Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 18 Issue 2 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Robotic Behavior based on Formal Grammars

By J. L. Briseño, M.A. Jiménez & G. Olague

Abstract- Formal grammars, studied by N. Chomsky for the definition of equivalence with languages and models of computing, have been a useful tool in the development of compilers, programming languages, natural language processing, automata theory, etc. The words or symbols of these formal languages can denote deduced actions that correspond to specific behaviors of a robotic entity or agent that interacts with an environment. The primary objective of this paper pretend to represent and generate simple behaviors of artificial agents. Reinforcement learning techniques, grammars, and languages, as defined based on the model of the proposed system were applied to the typical case of the ideal route on the problem of *artificial ant*. The application of such techniques proofs the viability of building robots that might learn through interaction with the environment.

Keywords: *context-free grammars, robotic behavior, intelligent agent, machine learning.*

GJCST-D Classification: *I.2.9*

Strictly as per the compliance and regulations of:

Robotic Behavior based on Formal Grammars

J.L. Briseño [α], M.A. Jiménez [σ] & G. Olague [ρ]

Abstract- Formal grammars, studied by N. Chomsky for the definition of equivalence with languages and models of computing, have been a useful tool in the development of compilers, programming languages, natural language processing, automata theory, etc. The words or symbols of these formal languages can denote deduced actions that correspond to specific behaviors of a robotic entity or agent that interacts with an environment. The primary objective of this paper pretend to represent and generate simple behaviors of artificial agents. Reinforcement learning techniques, grammars, and languages, as defined based on the model of the proposed system were applied to the typical case of the ideal route on the problem of *artificial ant*. The application of such techniques proofs the viability of building robots that might learn through interaction with the environment.

Keywords: context-free grammars, robotic behavior, intelligent agent, machine learning.

I. Introduction

Advances in artificial intelligence (AI) have supported to the development of sciences and disciplines whose influence leads to experimentation and application of new techniques that seek an optimal solution or high performance, depending on the problem that has arisen. The problem of emulating human navigation skills in an environment involves the analysis of behaviors of the humans, which are mapped to cases of robots or humanoids to develop in them mechanisms needed to help find solutions to specific problems. Such mechanisms form the structure of models that encompass interactions between the modules involved in the solution of such a problem. In this paper, the proposed model follows the basic structure defined by Russell and Norvig on learning agents [1]; then considering modules of events and actions that interact with the environment through sensors and actuators. The functionality of such modules lies in the implementation of context-free grammars as a tool for describing and build robotic behaviors. The purpose of this work is to manage behaviors of agents acting within an environment, based on grammars and formal languages. Some work in this direction will be mentioned briefly in the next section. In section 3 the fundamental principles of regular grammars will be addressed; as they are used to describe knowledge about events and actions, as well as determine the language of valid words that define behavior. Section 4 describes the fundamental concepts of grammars, terminology, and reinforcement learning (RL) used in finding the solution on the Santa Fe Trail problem [2]. The last two sections 5 and 6 show the results and conclusions respectively.

Author α: e-mail: briseno@cicese.mx

II. Related Work

In the recent years, few approaches based on formal grammars have been used to learn behaviors in robot navigation problems. Some of these approaches outline the use of bio-inspired mechanisms to build systems with human-like behavior based on spiking neural networks (SNN) [3]; Others that use the RL, Bayesian techniques, and decision trees, rely on human experience and pretend that a robot agent learns to make decisions in generalized models [4], [5]. Otherwise, the ART system [6] considers learning behaviors as a classification problem, and uses a neural network to increase the learning capacity without losing earlier information of other behaviors when they are learning new ones; also integrated audio, and vision models in the current perception of the behavior are observed. Within the first work on behavior and autonomy of robots, in [7] these issues are described based on the representation in nets and using genetic algorithms (GA) as a method to search movements which are learned and generated from the states involved in a net. Regarding the generation of behaviors in [8] are proposed architectures that emulate cortical regions of the human brain which relate modules perception declarative memory, and objectives management. Such modules supported in production systems respond to patterns of information stored on the work memory engaged. In [9], behaviors are in software architecture designed to handle reusable components where each of them has a specific role based primarily on finite state machines (FSM). On the other hand, below the context of using of formal languages for generating robotic behaviors, Manikonda, et al., [10] propose techniques and models with differential equations based on kinetic state machines. In more recent work, Dantam and Stillman [11] formally develop the concept Grammar Movements doing a complete linguistic analysis for robotic control, being very similar some of its specific goals with the particular implementation presented in this article.

