Submitted to Cognitive Computation manuscript No.
(will be inserted by the editor)

Perceptions or actions? Grounding how agents interact within a software architecture for cognitive robotics

Rebeca Marfil · Adrián Romero-Garcés · Juan P. Bandera · Luis J. Manso · Luis V. Calderita · Pablo Bustos · Antonio Bandera · Javier García-Polo · Fernando Fernández · Dimitri Voilmy

Received: date / Accepted: date

Abstract One of the aims of cognitive robotics is to endow robots with the ability to plan solutions for complex goals and then to enact those plans. Additionally, robots should react properly upon encountering unexpected changes in their environment that are not part of their planned course of actions. This requires a close coupling between deliberative and reactive control flows. From the perspective of robotics, this coupling generally entails a tightly integrated perceptuomotor system, which is then loosely connected to some specific form of deliberative system such as a planner. From the high-level perspective of automated planning, the emphasis is on a highly functional system that, taken to its extreme, calls perceptual and motor modules as services when required.

This paper proposes to join the perceptual and acting perspectives via a unique representation where the responses of all software modules in the architecture are generalized using the same set of tokens. The proposed representation integrates symbolic and metric information.

The proposed approach has been successfully tested in CLARC, a robot that performs Comprehensive Geriatric Assessments of elderly patients. The robot was favourably appraised in a survey conducted to assess its behaviour. For instance,

R. Marfil, A. Romero-Garcés, J.P. Bandera, A. Bandera
University of Málaga, Spain
E-mail: {rebeca, argarces, jpbandera, ajbandera}@uma.es

L.V. Calderita, P. Bustos
University of Extremadura, Spain
E-mail: {lvcalderita, pbustos}@unex.es

L.J. Manso
Engineering and Applied Science School, Aston University, UK
E-mail: l.manso@aston.ac.uk

J. García-Polo, F. Fernández
University Carlos III Madrid, Spain
E-mail: {fjgpolo,ffernand}@inf.uc3m.es

D. Voilmy
University of Technology of Troyes, France
E-mail: dimitri.voilmy@utt.fr

using a 5-point Likert scale from 1 (strongly disagree) to 5 (strongly agree), patients reported an average of 4.86 when asked if they felt confident during the interaction with the robot.

This paper proposes a mechanism for bringing the perceptual and acting perspectives closer within a distributed robotics architecture. The idea is built on top of the blackboard model and scene graphs. The modules in our proposal communicate using a short-term memory, writing the perceptual information they need to share with other agents and accessing the information they need for determining the next goals to address.

Keywords Cognitive robotics · Automatic planning · software architectures

1 Introduction

Abstract reasoning about concrete phenomena is intimately tied with the existence of an internal representation of this reality. From a robotics perspective, this implies the establishment and maintenance of a connection between what the robot reasons about and what it can sense [1]. At the core of this connection resides the *symbol binding problem*, which is deeply connected to the more general question of meaning. It directly points at the problem of creating and keeping a bond between a concrete object and a symbol. This problem is complex due to the high dimensionality of the spaces required to represent real world entities. Moreover, there are many parameters that can affect the internalization process. For example, the same entity may have different representations according to multiple conditions, not only related to external factors such as illumination, motion or the dynamic essence of the outer reality, but also related to internal issues such as the specific properties of the sensor or the internal parameters of the observer. On the other hand, given the importance of actions for Robotics, binding references and referents are usually enhanced by the additional matching of entities with goals and affordances [2]. As Marvin L. Minsky noted in 1986 [3]: *we need to combine at least two different kinds of descriptions. On one side, we need structural descriptions for recognizing chairs when we see them and, on the other side, we need functional descriptions in order to know what we can do with chairs.* The concept of object-action complexes [4] postulates a human-like description by which an entity is identified considering (a) its inherent features and (b) the actions that can be performed with it. This augments the internal representation of the world by including what the robot *wants* to achieve, and the opportunities afforded by a situation to achieve those objectives. New tokens -symbols- should then be employed to reflect the perceptual and acting views describing the same reality [5].

