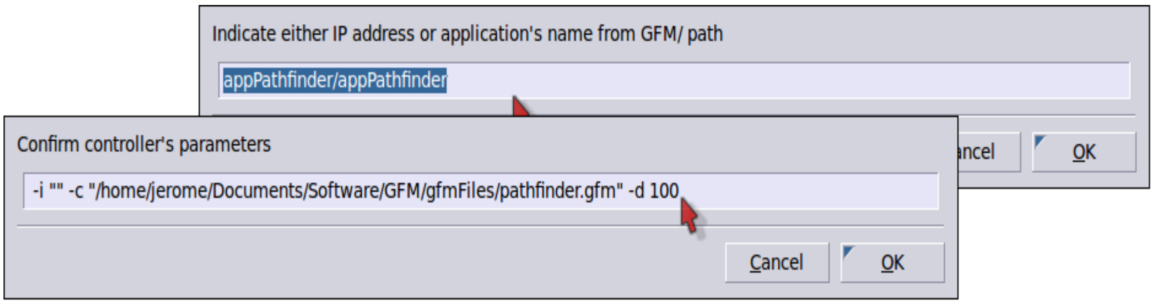
The next field of the Acting Module, corresponding to the output command to the system is assigned the string "@1[-6,6]" indicating that there is a single output command "1" whose amplitude may vary from -6 to 6. Next, the two required parameters according to the type "CI" are set to "0" and "10".

Finally, left clicking on the Global Goal allows to add a new connection from this Global Goal to the input of the Acting Module.

5. Running the controller

5.1 The "configure" command

Before running the controller, it is necessary to specify the communication's configuration. When pressing the "configure" button, two text windows appear (figure 8). The upper one allows to specify the application's localization in order to run it previously to the controller. Presently, the corresponding simulation will run on the same machine, thus the name of the corresponding executable file must be specified. Otherwise, in the same field, the user should specify the IP address of the server where the controlled system is running.


Indicate either IP address or application's name from GFM/ path

appPathfinder/appPathfinder

Confirm controller's parameters

-i "" -c "/home/jerome/Documents/Software/GFM/gfmFiles/pathfinder.gfm" -d 100

Figure 8. Configuration of the controller

The second and lower window specifies the controller's parameters including the descriptor file, the duration of a period, an eventual seed for the random number generator and so on...

5.2. The "save" command

At this step, the graphic design and configuration of the controller have been completed. Clicking the "save" button will generate a description of the controller that is written in a file identified by the controller's name and the extension "gfm". This file will be loaded by the controller when starting to set the components and their interconnections. The content of the file that corresponds to the controller "pathfinder.gfm" is given in figure 9.

```
#application's name
pathfinder
#number of errors
0
#errors

#execution configuration
 -i "" -c
"/home/jerome/Documents/Software/GFM/gfmF
iles/pathfinder.gfm" -d 100
appPathfinder/appPathfinder
#number of global goals
1
#number of sensing modules
1
#number of acting modules
1
#number of sensations
17
```

