Supporting Communication among Cognitive Robots in Simulated Environments

Jesica Devalle, Daniel Cristal, Edgardo Ferretti,* and Marcelo Errecalde

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional
Universidad Nacional de San Luis. (D5700HHW) San Luis, Argentina.
*Corresponding author e-mail: `ferretti@unsl.edu.ar`

Abstract. Despite the fact that the *Khepera* II is an experimental platform widely used in the scientific community related to robotics research, its application to cognitive robotics is not as extensive as it should be. Particularly, in this field of research, the *Khe*-DeLP framework has arisen as an interesting proposal for developing cognitive agents to control real and simulated *Khepera* II robots. Although *Khe*-DeLP allows to work with multiple robots within the same environment, at present, only non-intentional communication among them can be achieved in this framework. Therefore, in this work we present extentions to *Khe*-DeLP to be able to model simulated scenarios where multiple robots interact by using explicit communication among them. This new feature improves *Khe*-DeLP since any kind of coordination problems can be simulated within the framework. As concept test, an example is presented which aims to validate coordinated behaviours of the robots by using the new communication features included in *Khe*-DeLP.

Keywords: Cognitive Robotics, Communication, *Khe*-DeLP.

1 Introduction

Research in robotics in the last two decades was significantly influenced by the *behaviour-based* approach [3] to Artificial Intelligence (AI), which essentially postulates that in order to achieve good performance in a situated agent, like a robot, the agent's ability to properly react to the external environment should be the fundamental aspect to be considered. Nowadays, most of AI researchers recognize the importance of reactivity but also it is out of discussion, that this aspect alone is not enough to create successful situated agents capable of performing complex tasks. For instance, in scenarios with robots skilled with complex social interaction capabilities, high-level representation and reasoning is required.

In this context, where the importance of higher level representation and reasoning in robotics is recognized, several groups have begun to work on what is called *cognitive robotics*. Cognitive robotics intends to capture the application of logical formalisms and computational models of high-level cognitive functions, such as planning, to real-world and simulated robots.

At present, there are available a wide number of low-cost commercial platforms (*e.g.*, LEGO Mindstorms), to experiment with physical robots. Nonetheless, fixing the low-level problems that arise when working with these platforms is a time consuming activity which blurs the high-level problems that cognitive robotics aims to face. In this way, the use of simulators allows to concentrate in the particular problem is aimed to solve without taking care of whether the sensors values are correct or effectors have precision errors. Likewise, often it is needed to integrate in physical and simulated robots already existing cognitive tools whose application has been mainly targeted to software agents.

In this respect, Prolog is a programming language which has been effectively used to develop many applications in the field of cognitive robotics [19, 13, 7, 10, 4] as well as in general applications within AI [5]. For instance, LEGOLOG [19] is a system that allows to program LEGO robots by using primitives in Prolog. Besides, it provides a GOLOG interpreter [20], so that the robots can maintain an explicit world representation and reasoning about different courses of action to get a particular state of the world. Similarly, in [7, 10] was presented *Khe*-DeLP, a framework that provides facilities to program the *Khepera* II robot [17] with high-level specifications based on logic programming and argumentation, specifically *Defeasible Logic Programming* [14].

A key issue of *Khe*-DeLP, is that allows to work with either real or simulated *Khepera* II robots. Besides, it provides a *Ciao* Prolog [6] interface (called *KRolog* [9]) which implements all the sensorial and effectorial capabilities of the *Khepera* II robot and its extension modules (cameras K213 and K6300, and gripper). Nevertheless, at present this interface does not provide the communitation primitives of the *Khepera* II radio extension turret, which implies that in *Khe*-DeLP only scenarios with non-intentional communication can be modeled.

By *non-intentional* communication we mean those scenarios where there is no specific receiver of the information to be sent, and hence, as no explicit communication is involved, information is transmitted by modifying the environment [2], or the visible state of the agents [1]. When working with mobile robots, they should be able to distinguish among each other from their respective perceptions, so as to effectively develop their corresponding activities. Conversely, by *intentional* communication we mean those scenarios where specific devices are used to guarantee the communication from a sender to one o more receivers.