The symbol binding problem has been approached from very different points of view and by many researchers in recent decades (see some examples in the brief survey by Coradeshi et al. [1]). Among these proposals, recent contributions [6] [40] point towards the use of a shared, highly structured, internal representation. Contrary to purely symbolic representations, this shared memory is fed with the symbolic and numeric tokens that are collaboratively generated by all the software components participating in the control architecture. The power of this idea lies in the use of a graph as the supporting data structure whose growth is controlled by a generative grammar. Perceptive and action modules can edit the graph according to the rules of the grammar and thus, maintain a formally correct, rich representation of the robot itself and of the environment. With this scheme, symbols can be grounded not only using a combination of local features but also considering the context provided by other modules as modifications to the graph.

When looking into actions instead of perceptions, new interesting issues arise. The shared-memory idea can be extended into a deliberative architecture composed of modules -some of which can reason using high-level domain knowledge. One of these agents, the task planner, computes solutions to missions that are executed by the other modules sharing the graph. These same modules must nevertheless autonomously react to unforeseen changes and update the internal model accordingly. In this architecture there must be a balanced relationship between the top-down and bottom-up flows of control. The initial version of this model was proposed in [40]. In that work, the coordination of the agents executing a plan was assigned to a special agent called the Executive. In classical three-layer architectures, this role resembles the one in the Sequencer module [30] or the *goal manager* in the original Cosy Architecture Schema [7]. This agent sets a direct pipeline with the participating agents to determine the current goal and the behaviour to be executed to achieve that goal. This mechanism alleviates the internal, shared model from annotating actions and the agents from reasoning about those actions.

Significantly, this scheme implies that the modules will execute the required sub-task as they receive a direct command. Hence, the internalized state of the world does not guide its behaviour and it will be *only* used, as is typical in blackboard models, to share information about goals or partial results among the agents. This paper proposes to change this scheme by removing this *goal manager* and forcing all the modules in the architecture to encode perceptions and actions using the same set of tokens. Briefly, as it occurs with perceptions, actions will also be thought of as changes to the inner world. It is in this new proposal where the shared representation truly becomes the core of the architecture, storing all the data that is required for the software modules to perform their activities. This simplifies the architecture as a whole, as no further modules are required to take the responsibility of understanding the whole state of the robot and its context. Furthermore, this simplification eases intercommunication and generalization. The evolution of the representation with time will provide a complete, semantic description of the robot's activities, opening new doors to learning from previous experiences [9] or agency [10]. One major disadvantage, however, is that this scheme forces task-dependent modules to use a more complex logic to infer their activities from the state (affordances), as they will not receive specific action commands. However, they are also more easily modified, added or removed without affecting the rest of the architecture. This approach also eases the deployment of different behavioural schemes, such as stigmergic collaboration or competitive approaches (e.g. using more than one planner).

1.1 Relevant novelty with respect to our previous work

The use of a unique, shared representation including symbolic and metric information was already implemented with remarkable success in previous works [25–27]. These initial efforts introduced the so-called Deep State Representation (DSR) which will be described in Section 3. The relevant novelty of this contribution with respect to previous works is the elimination of the Executive agent, which imposes a more centralized flow of control by a distributed policy of annotating intentions and actions in the DSR in real-time. This improvement allows agents, for instance, to know that the **robot** is_interacting with a **person**, but also that it is_not **speaking** when that is the case. Although in our previous approach agents had limited access to what the robot was doing and with who, in the current approach all the action space could be read from the shared representation. The improvement introduced here is a crucial element to facilitate the reactive, fast response of all the agents in the architecture, for example allowing a Dialog agent to quickly be aware of attentive or physical changes in the human interlocutor. As a field validation of the complete CORTEX architecture and of the changes introduced in this paper, a series of experiments with the CLARC robot have been performed in real-world health care scenarios.