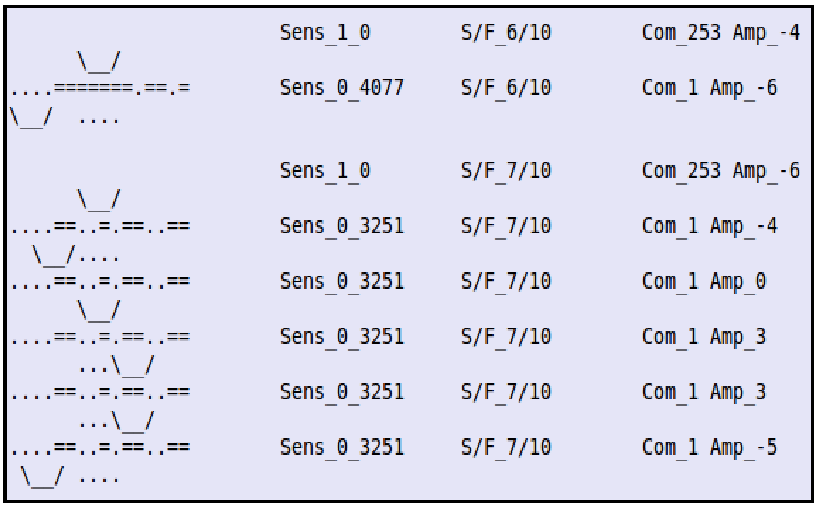
```
#number of perceptions
1
#sensations
free o1 o2 o3 o4 o5 o6 o7 o8 o9 o10 o11
o12 o13 o14 o15 o16
#x coordinate
4 3 112 221 331 441 551 34 144 254 363
473 80 190 300 409 519
#y coordinate
13 63 63 63 63 63 63 115 115 115 115 115
174 174 174 174 174

#sensing module type
Cbc
#sensing module ident
1
#input values position
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 -
1
```

```
#output values position
18 -1
#prediction position
1 -1
#extra parameters
20
#x coordinate
304
#y coordinate
357
#acting module type
Ci
#acting module ident
1
#amplitudes of commands
@1[-6,6]
#input acting module's ident
-1
#input global goal's ident
1
#output acting module's ident
-1
#feedback position
1 18 -1
#extra parameters
0 10
#x coordinate
355
#y coordinate
284

#global goal type
Cst
#global goal ident
1
#output acting module's ident
1
#extra parameters
1
#x coordinate
357
#y coordinate
24
```

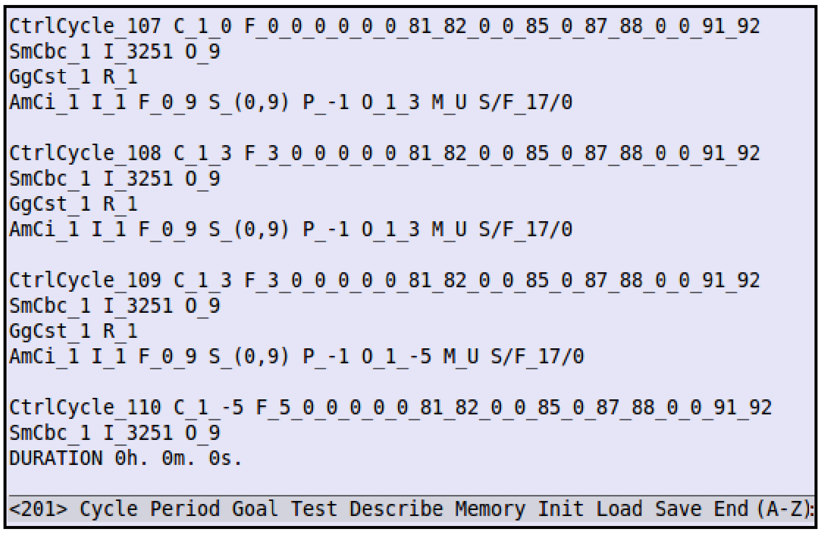
Figure 9. Content of the file corresponding to the "pathfinder" controller



```
                        Sens_1_0        S/F_6/10        Com_253 Amp_-4
      \__/
....=======.==.=        Sens_0_4077     S/F_6/10        Com_1 Amp_-6
\__/   ....

                        Sens_1_0        S/F_7/10        Com_253 Amp_-6
      \__/
....==..=.==..==        Sens_0_3251     S/F_7/10        Com_1 Amp_-4
  \__/....
....==..=.==..==        Sens_0_3251     S/F_7/10        Com_1 Amp_0
      \__/
....==..=.==..==        Sens_0_3251     S/F_7/10        Com_1 Amp_3
      ...\__/
....==..=.==..==        Sens_0_3251     S/F_7/10        Com_1 Amp_3
      ...\__/
....==..=.==..==        Sens_0_3251     S/F_7/10        Com_1 Amp_-5
 \__/ ....
```