In this way, as *communication* is a key issue when developing multi-agent systems, in this paper we present a proposal to extend *Khe*-DeLP so that *KRolog* provides the communication primitives to allow intentional communication among simulated *Khepera* II robots. Besides, as concept test, an example is presented aimed to validate coordinated behaviours of the robots by using the new communication features included in *Khe*-DeLP.

The paper is organized as follows: Sect. 2 gives a brief overview of the *Khe*-DeLP framework. Then, Sect. 3 shows through a simple example how to program in *Khe*-DeLP, the communication among simulated *Khepera* II robots by using the new primitives implemented to this end. Finally, Sect. 4 puts forward conclusions and future work.

2 *Khe*-DeLP in a nutshell

Khe-DeLP [7, 10] is a layered framework that provides to the Khepera community facilities for programming high-level robots' behaviours using an argumentative approach. In *Khe*-DeLP lower-level layers hide low-level robot-computer communication providing a high-order set of predicates to interact with real and simulated *Khepera* II robots. Moreover, upper layers are dedicated to cognitive robotics implementations. The most abstract layer in this framework allows Defeasible Logic Programming (DeLP), which provides support for knowledge representation and high-level reasoning capabilities. DeLP is based on an argumentative formalism suitable for reasoning in real and dynamic environments, *i.e.*, scenarios where the information that the robot has about its environment, is usually incomplete or contradictory.

Besides, *Khe*-DeLP provides a *Ciao* Prolog interface, called *KRolog*. This interface consists of 46 predicates which implement all the sensorial and effectorial capabilities of the *Khepera* II robot and its extension modules. In Sect. 3 only the three predicates explained below are used in the example:[1]

– `move_forward_distance(+D,-Outb)`: Makes the robot move forward `D` millimeters passed as input parameter.
– `turn_left(+Dg,-Outb)`: Makes the robot turn left `Dg` degrees passed as input parameter.
– `turn_right(+Dg,-Outb)`: Makes the robot turn right `Dg` degrees passed as input parameter.

All the predicates in *KRolog* has an output parameter `Outb` where it is returned a boolean atom to indicate whether the predicate was successfully performed or not. In Fig. 1 *KRolog* is referred as "Sensorial/Effectorial layer."

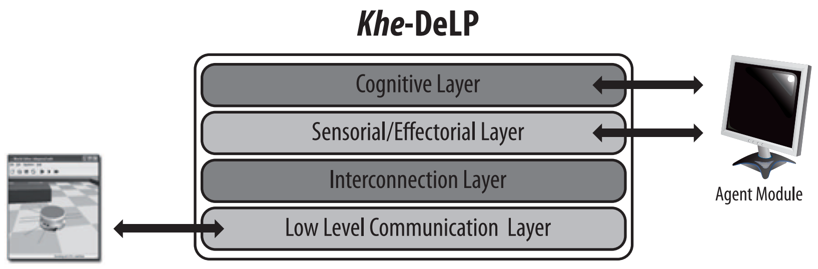


Fig. 1. The *Khe*-DeLP framework scheme

[1] In these predicates, parameters are denoted following the usual notation in Prolog, where '+' refers to an input parameter while '−' to an output parameter, respectively.

As stated in the introductory section, in this paper only simulated environments are considered. In this respect, it is worth mentioning that *Khe*-DeLP has been designed to work with the professional 3D simulator called *Webots* [22], and hence much of the work reported on Sect. 3 is based on particular features of this simulator.

3 Programming the Robots Communication

Nowadays, there is a new trend of applications in which the coordinated control of a robotic system is a key issue. In this respect, collective robotics aims at designing multi-robot systems to solve problems whose resolutions cannot be faced by a single robot, or designing such a robot is a costly task [18]. In this way, having a system composed of simpler robots in terms of design, allows a better performance in activities like rescue operations, search and exploration, cooperative object manipulation, among others.

It is worth mentioning that all the above-mentioned activities need to manage formation control of the robots to perform efficiently. That is why, Example 1 aims to illustrate the communications features of *Khe*-DeLP, facing this matter.

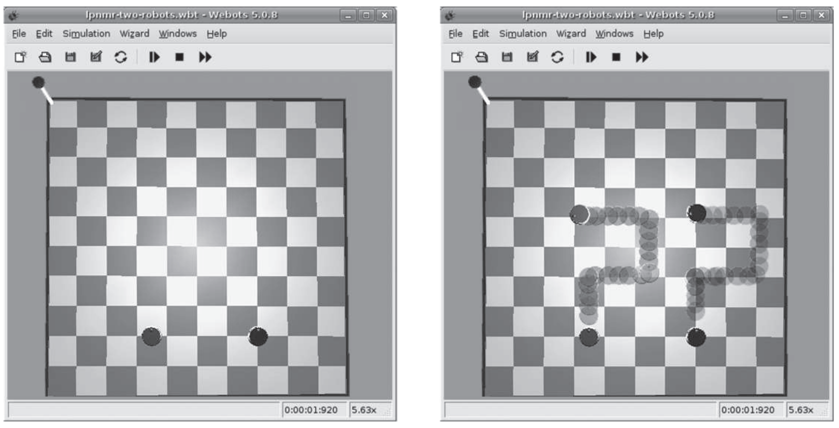
Example 1. Let us consider the situation depicted in Fig. 2(a) where there are two simulated *Khepera* II robots in a square arena of 100 units per side which is conceptually divided into square cells of 10 units per side each. The leftmost robot, from now on referred as K_0, is the team leader running the program shown in Fig. 3(a). In this case, K_0 program code is very simple since it has to follow a predetermined path and before executing each action, by calling the `send_msg_to_turret/3` predicate, it sends to K_1 (the rightmost robot in Fig. 2(a)) each command it will execute.

The predicate `send_msg_to_turret(+Dest_id,+Msg,-Outb)` receives as first input parameter the radio turret ID (an integer in the range $1, \ldots, 31)$[2] to whom the message will be sent. The message is the second input parameter and must be enconded as a list of atoms, while the third parameter is used to return a boolean value as output indicating whether the transmision was succesful or a failure occured.

For instance, lines (4), (6), (8), (10), (12) (14) and (16) of Fig. 3(a) show the use of this predicate to communicate to K_1 which action it should perform. That is why, the first argument in all these calls is 1 (K_1 has been assigned 1 as turret ID). Besides, all these calls have as third argument an atom, `true` in this case, so that the execution be aborted if any of these predicates cannot trasmit the message due to a communication failure.

In this particular case, the message to be transmitted as second argument of the `send_msg_to_turret` predicate, has been codified with a number to simplify the example. For instance, when K_1 receives number 1 as the command to be executed, this will imply that a `move_forward_distance(200,Z)` command will

[2] This limitation in the number of turrets ID is maintained to be consistent with the number of physical radio turrets that the radio base is able to manage.


lpnmr-two-robots.wbt - Webots 5.0.8
File Edit Simulation Wizard Windows Help
0:00:01:920
5.63x

(a) (b)

Fig. 2. Experimental environment

be issued (see lines 23 and 24 of Fig. 3(b), and lines (5), (9), (13) and (17) of Fig. 3(a)). Similarly, the commands `turn_right/2` and `turn_left/2` have been codified with numbers 2 and 3, respectively (lines (25)-(28) of Fig. 3(b), and lines (7), (11) and (15) of Fig. 3(a)).

Figure 3(b) shows K_1 program code. In the same way that K_0, K_1 program also import the *Khe*-DeLP module (line 1), but K_1 program code need to include other predicates in addition to `behave` (line 19 in Fig. 3(b)), the predicate which control the robot's behaviour. Lines (3) to (9) define the `live` predicate. Lines (4) and (9) call predicates to set up and shut down, respectively, the existing Prolog to Java interface provided in Ciao. Lines (5) and (6) are very important since they define the predicates `receive_msg_from_turret/1` and `behave/0` as concurrent ones. These predicates run in different threads which will communicate among each other by using K_1 knowledge base. Indeed, this communication will be achieved by inserting and deleting the predicate `turret_input/1` from K_1 knowledge base. Line 2 defines this predicate to be used in such a way. Finally, line (7) waits that thread with `Bid` as ID, which is running the `behave` predicate to finish its execution, while line (8) releases all the resources assigned to this thread.

The predicate `receive_msg_from_turret(+Server_id)` (line (11) of Fig. 3(b)) explicitly waits for a message from the team leader to replicate its actions. This predicate receives as input parameter the Java server ID which is dedicated to listen messages send to K_1. This task is perfomed by calling the predicate `wait_for_message(+S,-Msg)` which invokes predefinite predicates of the Java to Prolog interface to get the incoming message from Server `S`. The message received is returned as a list of atoms in `Msg` and it is inserted in the knowledge base (line (13)).

```
(1)  :-use_module(khe_delp).            :-use_module(khe_delp).

(2)                                     :- concurrent turret_input/1.

(3)  behave:-                           live:-
(4)  send_msg_to_turret(1,[1],true),    java_init('test.SocketServer',Serv,createSocket(_)),
(5)  move_forward_distance(200,true),   eng_call(receive_msg_from_turret(Serv),create,create),
(6)  send_msg_to_turret(1,[2],true),    eng_call(behave,create,create,Bid),
(7)  turn_right(90,true),               eng_wait(Bid),
(8)  send_msg_to_turret(1,[1],true),    eng_release(Bid),
(9)  move_forward_distance(200,true),   java_end(Serv,closeSocket(_)).
(10) send_msg_to_turret(1,[3],true),
(11) turn_left(90,true),                receive_msg_from_turret(S):-
(12) send_msg_to_turret(1,[1],true),    wait_for_message(S,Msg),
(13) move_forward_distance(200,true),   asserta(turret_input(Msg)),
(14) send_msg_to_turret(1,[3],true),    receive_msg_from_turret(S).
(15) turn_left(90,true),
(16) send_msg_to_turret(1,[1],true),    wait_for_message(S,Msg):-
(17) move_forward_distance(200,true).   java_invoke_method(S,getMsgFromTurret(_)),
(18)                                    java_invoke_method(S,getValue(Msg)).

(19)                                    behave:-
(20)                                    retract_fact_nb(turret_input([_,Act])),
(21)                                    execute(Act),
(22)                                    behave.

(23)                                    execute(1):-
(24)                                    move_forward_distance(200,true).

(25)                                    execute(2):-
(26)                                    turn_right(90,true).

(27)                                    execute(3):-
(28)                                    turn_left(90,true).
```

(a) K_0 (b) K_1

Fig. 3. Robots' program codes

While the thread running the `receive_msg_from_turret` predicate is continuously listening its associated Java Server, in turn, the `behave` predicate is expecting that a fact of the kind `turret_input(Msg)` be available in the knowledge base (line (20)). When this happens, this fact is deleted from the knowledge base and the list it had as argument is processed. In this particular example, processing the argument only consists of ignoring the first element of `Msg` (which is the sender ID) and using the second element `Act` as input argument to the `execute/1` predicate (line 21). After the execution of this action, the follower waits for a new instruction of the team leader and the process continuous as already described.

The above-mentioned discussion analysed how communication is achieved among the robots from a functional view. To complement this view, Fig. 4 shows how the interaction among the layers of *Khe*-DeLP is carried out when message passing occurs. Terminals labelled as K_0 and K_1 are running the agents modules depicted in Fig. 3. As K_0 is the team leader, before executing an action, by calling the predicate `send_msg_to_turret` the Prolog application connects the controller of K_0 within Webots (arrow 1 in Fig. 4), which in turn triggers the message to

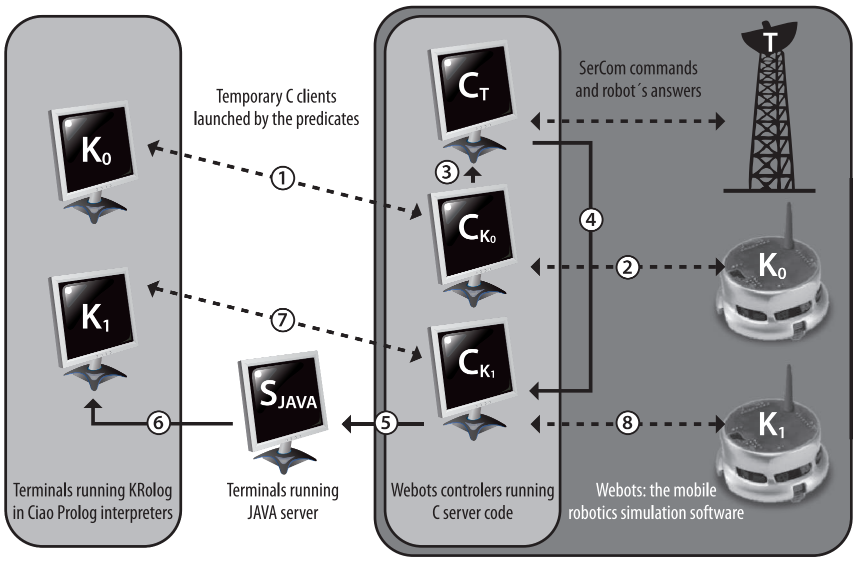

Temporary C clients launched by the predicates
SerCom commands and robot´s answers
T
C_T
K_0
C_{K_0}
K_0
K_1
C_{K_1}
S_{JAVA}
K_1
1
2
3
4
5
6
7
8
Terminals running KRolog in Ciao Prolog interpreters
Terminals running JAVA server
Webots controlers running C server code
Webots: the mobile robotics simulation software

Fig. 4. Layers' interaction when message passing occurs within *Khe*-DeLP

the radio base controller C_T (arrow 3). Then controller C_{K_0} is contacted again by K_0 to perform on the simulated robot the action previously passed as a message (arrow 2). Concurrently, controller C_T processes this message and forward it to the appropriate robot, K_1 in this case (arrow 4).

When C_{K_1} receives from C_T the action to perform, originally issued by K_0, instead of executing this action on the simulated robot K_1, it connects to a Java Server (belonging to the Interconection layer) to inform to agent module K_1 the message it received. From a conceptual view, C_{K_1} does not execute the action on K_1 given that all the actions to be performed on the robot should be issued from its respective agent module. Also, it must be noted that in this particular case the message consists of a primitive action of *KRolog* which could be carried out by the controller, but it could also be a high-level action involving further processing of agent module K_1, before sending a robot-specific action to the controller. In this way, when agent module K_1 receives this message (arrow 6) and process it, then this action is triggered back to C_{K_1} (arrow 7) and it executes it on the simulated robot K_1 (arrow 8).

As it can be noted, the controllers (low-level layer) are in charge of managing the dynamics on the environment (action execution and message passing) by modifying the properties of the objetcs (structured as VRML nodes in Webots) representing the radio base (labelled as T) and the *Khepera* II robots.

In the above-described example only two robots were considered, but any potential number of robots can be together passing messages among each other. We have not analysed yet how the communication overhead increases with respect to the number of robots communicating among each other, but as it will be stated in Sect. 4 we aim to face this issue as future work. To conclude this discussion it remains to note that Fig. 2(b) shows snapshots of the robots performance as result of the execution of programs depicted in Fig. 3.

3.1 Implementation issues

Khe-DeLP is delivered as a set of *Ciao* Prolog modules, thus any Prolog application which imports the main module, called `khe_delp.pl`, gets access to all the primitives of *KRolog* to program *Khepera* II robots in a declarative manner. Besides, by importing this module, the high-level services for reasoning and knowledge representation provided by DeLP-Server (the interpreter of DeLP) [12] are also accessed. When working with Webots, the simulator must be executed first and then each agent module will work with it by the port assigned to each robot.

There is one implementation issue which is worth mentioning. It is concerned with predicates `send_msg_to_turret` and `receive_msg_from_turret`. The communicaton primitives of the real *Khepera* II robot are limited to send 16 bytes per message. In fact, if messages are shorter then they are padded to get this amount of bytes. Thus, 16 bytes are always transmited by communication channels. Besides, as communication channels are usually noisy, data encryption and decryption is perfomed by the sender and receiver, respectively. In our implementation, as communication takes place within Webots, reading and writing internal buffers, thus neither encryption nor decryption of data is carried out. Moreover, the restriction of transmitting 16 bytes per message is not considered.

In this way, the radio base VRML node created in Webots, like the VRML node representing the *Khepera* II radio turret can be used by other applications running on Webots not depending on *Khe*-DeLP. Nonetheless, none of these applications will be able of cross-compiling the codes to be transferred to real *Khepera* II robots, as a consequence of not controlling the messages length to be sent. Data encryption and decryption would not be a problem, since this is automatically done by the radio turrets of the robots acting as sender and receiver, respectively.

4 Conclusions and Future work

Communication has been recognised as a very important issue within the field of research of Computer Science, where a great number of formalisms [16] have been developed to represent the different features that arise in concurrent systems. Particularly, when working with robots, communication among them increase their capacities and effectiveness. Likewise, communication also allows to restate hypothesis and get a deeper understanding of how interaction mechanisms evolve in systems, not necessarily artificials [21].

In this respect, the *Khe*-DeLP framework has been used to develop cognitive applications where multiple robots cooperatively works by using non-intentional communication [11, 8]. As stated in the introductory section, *Khe*-DeLP lacked communication primitives so that the *Khepera* II robots could explicitly communicate among each other. Therefore in this work, we have extended *Khe*-DeLP by programming the communication support, so that simulated robots be able of communicating among them in an explicit way. Through a simple example the new features of *Khe*-DeLP were presented. Besides, it was described how to program agent modules to achieve a successful communication among them. Finally, with the aim of getting a better understanding of why these agent modules should be programmed as described, it was also discussed how the messages were managed within *Khe*-DeLP.

At present, the Webots software used by *Khe*-DeLP to work with simulated robots is the only propietary platform, because both Ciao and DeLP-Server like the *KRobot* class[3] are freely-available technologies. Thus, as long-term future work, it is intended to modify *Khe*-DeLP to work with an open-source simulator as well.

As future short-term work it is planned to make experimental studies to evaluate how the communication overhead increases with respect to the number of robots communication among each other, within the framework. Moreover, it is also intended to program the necessary support so that intentional communication experiments can be performed with real *Khepera* II robots. In opposition to the current implementation, when this stage be performed, the low-level layer will have to control that messages conform the restrictions of the physical communication channels.

Acknowledgements

The authors Edgardo Ferretti and Marcelo Errecalde thank the Universidad Nacional de San Luis and the ANPCyT from which receive continuous support.

References

1. Arkin, R.C., Balch, T., Nitz, E.: Communication of behavorial state in multi-agent retrieval tasks. In: Proceedings of the 1993 IEEE International Conference on Robotics and Automation. vol. 3, pp. 588–594 (May 1993)
2. Arkin, R.C.: Cooperation without communication: Multiagent schema-based robot navigation. Journal of Robotic Systems 9(3), 351–364 (1992)
3. Arkin, R.C.: Behavior-Based Robotics. The MIT Press (1998)
4. Billington, D., Estivill-Castro, V., Hexel, R., Rock, A.: Architecture for hybrid robotic behavior. In: Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems. pp. 145–156 (2009)
5. Bratko, I.: Prolog Programming for Artificial Intelligence. Addison-Wesley (2001)
6. Bueno, F., Cabeza, D., Carro, M., Hermenegildo, M., López-García, P., Puebla, G.: The ciao prolog system. http://www.clip.dia.fi.upm.es/ (1997)

[3] Developed by Harlan *et al.* [15], this C++ class manages the serial port communication with the robot, hiding low-level robot-computer communication and allowing developers to focus on the robot/environment interaction.

7. Ferretti, E., Errecalde, M., García, A., Simari, G.: KheDeLP: A Framework to Support Defeasible Logic Programming for the Khepera Robots. In: Mayorga, R.V., Domínguez, O.A. (eds.) International Symposium on Robotics and Automation (ISRA). pp. 98–103. San Miguel Regla, Hidalgo, México (August 25-28 2006)
8. Ferretti, E., Errecalde, M., García, A., Simari, G.: Defeasible decision making in a robotic environment. In: Anales del XIII Congreso Argentino de Ciencias de la Computación (CACIC). pp. 1335–1446 (Octubre 2-5 2007)
9. Ferretti, E., Errecalde, M., Simari, G.: KRolog: A prolog based interface for the khepera robots. In: Actas del Octavo Workshop de Investigadores en Ciencias de la Computación (WICC 2006). pp. 17–21 (Junio 2006)
10. Ferretti, E., Errecalde, M.L., García, A.J., Simari, G.R.: Khepera robots with argumentative reasoning. In: Rückert, U., Sitte, J., Witkowski, U. (eds.) Proceedings of the 4th International AMIRE Symposium. pp. 199–206. No. 216 in AMIRE, Heinz Nixdorf Institut, Universität Paderborn, Buenos Aires, Argentina (October 2007)
11. Ferretti, E., Rotstein, N., Errecalde, M., García, A., Simari, G.: Defeasible decision making in a multi-robot environment. Research in Computing Science 32, 150–160 (September 2007), special Issue: Advances in Artificial Intelligence and Applications
12. García, A., Rotstein, N., Tucat, M., Simari, G.: An argumentative reasoning service for deliberative agents. In: Zhang, Z., Siekmann, J. (eds.) Proceedings of the 2nd. International Conference on Knowledge Science, Engineering and Management (KSEM). Lecture Notes in Artificial Intelligence, vol. 4798, pp. 128–139. Springer-Verlag (2007)
13. García, A.J., Simari, G.I., Delladio, T.: Designing an agent system for controlling a robotic soccer team. In: X Computer Science Argentine Conceference. pp. 1646–1656 (2004)
14. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming 4(2), 95–138 (2004)
15. Harlan, R.M., Levine, D.B., McClarigan, S.: The khepera robot and the krobot class: a platform for introducing robotics in the undergraduate curriculum. SIGCSE Bulletin 33(1), 105–109 (2001)
16. Huget, M.P. (ed.): Communication in Multiagent Systems, Agent Communication Languages and Conversation Policies, Lecture Notes in Computer Science, vol. 2650. Springer (2003)
17. K-Team: Khepera 2. http://www.k-team.com (2002), a miniature mobile robot designed as a research and teaching tool
18. León-Fernández, Y., Meléndez, A.M.: Generación de formaciones estáticas en robótica colectiva. Tech. Rep. CCC-04-008, Coordinación de Ciencias Computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, México (Diciembre 2004)
19. Levesque, H.J., Pagnucco, M.: Legolog: Inexpensive experiments in cognitive robotics. In: 2nd International Cognitive Robotics Workshop. pp. 104–109. Berlin, Germany (August 2000)
20. Levesque, H.J., Reiter, R., Lesperance, Y., Lin, F., Scherl, R.B.: GOLOG: A Logic Programming Language for Dynamic Domains. Journal of Logic Programming 31(1-3), 59–83 (1997)
21. Lipson, H.: Evolutionary robotics: Emergence of communication. Current Biology 17(9), 330–332 (2007)
22. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems 1(1), 39–42 (2004)