1.2 Organization of the paper

The rest of the paper is organized as follows. Section 2 introduces the use of shared representations on cognitive architectures for robotics. Section 3 briefly presents how concrete and symbolic information is included in the representation used in this work, including actions and perceptions. The proposal has been implemented in CLARC[1], a robot in charge of performing different comprehensive tests on geriatric patients. CLARC has been developed within one of the Public end-user Driven Technological Innovation (PDTI) programmes of the ECHORD++ project, which offers the research consortia funding to develop robotics technologies for real use-cases as Comprehensive Geriatric Assessments (CGA). Sections 4 and 5 show the unfolding of the proposal for performing a CGA test and practical results obtained from this work. An open discussion about the advantages or inherent problems of using the proposed scheme is sketched in Section 6. Conclusions and future work are outlined in Section 7.

2 Related work

The existence of explicit representation mechanisms is common to most cognitive architectures, which use a *working memory* for driving attention, reasoning and learning. The idea can be traced back to blackboard models [11] and architectures such as SOAR [12] and ACT-R [13], which emphasized the existence of this shared workspace. The blackboard model basically proposes a structure consisting of independent agents, which work on their own *portion* of the problem and manage their own data; a control module, which is responsible for choosing the agent/s whose response/s must be executed; and the blackboard, a common solution space where all agents write their partial results. Thus, the blackboard is a shared repository of problems, objectives and partial outcomes, which can be accessed and updated by the agents. For instance, in the AIS architecture [14], the blackboard scheme is used as an implementation of the cognitive layer with a central working memory, where all behaviours write perceptual inputs or previously executed behaviours,

[1] http://echord.eu/essential_grid/clark/

and to which all behaviours are connected to be triggered when certain conditions are fulfilled. The CoSy Architecture Schema (CAS) [7] employs a communication paradigm based on shared memories. Similarly, the Mala architecture uses different blackboards to provide storage and serve as a communications mechanism [15]. In the CRAM/KnowRob architecture, an active hub is employed for pro–actively querying perceptual agents to acquire and store knowledge within a central base [17]. In the architecture described by Lemaignan et al. [16], knowledge is centrally managed in an active semantic blackboard (the ORO server). Another usual mechanism for sharing information is the global workspace (GW) model. Here, several parallel agents compete and cooperate to modify a global workspace. Agents are responsible for task-dependent functionalities, such as visual perception, path planning, or speech understanding. When an agent modifies the global workspace, the information this agent offers is broadcast back to the rest of agents. The means by which access is granted to the common workspace is an attention mechanism. CERA is an architecture structured in layers that uses the blackboard-based model provided by CRANIUM [18]. Within CRANIUM, global workspace dynamics are modelled as an information processing system, whose input is the raw sensory data and whose output is a stream of artificial *qualia* (integrated multimodal representations). In the ARCADIA architecture [19], low-level processes are encapsulated in modules called *components*. Each component communicates its content and the focus of attention through *interlingua*, a repository of structured elements organized by topic. In CHARISMA [20], the relatively small chunk of information that is currently deemed most important is broadcast to a set of processes that work together to extract information, make associations, decompose problems, etc.

Complementary to the scheme proposed to manage this shared memory, the aforementioned proposals must determine how to represent the stored knowledge. CHARISMA organizes the representation as a semantic hyper network (SHYNE). SHYNE is composed of nodes (basic and complex) and links characterized by unique identifiers. These identifiers are passed between processes. The CAST model represents knowledge as a diffuse, pervasive resource [21]. In the ORO server [16], knowledge is represented as RDF (Resource Description Framework) triples (subject, predicate and object) in the OWL (description logics) sub-language. RDF triples imply the setting of binary predicates, an issue that constraints the expressiveness of the representation. To solve this problem, KnowRob interleaves RDF with Prolog [22]. Answer Set Programming (ASP) is a non-monotonic logic programming paradigm that is well-suited for representing and reasoning with common sense knowledge [23]. These proposals usually manage high-level, symbolic concepts. For instance, the ORO server is connected to deliberative agents [16]. When required, concrete knowledge is managed within dedicated agents. However, this information is not represented in the knowledge base. Samuel Wintermute and John E. Laird noted the problems that can arise when spatial abstraction does not capture the details necessary for correct internal reasoning [24]. To manage concrete and symbolic representations within an unified framework, the CORTEX architecture employs a central knowledge base that stores all information coming from the agents within an annotated graph [25–27]. Figure 1(left) shows how two different agents interact through this representation to unfold a 'go to person' behaviour provided by the deliberative agent (in this specific case, the PELEA module [28]). Domain-dependent agents, in charge of solving the necessary perceptual and acting tasks, use this plan and the shared world model to perform their activities. In this naive case, the `Person` agent detects the pose of the person and provides these data to the representation, and the `Navigation` agent takes these data and moves the robot. Contrary to other approaches such as KnowRob [17], where symbolic facts are evaluated when needed, the shared representation is cooperatively built and updated by all the agents. These agents are continuously running in the background, computing and asserting the knowledge representation. In this sense, our architecture works like the one proposed by Lemaignan [16].