Figure 10.a. Simulation's display: The characters "\__/" represents the front part of the car that should pass through the string "....==..=.==..==" representing the range of obstacles



```
CtrlCycle_107 C_1_0 F_0_0_0_0_0_0_81_82_0_0_85_0_87_88_0_0_91_92
SmCbc_1 I_3251 O_9
GgCst_1 R_1
AmCi_1 I_1 F_0_9 S_(0,9) P_-1 O_1_3 M_U S/F_17/0

CtrlCycle_108 C_1_3 F_3_0_0_0_0_0_81_82_0_0_85_0_87_88_0_0_91_92
SmCbc_1 I_3251 O_9
GgCst_1 R_1
AmCi_1 I_1 F_0_9 S_(0,9) P_-1 O_1_3 M_U S/F_17/0

CtrlCycle_109 C_1_3 F_3_0_0_0_0_0_81_82_0_0_85_0_87_88_0_0_91_92
SmCbc_1 I_3251 O_9
GgCst_1 R_1
AmCi_1 I_1 F_0_9 S_(0,9) P_-1 O_1_-5 M_U S/F_17/0

CtrlCycle_110 C_1_-5 F_5_0_0_0_0_0_81_82_0_0_85_0_87_88_0_0_91_92
SmCbc_1 I_3251 O_9
DURATION 0h. 0m. 0s.

<201> Cycle Period Goal Test Describe Memory Init Load Save End (A-Z):
```

Figure 10b. The control loop's information displayed by the controller

5.3. The "execute" command

Once performed the design and configuration, the controller and the local application are executed by pressing the corresponding button. In case of a local simulation, two terminal windows are generated. The first one, localized on the left side of the screen (figure 10.a), displays the behavior of the simulation; the second one, localized on the right side of the screen (figure 10.b), displays the behavior of the controller. Both application run concurrently and their respective contents allow the user to observe and monitor the control session. The last text line of figure 10.b displays the set of commands at cycle 201.

6. Displaying statistics

Clicking the button "statistics" opens three graphics displays of different log files describing the behavior of a previous control session. Each of these files contains a list of different kinds of events that have been occurring during the cycles of the control loop.

The first file, identified as <*controllerName*>*Growing.sts*, is a record of the sizes of acting and sensing modules throughout the control session. Instead of bytes, the size represents the number of basic components created to store a new information. Hence, rather than memory size, the value expresses the number of allocated objects. This is an always relevant information concerning the controller's behavior, in particular to detect modules' excessive growing. The corresponding graphic is displayed in figure 11.a.