III. Fundamentals of Languages, Grammars and System Model

The languages described by formal grammars are of great practical importance in the definitions of programming languages, formalization of the rules of the grammars based on the parsing, simplified

translations of languages, and applications for pre-processing chains (words). Each language determined by an abstract computing model: finite automaton, pushdown automata, linear bounded automata, and Turing machines, prescribes regular, context-free (CFLs), context-sensitive, and recursively enumerable languages, respectively.

a) Context-Free Grammars

A *context-free grammar (CFG)* is a finite set of variables (non-terminals or syntactic categories) where each represents a language. These languages represented by the variables are described recursively each other and regard also primary symbols called terminals. Rules that relate these variables are called productions. Formally a *CFG* is denoted as $G = (V, T, P, S)$, where V and T are finite sets of *variables* and *terminals* respectively. Each production in P is of the form $A \rightarrow \alpha$ where A is a variable which can be replaced by a chain (α) of symbols regardless of the context, α must be not empty, with symbols of the set $(V \cup T)^*$[1], S is a special variable called *start* symbol. Some common conventions for grammars are *a)* the capital letters *A, B, C, D, E,* and *S* denote variables; *S* is a start symbol unless it stated in another way. *b)* The lowercase letters *a, b, c, d, e, digits*, and *expressions* in "bold" are terminals. *c)* Capital letters *X, Y,* and *Z* denote symbols that can be variables or terminals. d) The lowercase letters *u, v, w,* and *z* denote terminals. e) The lowercase Greek letters α, β, and γ denote strings of variables and terminals. For further explanation on these issues see Hopcroft and Ullman [12].

In this paper, we will define in the next section, *Robotic Behavior Grammar* (*RBG*) according to the formal description of the *CFG*.

b) System Model

Respect to the system model, this paper follows the general concept of a learning agent according to [1] so that the functional mapping defines the relevant modules regarding specific applications addressed by the *RBG*. The model consists of four main components (Figure 1) linking an agent and an environment. The events module tells the element learning how things are happening about a standard fixed performance, in this case, the target's progress of the agent.

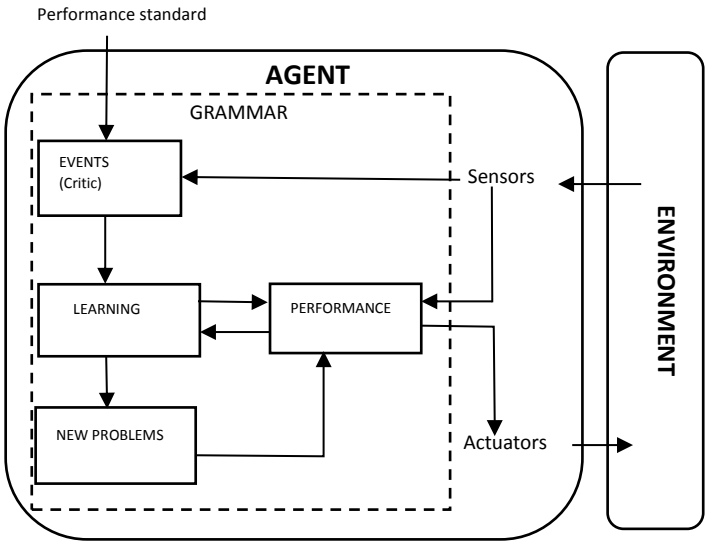

Figure 1: The general model of Robotic Behavior Grammar (*RBG*).

In the *RBG* model, the *sensors* detect occurrences of *events* in the *environment* which may be a valid event, a non-event, and a repeated event. If a valid event occurs in this *environment*, the *events* module recognizes it as a variable *V* corresponding to a fact derivable *Ed* then defining it as *VEd* and verifying that belongs to the set of productions to the derivation of events *PEd*. The learning element will process that information to decide whether the "word" (*event or fact*) is valid in the language generated by the grammar used. The performance module then looks for the production *P*, which relates an action *a*, and a derivate event *e* in the set of productions of actions to events *PEa*. Thus, an external operation carried out by the actor in the environment is selected. As the event is null when it was not generated or fired any event; or repeated, when the agent is in the same state as a previous one, the expansion of variables continues without affecting the current derivation. The process of the agent's behavior ends when its achieved the goal, or when that means reaches a predetermined time limit. For a more versatile agent, the problems generator module takes the current knowledge and can formulate new rules to new situations or to find particular areas of behavior that are susceptible to improvement.

[1] The asterisk right here indicates the possible set of relationships between variables and terminals involved.

IV. Terminology and Concepts

a) Grammars

A grammar of robotic behavior *RBG = (V, Σ, P, S)* define the components as follows: *V* is the set of variables *V = {VAc, VEa, VEd, Vaux}* where *VAc* are variables representing complex actions, *VEa* are variables representing events or states, *VEd* is called a derivation variable of facts, and *Vaux* are auxiliary variables. *S* is the symbol of the start function, which belongs to the subset *VEa*. *Σ* contains the terminal symbols for the primary actions of the agent. The symbol *P* are the productions divided into several subgroups directly related to subsets of *V*. These productions are: *PAc*, which are productions of complex actions; *PEa*, which are productions to select actions for events; *PEd* indicating outputs to derivation of events; *Paux* indicating auxiliary productions for operations; *PSEenv* indicating productions of extension, and *PSEnuln* productions of suppression.

b) Generating behaviors

The RBG can generate behaviors using grammars derivation process, which depends on the different subsets of variables and productions. Those behaviors lead to executing routines that rely on the expansion of production rules. For example, a variable that can be expanded in many different ways, unless there are certain conditions, will always be extended by the same production rule. Whenever is expanded a variable, and a new terminal symbol appears, a corresponding primary action for such a symbol will run. Derivation default will be leftmost derivation, which guarantees a particular order in the process that generates behaviors.

c) Learning

Reinforcement learning (RL) is the method considered in this work. With this method, the robotic agent learns to select an action from several of them possible involved in a particular state. With reinforcement learning the agent seeks to learn an optimal policy π, which for some state *s*, recommend an action *a* that helps the agent to meet its goal. For this process is performed a mapping between the elements of the grammar of behaviors and RL. The reinforcement learning process was carried out using standard algorithms, once are identified states, actions and a control policy for a grammar of behaviors (in this case the RBG). Note that transitions, utilities, and the learning, tend to vary depending on the problem. In an RBG = {V, Σ, P, S}, the productions of the subset *PEa* (member of the set *P*), have the form *Ex → Ai* where *Ex* is a variable that represents an event, and belongs to the subset *VEa*, *Ai* is a variable (where *i=1,2,…n)* representing an action which belongs to the subgroup *VAc*. The purpose of the agent is to learn a policy π: S→A, such that it dictates the next action *at* to be executed when the agent is in a state *st*, ie, $\pi(st)=at$.

Using that policy to generate the most significant utility it's found one way to specify which strategy to use. An arbitrary policy π from a random initial state *st* defined the cumulative value $V\ \pi$(st) according to the equation:

$$V(st) \equiv \Sigma \gamma i \infty i = 0 rt + i \quad (1)$$

where $rt+i$ indicates the sequence of generated rewards, which starts at a state *st* as a result of using a policy π for selecting actions. The formula (1) indicates that starting in state *st* and then implementing the actions recommended by the policy π, the sequence of rewards is obtained [13]. The parameter γ (*gamma*) defined by an assignment of values between 0 and 1, is known as the discount factor, and it indicates the preference between immediate rewards on future rewards. The smaller the value of γ, future rewards will be less significant for the agent. In general, it's intended that the agent learns a policy that maximizes $V\pi(s)$ for all states *s*. This procedure is called optimal policy, given by

$$\pi * \equiv argmax V\pi(s), (\forall s) \quad (2)$$

V. Development and Implementation

a) Grammar for the passage of Santa Fe

It will describe the design and implementation of the RBG based on the context of the problem of *artificial ant* in the particular instance Santa Fe Trail [2]. The aim of the *ant* is eating all the food (cells in black), running all the way, in the shortest possible time; the stage in which the artificial *ant* acts is a 32x32 grid of squares (cells), as shown in Figure 2. Its considered the ant as being the agent that starts in the box at the left upper corner (0,0) facing right and has the following limitations: partially observable environment (only can see the place in which it is) and will have three basic actions:

- Move a unit forward.
- Rotate 90 degrees clockwise.
- Rotate 90 degrees counterclockwise.

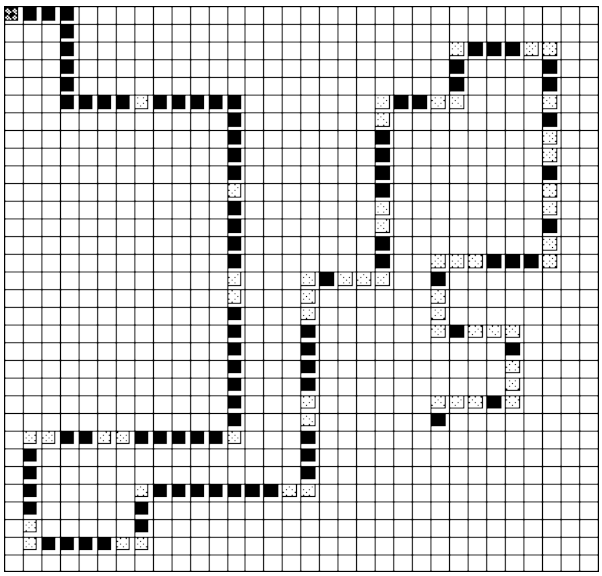

Figure 2: The artifical ant on the Santa Fe Trail problem

The path is irregular and consists of 89 pieces of food, which can be placed consecutively or separated by jumps. Jumps can vary but follow specific patterns, which are: jumps in a straight line, single and double; jumps corner, individual, double (how a horse moves in chess) and triples. The *artificial ant* (Agent) will do the corresponding derivations of the *RBG* after the completion of the algorithm of Figure 3. In this algorithm, the implied grammar will define the events that the agent will receive. Are defined the actions as production rules, and they are explained by a control policy which will select the step to take when an event happens.

The Agent can do three necessary actions, which in the grammar through the terminal symbols *(f), (r), (l),* could represent movement toward the front, turn right 90º, and turn left 90º, respectively. When through the derivation process a new terminal symbol has been adding to the string, the necessary action that corresponds with that terminal symbol will run. From basic movements, are formed complex actions which belong to the subset of productions *PAc*. When is generated an event, will be selected any of these productions (rules). In the case of agent *RBG = (V, Σ, P, S)*, its components are defined as follows:

V={*VAc, VEa, VEd*} P={*PAc, PEa, PEd, PSEev1*}

S={*[START]*} Σ= {*(r), (l), (f)*}

VA_c ={*[ROT RIGHT], [ROT LEFT], [FORWARD]*}

VE_a ={*[START], [ROT 360], [WALL INFRONT], [FOOD], [NO FOOD]*}

VE_d ={*[EVENT]*}

PA_c = {*[ROT RIGHT]* → *(r)[EVENT][ROT RIGHT], [ROT LEFT]* → *(l)[EVENT][ROT LEFT], [FORWARD]* → *(f)[EVENT][FORWARD]*}

PE_d = {*[EVENT]* → *[START]* | *[ROT 360]* | *[WALL INFRONT]* | *[FOOD]* | *[NO FOOD]*}

PE_a = {*[START]* → *[ROT RIGHT]* | *[ROT LEFT]* | *[FORWARD], [ROT 360]* → *[ROT RIGHT]* | *[ROT LEFT]* | *[FORWARD], [WALL INFRONT]* → *[ROT RIGHT]* |*[ROT LEFT], [FOOD]* → *[ROT RIGHT]* | *[ROT LEFT]* | *[FORWARD], [NO FOOD]*→*[ROT RIGHT]* | *[ROT LEFT]* | *[FORWARD]*}

PSE_{ev1} = {*[ROT RIGHT]* → *ϵ, [ROT LEFT]* → *ϵ, [FORWARD]* → *ϵ*}

Note that will use square brackets and parentheses to frame a variable or a terminal, respectively.

The RBG algorithm

1. Start
2. Selecting actions
3. Execution of complex action
4. Running basic action
5. Check for event generated
6. Deletion and / or extension of variables
7. Derivation of events. The event is valid?

Yes.↑ 2
No. ↓ 8
8. Derivation leftmost. Is it produce an action?

Yes. ↑ 4
No. ↓ 8

Figure 3: The general flow of grammar robotic behavior (RBG)

b) Transitions, frequencies and utilities in the Reinforcement Learning (RL)

It will treat a nondeterministic transition function in the implementation of reinforcement learning. That is, when is required to do something to a state, the result will not always be the same, i.e., it hold a probabilistic transition *Pr = (s, a, s ')*, which means the chance of going to the state *s'* by acting *a* from the state *s.* This function is unknown to the agent and, to approximate such function, it will use a table of frequency (*TF*) defined as:

$$TF\,[s, a, s\,'] = TF\,[s, a, s'] + 1$$

That is, for each time it's tried the pass from state *s* to *s'* using the action *a*, it will be adding 1 to the entry table that corresponds to such movement. Just as transitions between states occur, the utility function it presents in a non-deterministic way. Then, it's used a table of utilities (*TU*) to get updated as does the table of frequencies but, the difference is that it's updated *TU* only if the agent finds food in the new state s'. That is,

$$TU[st, at, s't+1] \leftarrow TU[st, at, s't+1] + Ct+1$$

iff s't+1 contains food, otherwise

$$TU[st, at, s't+1] \leftarrow TU[st, at, s't+1] - 0.5$$

Where *Ct+1* is the number of boxes with food that the agent found at time t + 1.

i. *Q-Learning Function*

Since a strategy (policy) is required to show the best action for each state, a function is then used to recommend the optimal movement for that state when using frequency tables and utilities. It needs to update a Q-table using,

$$Q[s, a] \leftarrow updateQValues(s, a), \forall(\text{state-action pair } (s, a))$$

Where, *updateQValues (s, a)* is a call to a function in the algorithm of Figure 4. If it's updated the table Q, it is possible to get simply the optimal policy, which, for a state *s*, is given by the following expression:

$$\pi* (s) \leftarrow \max a[s,a],(\forall(a),\forall(s))$$

During the execution of the algorithm, where defined values of 10 for the size of an epoch (times that it's repeated the journey), and values of 400 for the time limit permissible (necessary actions allowed) before concluding such a period.

The resulting policy guides the agent to find the 89 pieces of food on the Santa Fe path, and this policy it's represented by a string of integers, where each number represents the index of the optimal action for any state.

Algorithm *updateQValues(s,a)*

FUNCTION *updateQValues(s, a)*
INPUT: *state-action pair(s, a)*
var *n* ←0
var *total* ← (times repeated of the pair*(s, a)*
var *count* ← *0*
Repeat for all events *s'*
  var *c* ← *TF(s, a, s')*
  if *c = 0?* **then**
    n ← *n + (UT[s, a, s'] * (c/total))*
    count ← *count + 1*
if *count > 0?* **then**
  OUTPUT: *(n/count)*
if *count <= 0?* **then**
  OUTPUT : 0

Figure 4: Auxiliary function for the updating of the table Q(s, a)

For example, for a numerical policy *3 3 1 3 1* wherein, the mapping corresponding with each production rule that belonging to subset *PE* takes the form:

$$[E] \rightarrow [A1] \mid [2] \mid [A3] \mid \ldots [Ax]$$

and knowing that the policy is $\pi*([E]) = n$ where *n* indicates the *n-th* optimal action to follow for such an event [*E*], then the resultant learned control policy (*3 3 1 3 1*) for the agent can be rewritten, as follows:

$\pi*([START]) = 3$

$\pi*([ROT\ 360]) = 3$

$\pi*([WALL\ INFRONT]) = 1$

$\pi*([FOOD]) = 3$

$\pi*([NO\ FOOD]) = 1$

VI. Results and Conclusions

The software implemented to meet results to the issues raised in this paper can be found in *ftp://ftp.cicese.mx/pub/divFA/ciencomp/briseno/* where its possible to download the software, and all the necessary information to verify the algorithmic implementation of the agent's behavior. The Santa Fe Trail problem has also been addressed based on other algorithms applicable to machine learning, such as genetic algorithms (GA). The last option, although not described in this paper, was also considered in the experimental runs for comparative purposes. It's possible to see, in Figure 5, that the reinforcement learning algorithm of the RBG found an effective strategy after the sixth attempt in about 13 seconds to get all the foods.

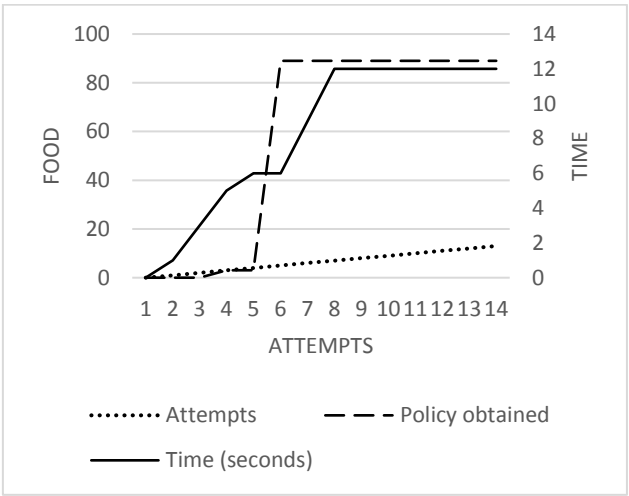

Figure 5: Performance of the Reinforcement Learning Algorithm to learn the best policy based on the Robotic Behavior Grammar applied to the Santa Fe Trail problem

On the other hand, the RBG implemented with Genetic Algorithms, in Figure 6, describes the performance of the best individual, the average individual, the worst individual, and the convergence of them after the eighth generation, between 100 to 140 seconds. The above proves a better performance when it's used the Reinforcement Learning technique according to the procedure described in section 5.

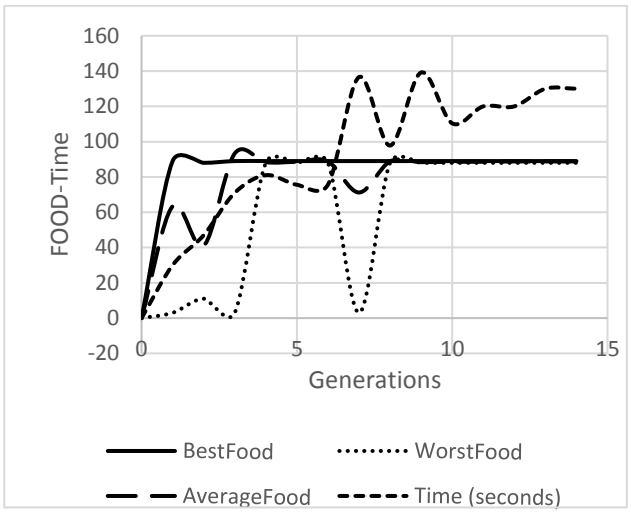

Figure 6: Performance of the Genetic Algorithm to learn the best individual based on the Robotic Behavior Grammar applied to the Santa Fe Trail problem

The results showed that formal grammars are a possible method to represent and generate behavior of agents (robots) that interact with an environment. Besides, the techniques of machine learning, as in this case reinforcement learning, can help in the derivation process of the productions involving these grammars, so the agent can optimally meet your goal. In future work, it is considered to include grammars induction techniques to automate the capture of grammars applicable to unforeseen circumstances or in cases of dynamic environments.

References Références Referencias

1. Russell, S. y Norvig, P. (2009). Artificial Intelligence: A Modern Approach. Prentice Hall Press, (3rd ed.). Upper Saddle River, NJ, USA.
2. Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. MIT Press.
3. Gamez, D., Fountas, Z., y Fidjeland, A. K. (2013). A neuronal controlled computer game avatar with humanlike behavior. Computational Intelligence and AI in Games, IEEE Transactions on, 5(1): 1–14. J.L. Briseño et al.
4. Bauckhage, C., Gorman, B., Thurau, C., y Humphrys, M. (2007). Learning human behavior from analyzing activities in virtual environments. MMI-Interaktiv, 12: 3–17.
5. Hester, T., Quinlan, M., y Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, pp. 2369–2374.
6. Gu, L. y Su, J. (2006). Humanoid robot behavior learning based on art neural network and cross-modality learning. In: Advances in Natural Computation. Springer, pp. 447–450.
7. Ogura, T., Okada, K., Inaba, M., y Inoue, H. (2003). Behavior network acquisition in multisensory space for whole-body humanoid. In: Multisensor Fusion and Integration for Intelligent Systems, MFI2003. Proceedings of IEEE International Conference on. IEEE, pp. 317–322.
8. Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., y Qin, Y. (2004). An integrated theory of the mind. Psychological review, 111(4): 1036.
9. Martın, F., Canas, J. M., Agüero, C., y Perdices, E. (2010). Behavior-based iterative component architecture for robotic applications with the NAO humanoid. En: XI Workshop de Agentes Fısicos. Valencia (Spain).
10. Manikonda, V., Krishnaprasad, P. S., y Hendler, J. (1995). A motion description language and a hybrid architecture for motion planning with nonholonomic robots. In: Robotics and Automation, 1995. Proceedings, 1995 IEEE International Conference on. IEEE Vol. 2, pp. 2021–2028.
11. Dantam, N. y Stillman, M. (2013). The motion grammar: Analysis of a linguistic method for robot control. Robotics, IEEE Transactions on, 29(3): 704-718.
12. Hopcroft J. E. and Ullman J. D. (1979). Introduction to Automata theory, Languages, and Computation. Addison Wesley Publishing Company, Inc.
13. Mitchell, T. M. (1997). Machine learning. 1997. Burr Ridge, IL: McGraw Hill, 45.