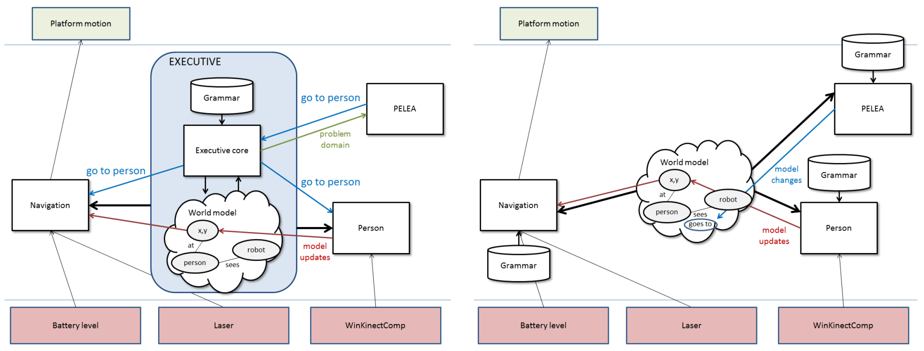


Fig. 1 (left) A brief scheme of a possible instantiation of the CORTEX architecture, showing its three main components: `PELEA` (a high-level module for planning, monitoring, and learning), an `Executive` in charge of redirecting the plans from the planning module to the corresponding low-level modules and managing the representation of the world, and a set of domain-dependent agents (in this case represented by `Navigation` and `Person`). Modules with a red background (Battery level, Laser, etc.) provide inputs to the agents, while those with a green background (Platform motion) receive the results from the agents. Both sets constitute the Hardware Abstraction Layer (HAL) of the system. (right) Our proposed scheme for dealing with this same task. There is no `Executive` agent; instead, `PELEA` is responsible for changing the representation to achieve the correct response from the `Person` and `Navigation` agents (see text).

As Figure 1(left) depicts, the execution of the high-level action originating from PELEA is controlled by the `Executive` agent, a component that also provides the interface for the representation. The `Executive` interfaces with `PELEA`, from which it receives the plan to execute and to which it reports changes in the representation through asynchronous events. The `Executive` publishes the representation to all agents (blank arrows in Figure 1(left)), and it is also the only module in charge of checking whether the changes in the representation coming from the agents are valid or not. More details can be read in [8]. The scheme proposed in this work is depicted in Figure 1(right). The idea is that all agents in the architecture, deliberative or reactive, annotate the internal representation not only with perceptual information but also with acting information. Thus, the current action is not directed from a specific manager to the rest of agents in the architecture, alleviating the interaction among agents and reducing the response time to any detected event. On the contrary, this forces the representation to include information about the running tasks (e.g., **robot** goes_to **x,y**), as the rest of agents must be informed through the shared representation. The agents must now have some specific knowledge about the domain. The example shows an additional objective of our design:

by leaving out the lowest-level representations, we have tried to make this annotation as close as possible to the one used by humans. In a human-robot interaction context, this will allow the effective support of human-robot communication.

Finally, Table 1 summarizes a qualitative comparison of the CORTEX architecture with respect to some of the architectures described in this Section. The relevant features of this comparison are a subset of those proposed in the deep analysis addressed by I. Kotseruba et al. (2016) [39]. Four topics are considered: sensory modalities, action selection, memory and learning. The first topic is essential to the development of a cognitive robot, as it should need to have channels to perceive the outer world and internalize this information. In the Table, eight options are considered. Smell, audition, touch, proprioception and vision refer to typical sensory modalities. Vision is present in all architectures, and the rest could be surely added if needed. Symbolic inputs include inputs in the form of text commands and are not considered in most hybrid architectures. Crucially, CORTEX is able to consider these inputs if they come from a Dialogue agent. The Multimodal option refers to the ability to perform feature extraction from several sensors in an independent and concurrent manner. The second topic refers to how the course of action is determined. CORTEX merges reactive mechanisms of actuation with a planning framework that determines the sequence of actions for reaching a goal. As these are the typical methods in most hybrid approaches, other solutions consider a set of alternatives and then use a selection method (winner-take-all (WTA), probabilistic, predefined) and selection criteria (relevance, utility, emotion). The choice is the best action according to these criteria (i.e., action with the highest activation value). The memory topic refers to the systems implemented in most architectures for managing the intermediate results of computations. It is difficult to address cognition without an internal representation. Just as the other architectures in the table, CORTEX manages a working memory where the information related to the current task is available. As shown in Section 3, CORTEX also stores information about objects and the relationships among them (semantic memory) and a long-term procedural memory of implicit knowledge (the "Grammar" boxes in Figure 1). Contrary to the rest of the approaches in this qualitative comparison, CORTEX manages all of these representations within a unified structure (a global memory), where both sensory data and high-level symbols describing the state of the robot and the environment are stored. Finally, the learning topic determines the ability of an architecture to improve over time. It is one of the most active topics we are working on to improve CORTEX, as our architecture currently only implements procedural learning (see Section 4.2).

3 The Deep State Representation

The Deep State Representation (DSR) is a multi-labelled directed graph that holds symbolic and geometric information within the same structure. Symbolic tokens are defined as logic attributes related by predicates that are stored within the graph in nodes and edges. Geometric information is stored as predefined object types linked by 4×4 homogeneous matrices. This information is also stored as nodes and edges of the graph. Figure 2 shows a reduced view of the state associated with

Table 1 Qualitative comparison of several cognitive architectures [39]

		CORTEX	[7]	[18]	[19]	[20]	[21]
sensory modalities	Multimodal	X	X	X			
	Smell						
	Audition	X	X				X
	Touch			X			
	Proprioception	X		X		X	
	Symbolic input	X					
	Vision	X	X	X	X	X	X
	Other senses	X	X	X			
action selection	Planning	X	X			X	X
	WTA						
	Probabilistic						
	Predefined				X		
	Relevance				X		X
	Utility			X		X	
	Internal factors			X		X	
	Reactive	X	X	X		X	X
memory	Sensory				X	X	
	Working Memory	X	X	X	X	X	X
	Semantic	X	X	X		X	X
	Procedural	X	X	X		X	X
	Episodic					X	X
	Global	X					
learning	Declarative		X			X	
	Perceptual		X				
	Procedural	X					X
	Associative		X			X	
	Nonassociative						
	Priming		X				

the execution of a test (see Section 4). The **person** and **robot** nodes are geometrical entities, both linked to the **world** node (a specific anchor providing the origin of coordinates) by a rigid transformation. However, at the same time that we can compute the geometrical relationship between both nodes ($RT^{-1} \times RT'$), the **person** can be located (is_with) close to the **robot**. Furthermore, an agent can annotate that the **robot** is_not **speaking**.

3.1 Data structure

As a hybrid representation that stores information at both geometric and symbolic levels, the nodes of the DSR store concepts that can be symbolic, geometric or a combination of the two. Metric concepts describe numeric quantities of objects in the world, which can be structures such as a three-dimensional mesh, scalars such as the mass of a link, or lists such as revision dates. Edges represent relationships between nodes. Two nodes may have several kinds of relationships but only one of them can be geometric. The geometric relationship is expressed with a fixed label called RT. This label stores the transformation matrix (expressed as a rotation-translation) between them.

The DSR can be described as the union of two *quivers*: one associated with the symbolic part of the representation, $\Gamma_s = (V, E_s, s_s, r_s)$, and the other related to the geometric part, $\Gamma_g = (V_g, E_g, s_g, r_g)$. A quiver is a quadruple, consisting of a

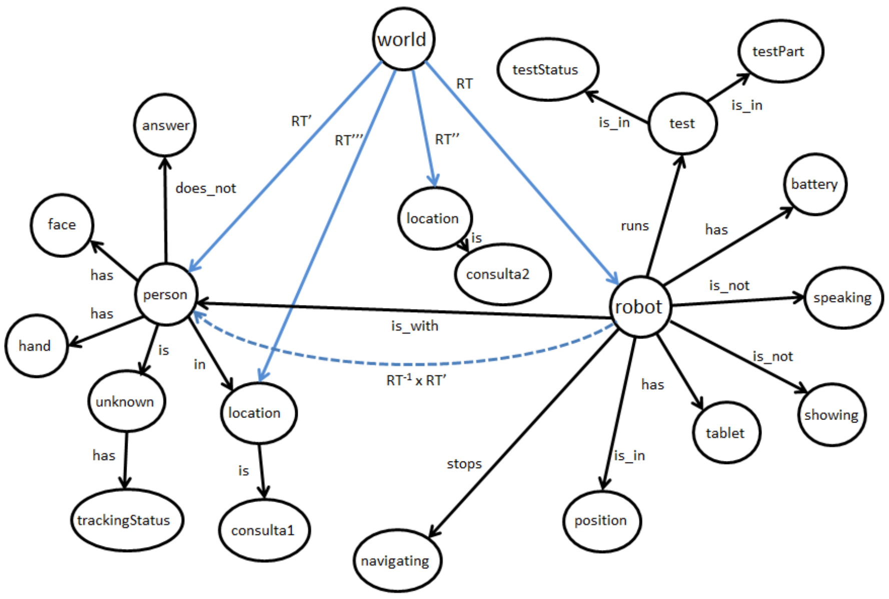


Fig. 2 Unified representation as a multi-labelled directed graph. For instance, edges labelled as is_with or is_not denote logic predicates between nodes and they belong to $\mathit{\Gamma}_s$. On the other hand,edges starting at **world** and ending at **person** and **robot** are geometric and they encode a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems (see text).

set V of nodes, a set E of edges, and two maps $s, r : E \rightarrow V$. These maps associate each edge $e \in E$ with its starting node $\mathbf{u} = s(e)$ and ending node $\mathbf{v} = r(e)$. Sometimes we denote an edge by $e = \mathbf{uv} : \mathbf{u} \rightarrow \mathbf{v}$ with $\mathbf{u} = s(e)$ and $\mathbf{v} = r(e)$. Within the DSR, both quivers are finite, as both sets of nodes and edges are finite sets. A *path* of length m is a finite sequence $\{e_1, ...e_m\}$ of edges such that $r(e_k) = s(e_{k+1})$ for $k = 1...m-1$. A path of length $m \geq 1$ is called a *cycle* if $s(e_1)$ and $r(e_m)$ are identical.

The properties of the symbolic quiver $\mathit{\Gamma}_s$ can be stated as follows:

1. The set of symbolic nodes V contains the geometric set V_g (i.e. $V_g \in V$)
2. Within $\mathit{\Gamma}_s$ there are no cycles of length one. That is, there are no *loops*
3. Given a symbolic edge $e = \mathbf{uv} \in E_s$, we cannot infer its inverse $e^{-1} = \mathbf{vu}$
4. The symbolic edge $e = \mathbf{uv}$ can store multiple values

Similarly, given the properties of the transformation matrix RT, the characteristics of the geometric quiver $\mathit{\Gamma}_g$ can be stated as follows:

1. Within $\mathit{\Gamma}_g$ there are no cycles (acyclic quiver)
2. For each pair of geometric nodes $\mathbf{u}$ and $\mathbf{v}$, the geometric edge $e = \mathbf{uv} \in E_g$ is unique
3. Any two nodes $\mathbf{u}, \mathbf{v} \in V_g$ can be connected by a unique simple path
4. For each geometric edge $e = \mathbf{uv} = RT$, we can define the inverse of e as $e^{-1} = \mathbf{vu} = RT^{-1}$

Thus, the quiver $\mathit{\Gamma}_g$ defines a directed rooted tree or rooted tree quiver [31]. The *kinematic chain* $C(\mathbf{u}, \mathbf{v})$ is defined as the path between the nodes $\mathbf{u}$ and $\mathbf{v}$. The equivalent transformation matrix RT of $C(\mathbf{u},\mathbf{v})$ can be computed by multiplying all RT transformations associated with the edges on the paths from nodes $\mathbf{u}$ and $\mathbf{v}$ to their closest common ancestor $\mathbf{w}$. Note that the values from $\mathbf{u}$ to the common ancestor $\mathbf{w}$ will be obtained by multiplying the inverse transformations. One example of computing a kinematic chain is shown in Figure 2.

3.2 Internalizing the outer world within the DSR

The complexity of the domain-dependent modules typically implies that they will be internally organized as networks of software components, or (*compoNets*). Within each compoNet, the connection with the DSR is achieved through a specific component, the so-called *agent*. These agents are also present in previous architectures, such as CAS or RoboCog. In fact, they are needed in any blackboard-based scheme. However, the degree of complexity significantly changes when we move from these schemes to the proposed one. With the removal of the *goal managers*, the agents in our architecture need to search for those changes in the DSR that launch the specific problem-solving skills (actions) of the compoNets they represent (e.g. detecting a person's pose, or starting to say a sentence). This mapping between subgraphs in the DSR and actions constitutes the knowledge of each agent, which is intimately linked to our definition of affordances. The internal data flow of these agents is briefly outlined in Algorithm 1.

```
subscribe to DSR updates;
while (1) do
    search_for_changes { output: action };
    process { action };
    if DSR_changes then
        update DSR;
    end
end
```

Algorithm 1: Procedure of an agent in our proposal

The **search_for_changes** skill depends on each agent and the behaviours that the compoNet can solve. Within the algorithm, it is stated that this function returns the *action* to perform. This is the most significant difference between other blackboard-based approaches and our proposal: in other approaches, the action is imposed by an external module, but in our approach, the action is determined by the agent. As we will briefly discuss in Section 6, this opens new ways for dealing with the top-down and bottom-up mechanisms that determine what the next action to perform will be or that implement reflexive behaviours. The whole execution of the compoNet is conditioned by the rules of its internal grammar, i.e., triplets with the states of the DSR after, during and before the compoNet executes a specific action. Figure 3 shows one example stored in the `Speech` agent (see Section 4). Figure 3(left) shows the state of the DSR before the deliberative planner indicates that the robot should say the sentence yyy. When the planner changes the DSR (changing the attribute xxx to yyy between **test** and **test_part**), and the agent `Speech` receives the new state, `Speech` uploads the DSR to inform all agents that the **robot** is **speaking**. When the sentence ends, `Speech` changes the

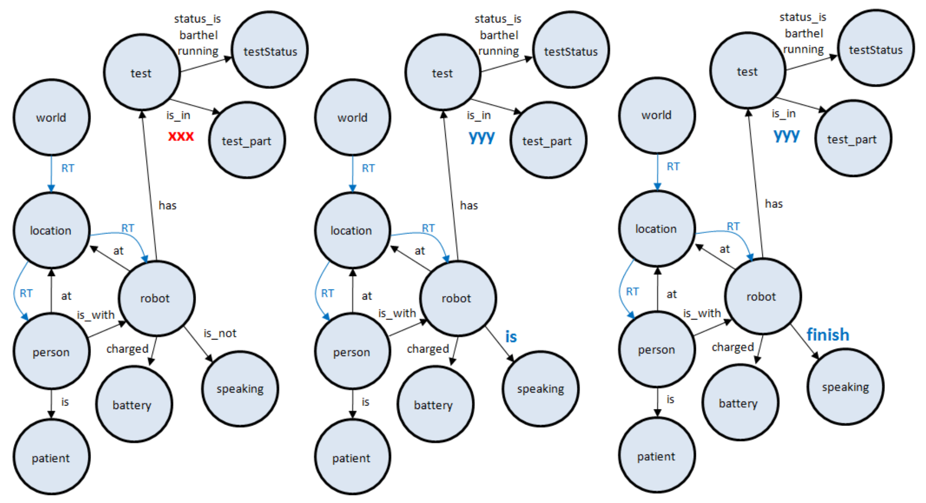


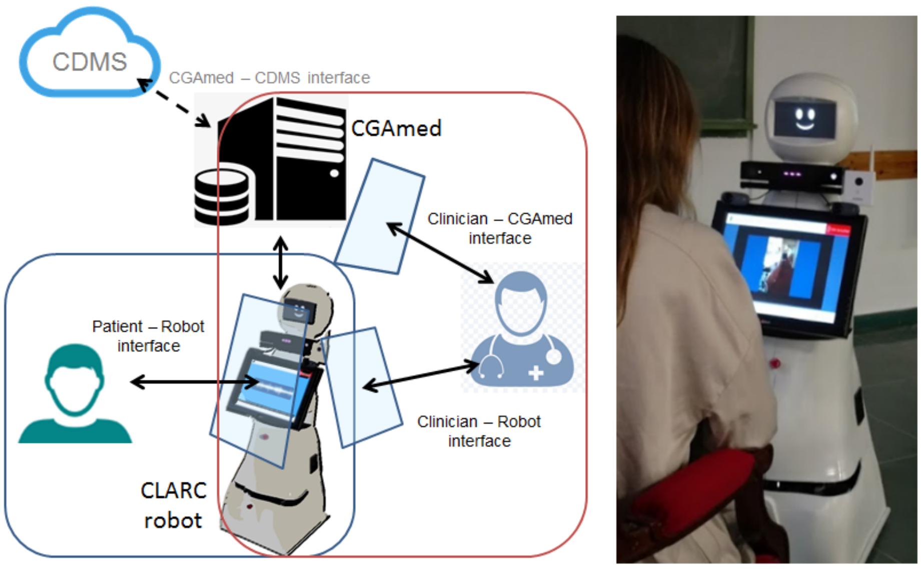
Fig. 3 (left) The state of the DSR before PELEA states that the robot should say a sentence yyy; (center) PELEA changes the text to speech (from xxx to yyy) and then, when the Speech agent reads the DSR, the robot starts to speak (**robot** is **speaking**). (right) The sentence has been said and the Speech agent informs the rest of the agents through the DSR (**robot** finish **speaking**).

DSR to indicate that the **robot** finish **speaking**. It is important to note how our agents must inform the rest of the architecture that they *are executing* an action. Initially, this constitutes the current way to prevent other agents in the architecture from modifying that part of the DSR while an action is being executed. However, a sudden change in the outer world could force an agent to propose speaking a new sentence or could even stop the execution of the `Speech` agent. To achieve this, it is sufficient to update the sentence yyy or to change it to a *null* sentence. This assures that the architecture reacts fast enough to change the goals [7]. This also hides a problem: the agents are responsible for not unnecessarily disturbing a course of action. Their inherent grammar rules must be carefully designed.

4 Building a whole architecture around the DSR

4.1 The CLARC project

CLARC is waiting in Room 1 for its first patient. When Dr. Cesar presses the Start button on his mobile phone, CLARC wakes up and looks for Carlos, his patient, who should be sitting in front of it. When CLARC sees him, it greets him and presents itself as the one responsible for conducting a small test, which will help the doctor to know how he is. It also briefly describes to him what the test will be: a collection of questions that must be answered by selecting one of 3-4 options described. Then, the test starts. Sometimes, CLARC hears words that it does not understand. Sometimes, it just hears nothing. However, these situations are detected... and there are planned solutions for solving them. CLARC is patient and can repeat


CDMS
CGAmed – CDMS interface
CGAmed
Clinician – CGAmed interface
Patient – Robot interface
Clinician – Robot interface
CLARC robot
Fig. 4 (left) Global overview of the CLARC framework, and (right) the CLARC robot

the phrase several times, suggests leaving it and going to the next question, and always offers the option to answer using the touch screen on its chest. After 10-15 minutes, the test ends. It is time to say goodbye to Carlos and to send an internal message to Dr. