DALI for Cognitive Robotics: Principles and Prototype Implementation

Stefania Costantini(✉), Giovanni De Gasperis, and Giulio Nazzicone

DISIM, Università di L'Aquila, L'Aquila, Italy
stefania.costantini@univaq.it

Abstract. DALI is a logic Prolog-based Multi Agent System Language and Framework (publicly available on GitHub) developed at University of L'Aquila since 1999, and includes features aimed at user monitoring and training in Ambient Intelligent applications. In this paper, we show how such features can be integrated and extended in view of cognitive robotic applications; we then illustrate the extensions to the DALI implementation that allow DALI agents to interact with robotic platforms even through the cloud.

1 Introduction

Quoting from http://www.ieee-ras.org/cognitive-robotics, "There is growing need for robots that can interact safely with people in everyday situations. These robots have to be able to anticipate the effects of their own actions as well as the actions and needs of the people around them. To achieve this, two streams of research need to merge, one concerned with physical systems specifically designed to interact with unconstrained environments and another focusing on control architectures that explicitly take into account the need to acquire and use experience."

Several papers on cognitive robotics can be found in the proceedings of main Conferences on Artificial Intelligence (e.g., ECAI, IJCAI) and on Agents (e.g., AAMAS). The importance of developing "intelligent" adaptive robots can be particularly appreciated in view of the societal issue of helping the elderly and the disabled; in fact life expectancy is increased, and consequently the number of persons needing personalized assistance is increasing as well. ICT (Information and Communication Technologies) can potentially (and partly already are) of great help. According to the guidelines provided by the European Union (http://ec.europa.eu/health/ehealth/policy/index_en.htm), the eHealth scenario "refers to tools and services using information and communication technologies (ICTs)" that can improve prevention, diagnosis, treatment, monitoring and management but, also, "Bringing ICT and healthcare together is not simply a matter of digitizing and communicating matters of health, but rather opening a new world of doing things in ways that were not possible or even conceivable before...". Cognitive robotic systems can potentially also help in any situation where there is an impaired or disabled person, or more generally any person in need of special

Y. Lierler and W. Taha (Eds.): PADL 2017, LNCS 10137, pp. 152–162, 2017.
DOI: 10.1007/978-3-319-51676-9_10

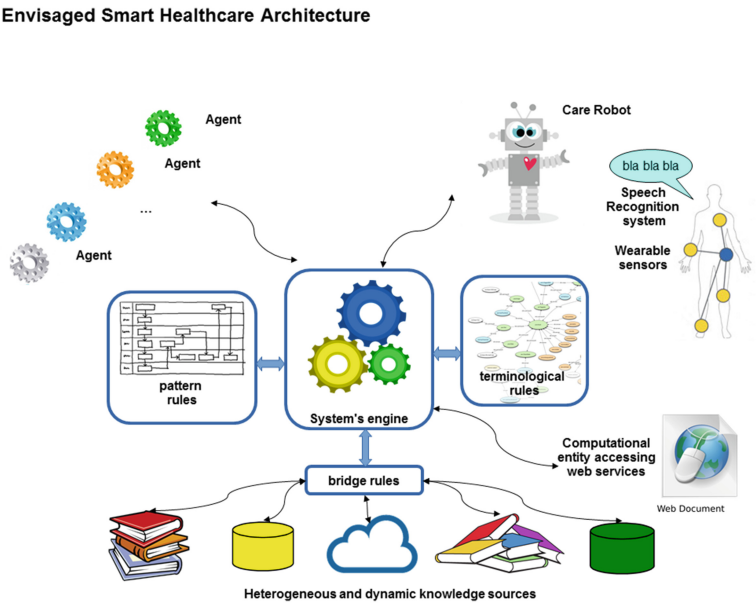

Fig. 1. Envisaged Smart Healtcare Architecture

assistance. This may include children, not as a substitute but as a support to parents, family and caregivers.

In [1] we have outlined a comprehensive system (depicted in Fig. 1 and called Friendly&Kind, for short F&K) based upon logical agents for monitoring patients, where a patient's personal agent can be embodied in a robot. Some of the monitoring can be performed locally, where however agents are able, upon need, to interact with human specialists, knowledge bases, and with the health system. This while performing routine care chores and, possibly, entertaining the patient. This should alleviate the healthcare and social security systems from the burden of having to provide full-time highly specialized human assistants while allowing elderly people to leave at home rather than be moved to an institution. The level of synergic knowledge retrieval and integration that such a system can provide goes beyond the capabilities of a human nurse or personal assistant.

It has been demonstrated that, understandably, humans prefer friendly interfaces and robots that show some kind of intelligent and also affective and "emotional" behavior. There is interesting ongoing work, e.g., by the group of Prof. Johan Hoorn at Vrije Universiteit Amsterdam about social robotics, which considers the impact of robots on the user from the point of view of ethics [2], interaction with the disabled [3] and even acceptable robot appearance. Some of this work is reported in a famous documentary "Alice cares" (https://vimeo.com/116760085, a scene from this documentary is reported in Fig. 2), which shows the positive interaction among three old women and a friendly humanoid care robot. From a cognitive point of view and for exploring

Fig. 2. Text reading robot assistant, courtesy of "Alice cares" trailer video from Vimeo

the social acceptability of robots as human companions these experiments are certainly of great importance. However, as concerns "intelligent" behavior the robots used in the experiments are still under remote control of a human operator.

In the perspective of making such robots really intelligent and autonomous, research results from many fields of Artificial Intelligence, Automated Reasoning and Intelligent Software Agents can be usefully exploited. We strongly believe that in this and in other fields it can be advantageous to define a robot's cognitive part as an agent or Multi-Agent System (MAS) defined via declarative agent-oriented languages. In fact, the behavior of a robot which is aimed at assisting a human user should be based on user observation, monitoring and training. I.e., the robot should be equipped with a basic user profile defining the user's needs, habits, and preferences; such profile should be then refined via the robot's own observation of the user's behavior over time; the robot should then be able to supervise and check the user's activities, and to teach (or remind) a user of how to perform tasks. In all this, the robot should be able to recognize relevant complex events from sets of simple ones, and to check and to adapt its own behavior upon changing circumstances.

There are many logic agent-oriented languages and architectures in computational logic apt to these aims, among which MetateM, 3APL, GOAL, AgentSpeak, Impact, KGP and DALI (the reader may refer to the surveys [4–6] and to the references therein), that might be usefully exploited in robotics, as in fact many of them already have, or at least many of the examples provided in the literature concern potential robotic applications.

The DALI language [11,12] has been empowered and experimented over the years concerning capabilities for the definition and management of an agent's memory and experience and for user monitoring and training also by learning new behavioral patterns (via deep learning or via knowledge exchange); DALI agents are able to perform complex event processing, and to dynamically check

and modify their own behavior also in terms of a special interval temporal logic (cf. [7–10] and the references therein). However all these features, though experimented in software agents, have never been applied since recently to robotic applications because DALI lacked a suitable plug-in, that we have now developed.

Such extension to the basic DALI implementation allows action commands to be exchanged between DALI agents and any robotic platform by using the YARP middleware. In addition, we have implemented ServerDALI which allows to locate DALI agents and MAS on a server. This is relevant, as for instance in the architecture of Fig. 1 the caregiver agents will presumably be copies of the same one, to which robots' cognitive functioning can refer; so, a cloud solution eliminates the need of equipping the (possibly diverse) robot hardware with sophisticated software; moreover, computationally heavy automated reasoning tasks can be more efficiently executed on the server.

The novel contribution of this paper is twofold: on the one hand, we have re-elaborated and extended past work on DALI in the perspective of robotic applications for the care of persons in need; on the other hand, we have realized and experimented a practical efficient implementation constructed out of open-source components. At the present stage, we have been experimenting the use of a declarative language for defining the cognitive part of robots, and for our experiments we have adopted simulators rather than real robot hardware. So, in this context we are not concerned with physical aspects concerning sensors, actuators, vision, etc., that are however widely studied by specialists.

In Sect. 2 we recall the basic DALI language, while in Sect. 3 we discuss, also by means of small though significant examples the potential applicability of DALI in robotic user monitoring and training. In Sect. 4 we illustrate the extension to robotics of the DALI implementation. Finally, in Sect. 5 we conclude.

2 The Basic DALI Language and Architecture

DALI [11,12] (cf. [13] for a comprehensive list of references) is an Agent-Oriented Logic Programming language. The DALI Prolog-based Multi Agent System Language and Framework has been developed at University of L'Aquila since 1999.

DALI agents are able to deal with several kinds of events: external events, internal, present and past events.

External events are syntactically indicated by the postfix *E*. Reaction to each such event is defined by a reactive rule, of the form $EvE :> Reaction$ where $:>$ is a special token. The agent remembers to have reacted by converting an external event into a *past event* (postfix *P*). An event perceived but not yet reacted to is called "present event" and is indicated by postfix *N*.

In DALI, **actions** (indicated with postfix *A*) may have or not preconditions: in the former case, the actions are defined by actions rules, in the latter case they are just action atoms. An action rule is characterized by the new token $:<$. Similarly to events, actions are recorded as past actions.

Internal events is the feature that makes DALI agent agents proactive. An internal event is syntactically indicated by the postfix *I*, and its description is

composed of two rules. The first one contains the conditions (knowledge, past events, procedures, etc.) that must be true so that the reaction (in the second rule) may happen. Thus, a DALI agent is able to react to its own conclusions. Internal events are automatically attempted with a default frequency customizable by means of directives in the initialization file.

The DALI communication architecture implements the DALI/FIPA protocol, which consists of the main FIPA primitives, plus few new primitives which are particular to DALI and provides the possibility of defining meta-rules for filtering incoming and out-coming messages, and for accessing and querying external ontologies in the semantics web.

DALI provides a plugin to an answer set solver, so complex reasoning tasks such as, e.g., planning and preference handling can be performed in Answer Set Programming (ASP), which is a state-of-the art technology for dealing with hard computational problems (cf., among many, [14] and the references therein); several efficient ASP solvers are in fact freely available and are periodically checked and compared over well-established benchmarks, and over challenging sample applications proposed at the yearly ASP competition (cf. the ASPCOMP web sites).

3 DALI Advanced Features and Possible Applications to Robotics

The robotic applications that we particularly envisage concern (since [15]) user monitoring and training in any context, but especially for the care of elderly and disabled persons. In our setting, agents interact with users (i) with the objective of training a user in some particular task, and (ii) with the aim of monitoring the user for ensuring some degree of consistence and coherence in user behavior.

Agents are able to be aware, by prior knowledge or via some form of learning, of the behavioral patterns that the user is adopting, and to learn rules and plans also from other agents (by imitation or by being told). Assume as a simple example that an agent has been somehow able to learn that the user normally takes a drink when coming back home. This can be represented by a rule such as:

$$drink \text{ :- } arrive\_home.$$

This learned rule can possibly be associated with a certainty factor. When the rule becomes later confronted with subsequent experience, its certainty factor will be updated accordingly. Whenever this factor exceeds a threshold, this may lead to assert new meta-rules, such as:

$$USUALLY\ drink\ WHEN\ arrive\_home.$$

User monitoring can be performed via temporal-logic-like rules like the following one:

$$NEVER\ drink\_alchool\ AND\ take\_medicine.$$

Such a rule acts as a constraint which has priority over former ones; so, the agent will actively discourage the user to drink while taking medicines. In [16] the semantics of such expressions is defined, also in relation to the possibility of defining the *interval* where some events/actions must or must not occur.

The following example concerns a robot aiding to supervise a baby, thus relieving caregivers from some of their tasks. If the baby is hungry, the robot should feed the baby with available baby food (feeding is an action, indicated with postfix A) paying attention to choose the healthier among those that the baby likes. Conjunction $food(F)$, $available(F)$ provides a number of values for F, among which one is chosen. In particular, the choice will correspond to a maximum in the partial order imposed by the binary predicates *best_preferred* and *healthier* in the given order. This construct for complex preference, the p-set, was originally introduced in [17].

$$baby\_is\_hungryE :> \{feed\_babyA(F) : food(F),\ available\_babyf(F) :\ best\_preferred, healthier\}.$$

In the example below, the robot again assists parents taking care of a child. The child has to go to school (mandatory goal, indicated by postfix G) and is about to skip breakfast because she prefers cereals that unfortunately are finished. The agent, based upon the monitoring condition (never skip breakfast) will be able to suggest alternative food, in particular the best preferred among available options.

$$go\_to\_schoolG :\ NEVER\ skip\_breakfast(D) ::\ cereals\_finished :::\ suggestA(alternative\_food)\ IN\ \{cookies, cake\_slice : best\_preferred\}.$$

The monitoring component can however also include meta-axioms such as for instance the following one, which states that a user action which is necessary to reach a mandatory objective should necessarily be undertaken. The agent can fulfill this statement either by convincing the user to do so, or to resort to human caregivers' help:

$$ALWAYS\ do(user, A)\ \ WHEN\ mandatory\_goal(G),\ required(G, A)$$

Such a meta-rule could be applied to practical cases such as the following:

$$mandatory\_goal(healthy).$$
$$required(healthy,\ take\_medicineA).$$

ASP modules can be exploited in order to plan actions which might be performed in given situations, and to extract *necessary* actions, which are those actions included in all possible plans. Given ASP module M (defined in a separate text file), in the example below reaction to event evE can be either any action which can be inferred (from M) as a possible reaction, or a *necessary* action, again according to M. Events are indicated with postfix E, reaction is indicated with $:>$. Connective $>$ expresses preference: the former option is preferred over the latter if the condition after the :- holds; *necessary* and *action* are

distinguished predicates applicable over ASP modules' results. So, in this sample rule necessary actions are preferred in a critical situation. Otherwise, any of the two options may be taken.

$$evE :> necessary(M, N) | action(M, A) \ : \ M > A \text{ :- } critical\_situation.$$

The above examples are witnesses of a re-elaboration of past work on DALI in the perspective of cognitive robotics applications. Though small, the examples should have practically demonstrated that DALI has indeed the potential for acting as an agent language in this realm. However, a suitable interface between DALI agents and robotic hardware or simulators was lacking. Such an interface has been recently implemented, and is presented in the next section.

4 The Extended DALI Implementation

The DALI programming environment at the current stage of development [18] offers a multi-platform folder environment including Sicstus Prolog programs (as DALI is implemented in Sicstus), shells scripts, and Python scripts.

For the development of DALI agents and MAS, a programmer can simply use any text editor to write DALI agents' programs and the necessary start/configuration scripts; more proficiently, she could use a web-based system-independent integrated development environment where agents editing is managed through an HTML5/AJAX-based online editor, with start/stop command buttons and agents logs output for runtime verification, handling signals and events from the DALI engine running in the background. The system is designed so as to be able to interact with other services by means of JSON data events. Such an external service can be a virtual robotics simulator. Thus, an entire complex anthropomorphic cognitive robot like the iCub [19] could be controlled by a DALI MAS.

The software components diagram in Fig. 3 shows how DALI has been encapsulated and integrated with other modules through a Python "glue code" layer, called PyDALI. Each DALI agent is an instance of the Prolog program "DALI Interpreter". The multi-platform open source library *pexpect* (http://github.com/pexpect/pexpect) has been adopted for building a Python middle layer to automate the interaction with the Sicstus Prolog environment, seen as an instance of the class PySicstus. In this way, by abstracting via the PyDALI class, a DALI agent instance process can be configured, loaded, started, executed and terminated. A MAS can then be handled via the most abstract class "MAS".

The Python code can then been imported in any Python program by using the open source Twisted (http://github.com/twisted/twisted) programming library. This allows the interaction of DALI agents with other software modules/server/clients by means of asynchronous JSON events. In particular, what we call the *Multi-standard DALI Bus* is in practice a middle layer communication protocol that converts any JSON event coming from the outside

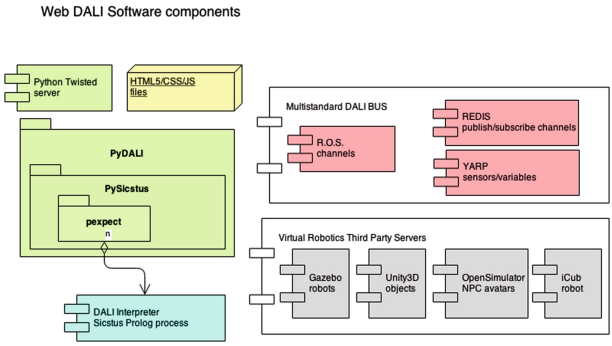

Fig. 3. Software components diagram of the extended DALI architecture

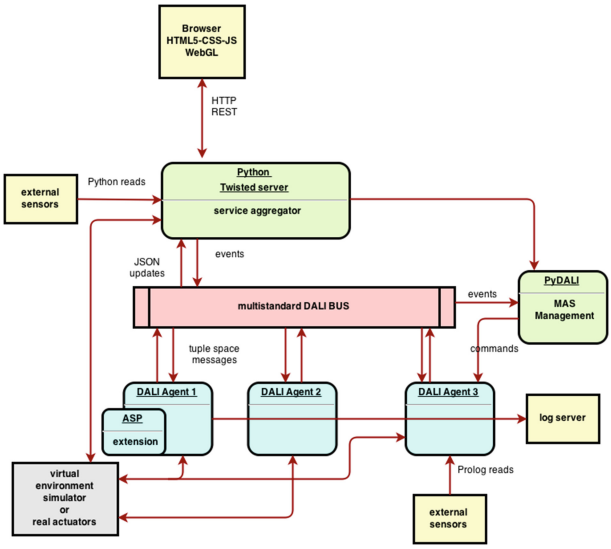

Fig. 4. Runtime deployment diagram of the extended DALI architecture

world to an internal FIPA event in a Linda tuple space[1], that the DALI MAS thus receives as an external event. Specific actions performed within the DALI MAS can generate FIPA events that are converted to JSON event so as to send commands to external actuators, that can be either real robotic actuators or virtual robotic components in a virtual robotics simulator. A typical runtime deployment diagram can be seen in Fig. 4.

[1] *Linda* is a model of coordination and communication among parallel processes providing a logically global *associative memory*, called a "tuplespace", in which processes store and retrieve tuples. It is available for Sicstus Prolog and it is therefore used as a communication middleware in the DALI implementation.

The central Multi-standard DALI bus collects asynchronous data events from different sources, translating them into counterparts in the Linda tuple space whenever an agent is the destination. It also collects action messages from agents and translates them into JSON structures compatible with the destination, through the Python service aggregator/container. There may also be external sensors that directly generate Linda tuple messages, or external sensors mediated by the Python container.

YARP Integration. YARP, "Yet An other Robotic Platform" (http://github.com/robotology/yarp) "*supports building a robot control system as a collection of programs communicating in a peer-to-peer way, with an extensible family of connection types (tcp, udp, multicast, local, MPI, mjpg-over-http, XML/RPC, tcpros,...) that can be swapped in and out to match your needs.*". A C++ program, typically embedded in a robot, generates raw data and sends them to the YARP port "/sender". This port could be connected to a "/receiver" YARP port by means of a channel configurator. We have developed a simple Python program which registers itself as the handler of the "/receiver" port, and translates the data into a Linda tuple space accessible by DALI agents.

DALI MAS Controlling the iCub Virtual Robot. "*The iCub is a 53 degree-of-freedom cognitive humanoid robot which has been developed as an open-systems research platform*" [19]. iCub uses YARP extensively as robotic protocol for internal data events. So, DALI agents can be developed to asynchronously receive data events from iCub sensors and send outcomes of logical decisions/actions through YARP ports. Ports have to be accurately selected in order to work at the highest possible level of abstraction, where logic programming and reasoning capabilities of DALI agents are more appropriate. Lower level ports should be controlled by conventional cybernetic controllers, in a hierarchical control structure where loop speed is higher closer to the hardware (or virtual hardware in case of a simulator).

ServerDALI. The DALI cloud solution is encapsulated in "docker" container, that[2] includes everything needed to run the code in a platform-independent way. Composed together with the iCub YARP docker container, a cloud computing based MAS could control the cognitive aspect of an embodied robot. The ServerDALI application allows a DALI MAS to be made available to users also via web or mobile applications. ServerDALI and a sample web interface have been programmed using PHP, CSS3, Javascript and HTML5. The entire MAS is made available analogously to a single object, so its external users are not required to possess any notion about Agents or Artificial Intelligence. This is accomplished via a special agent (called Ermes) which is added to any MAS and acts as an interface between the MAS and the external web based environment; in particular, via the ServerProlog library PHP and JSON objects can be translated into messages that Ermes can then dispatch, and vice versa. This solution can be generalized to other agent-oriented frameworks and to different external languages.

[2] *Docker* is an open-source multi-platform tool to automate the deployment of Linux lightweight containers, see http://www.docker.com/technologies/overview.

5 Conclusions

In this paper we have showed the potential usefulness of the DALI logical agent-oriented programming language in the cognitive robotic domain; we particularly envisage applications for user monitoring and training concerning elderly or disabled persons, or children (in cooperation with parents or caregivers). We have then illustrated in some detail the extensions to the previously-existing DALI implementation which allow DALI agents to be actually exploited in the robotic realm. Therefore, DALI agents can now be developed to act as high level cognitive robotic controllers, and can be automatically integrated with conventional embedded controllers. The cloud package ServerDALI allows a DALI MAS to be integrated in any practical environment. Realistic experiments are planned in the near future in the context of the F&K project.

References

1. Aielli, F., Ancona, D., Caianiello, P., Costantini, S., De Gasperis, G., Di Marco, A., Ferrando, A., Mascardi, V.: Friendly&Kind with your health: human-friendly knowledge-intensive dynamic systems for the e-health domain. In: Bajo, J., et al. (eds.) PAAMS 2016. CCIS, vol. 616, pp. 15–26. Springer, Heidelberg (2016)
2. van Kemenade, M., Konijn, E.A., Hoorn, J.F.: Robots humanize care - moral concerns versus witnessed benefits for the elderly. In: Verdier, C., Bienkiewicz, M., Fred, A.L.N., Gamboa, H., Elias, D. (eds.) Proceedings of HEALTHINF 2015, pp. 648–653. SciTePress (2015)
3. Paauwe, R.A., Keyson, D.V., Hoorn, J.F., Konijn, E.A.: Minimal requirements of realism in social robots: designing for patients with acquired brain injury. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 2139–2144. ACM (2015)
4. Fisher, M., Bordini, R.H., Hirsch, B., Torroni, P.: Computational logics and agents: a road map of current technologies and future trends. Comput. Int. J. **23**(1), 61–91 (2007)
5. Bordini, R.H., Braubach, L., Dastani, M., ElSeghrouchni, A.F., Gomez-Sanz, J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica (Slovenia) **30**(1), 33–44 (2006)
6. d'Inverno, M., Fisher, M., Lomuscio, A., Luck, M., de Rijke, M., Ryan, M., Wooldridge, M.: Formalisms for multi-agent systems. Knowl. Eng. Rev. **12**(3), 315–321 (1997)
7. Costantini, S., De Gasperis, G.: Memory, experience and adaptation in logical agents. In: Casillas, J., Martínez-López, F.J., Vicari, R., De la Prieta, F. (eds.) Management Intelligent Systems. AISC, vol. 220, pp. 17–24. Springer, Heidelberg (2013)
8. Costantini, S., Dell'Acqua, P., Pereira, L.M.: Conditional learning of rules and plans by knowledge exchange in logical agents. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011. LNCS, vol. 6826, pp. 250–265. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22546-8_20
9. Costantini, S.: ACE: a flexible environment for complex event processing in logical agents. In: Baldoni, M., Baresi, L., Dastani, M. (eds.) EMAS 2015. LNCS, vol. 9318, pp. 70–91. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26184-3_5

10. Costantini, S., De Gasperis, G.: Runtime self-checking via temporal (meta-)axioms for assurance of logical agent systems. In: Proceedings of LAMAS 2014, 7th Workshop on Logical Aspects of Multi-agent Systems, held at AAMAS 2014, pp. 241–255 (2014)
11. Costantini, S., Tocchio, A.: A logic programming language for multi-agent systems. In: Flesca, S., Greco, S., Ianni, G., Leone, N. (eds.) JELIA 2002. LNCS, vol. 2424, pp. 1–13. Springer, Heidelberg (2002). doi:10.1007/3-540-45757-7_1
12. Costantini, S., Tocchio, A.: The DALI logic programming agent-oriented language. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS, vol. 3229, pp. 685–688. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30227-8_57
13. Costantini, S.: The DALI agent-oriented logic programming language: summary and references 2016 (2016). http://www.di.univaq.it/stefcost/info.htm
14. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, New York (2003)
15. Costantini, S., Dell'Acqua, P., Pereira, L.M., Toni, F.: Towards a model of evolving agents for ambient intelligence. In: Proceedings of the Symposium on Artificial Societies for Ambient Intelligence (ASAmI 2007) (2007)
16. Costantini, S.: Self-checking logical agents. In: 8th Latin American Works, LANMR 2012. CEUR Workshop Proceedings, vol. 911. CEUR-WS.org (2012). 3–30 Invited Paper, Extended Abstract in Proceedings of AAMAS 2013
17. Costantini, S., Formisano, A.: Modeling preferences and conditional preferences on resource consumption and production in ASP. J. Alg. Cogn. Inf. Logic **64**(1), 3–15 (2009)
18. De Gasperis, G., Costantini, S., Nazzicone, G.: DALI multi agent systems framework, July 2016. http://github.com/AAAI-DISIM-UnivAQ/DALI
19. Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., Von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., et al.: The iCub humanoid robot: an open-systems platform for research in cognitive development. Neural Netw. **23**(8), 1125–1134 (2010)