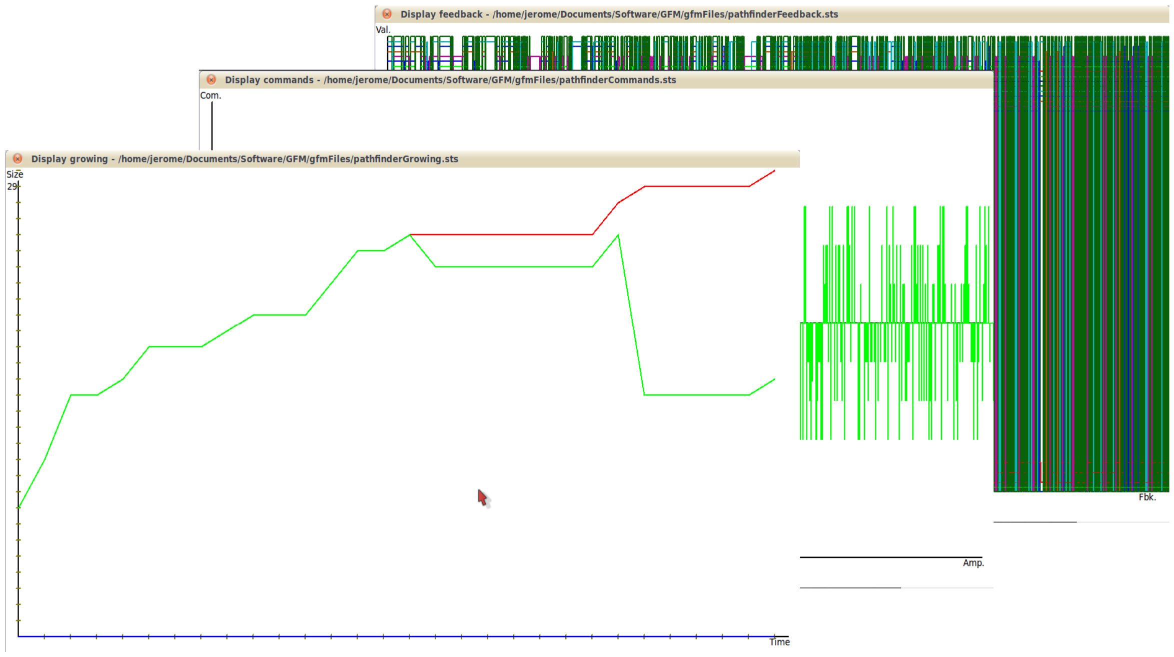

Display feedback - /home/jerome/Documents/Software/GFM/gfmFiles/pathfinderFeedback.sts
Val.
Fbk.
Display commands - /home/jerome/Documents/Software/GFM/gfmFiles/pathfinderCommands.sts
Com.
Amp.
Display growing - /home/jerome/Documents/Software/GFM/gfmFiles/pathfinderGrowing.sts
Size
29
Time

Figure 11. Graphics displays of the controller's behavior (from left to right)
a) Growing of the modules, b) Commands, c) Feedback

The second file, named <*controllerName*>*Commands.sts*, is a record of the command-amplitude values send by the controller at the beginning of each control loop. This could lead to meaningful interpretation, for instance, a diminution over time of the amplitudes reflects that learning is favoring the system's performance: reaching a similar behavior with less commands. The corresponding graphic is displayed in figure 11.b.

The third file, named <*controllerName*>*Feedback.sts*, is a record of the feedback values send back to the controller at the end of each control loop. Displaying these values exhibits the actuators' and sensors' behaviors, reflecting thus the controller's fitness over time. In particular, these data to evaluate the precision the resulting control. The corresponding graphic is displayed in figure 11.c. These three files must be previously created during the control's session; in practice, they may be opened or closed at each cycle of the control loop.

Nevertheless, the main result concerning the learning ability of the controller, the evolution of the accumulated error through time, is not displayed as part of these statistics. The reason is that this task must be performed by the application/trainer which always has more accurate criterion to evaluate a successful behavior. For example, considering the present application, to be considered successful, a path through a range of obstacles must be found at the first attempt. The resulting curve of accumulated errors is presented figure 12. As shown, a rather trustful controller is obtained after 800 challenges (approximately 2000 cycles); nevertheless, sporadic errors still occur due to side effects of permanent learning (learning is not limited to a previous phase).

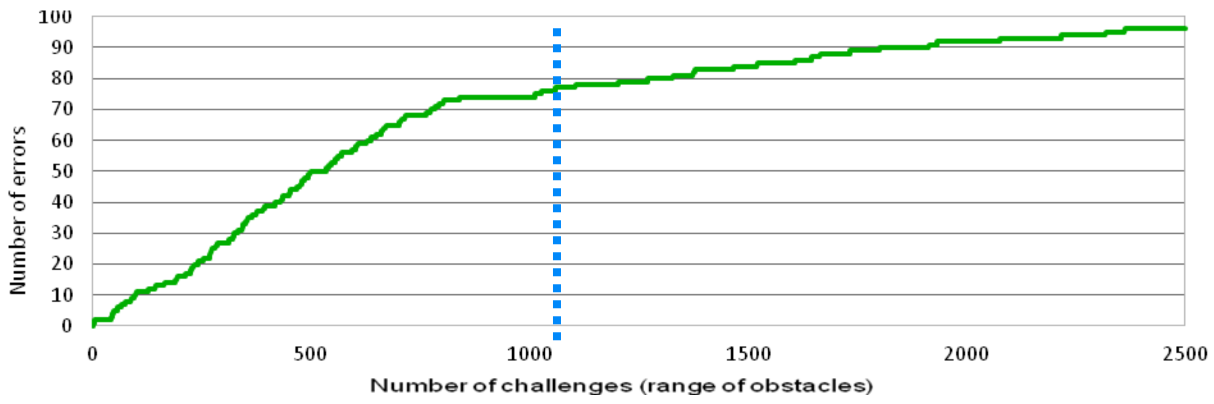


Figure 12. Evolution of errors' number through challenges' number

7. Conclusions

A minimalist GFM controller should integrate at least three components: one Sensation, one Acting Module and one Global Goal. The controller described herein, includes seventeen Sensations, one Sensing Module, one Acting Module and one Global Goal, in order to give a complete overview of the GFM proposal. More complex controllers have already been designed. Among others, the editor has been employed to ease the creation, actualization and visualization of two recent masters' thesis projects: an auditory subsystem conceived to learn in context [6] and an equilibrium subsystem implemented in a humanoid robot [7]. For both projects, the editor has become a very helpful tool. In fact, both applications have contributed to improve the functionalities integrated in the editor but also their corresponding counterpart. In particular, the format of the gfm files that contain the description of the components' interconnections has been updated to be compatible with the editor and the controller.

The design process described herein, clearly shows the role of the different components and of their interactions specified by their interconnections. Moreover, the behavior of the controller may be analyzed by displaying the record files of the control loop. Further developments of the editor will consider improving its visualization functionalities. In particular, module activations or module growing could be displayed in real time.

Finally, the main goal of this research project has been to develop artificial brains that should allow a controlled system to develop an emergent behavior while interacting with its environment in order to satisfy some predefined global goals. The controller described in the present paper offers an illustration of such behavior. Therefore, GFM is a paradigm that would potentially allow the programming phase of a control system to be replaced with the visual design of a learning based controller.

References

[1] Leboeuf-Pasquier, J., Lecture Notes in Artificial Intelligence 3789, Springer (2005), 959-969 *Applying the GFM Prospective Paradigm to the Autonomous and Adaptive Control of a Virtual Robot.*

[2] Leboeuf, J., González, E., Herrera, K., Galván, R., Gómez, G., Robles, J., Rodríguez L., Nava, D., Bautista, J., Cambrano, G., González, R., Avances del Cuerpo Académico "Ingeniería de Manufactura" (2011), *Desarrollo del Paradigma Growing Functional Modules: Avances 2010*

[3] Gaytán, N., Master Thesis in Information Technology, CUCEA, University of Guadalajara (2007), *Una Herramienta Gráfica para la Generación Automática de Controladores GFM.*

[4] Leboeuf, J., Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society, IECON'06 (2006) 3945-3950, *Improving the RTR Growing Functional Module.*

[5] Leboeuf, J., Proceedings of the 4th Intl Conference on Informatics in Control, Automation & Robotics ICINCO (2007), 456-463, *A Growing Functional Module Designed to Trigger Causal Inference*, 456-463.

[6] Gómez-Avila, G., Master Thesis in Information Technology, CUCEA, University of Guadalajara (2011), *Development of an auditive subsystem based on the GFM paradigm.*
[7] Galvan, R., Master Thesis in Information Technology, CUCEA, University of Guadalajara (2012), *Desarrollo del subsistema de equilibrio de un robot humanoide basado en el paradigma GFM.*

Acknowledgement
This paper is based upon work supported by the Jalisco State Council of Science and Technology of Mexico (project FOMIXJAL-2008-05-98486).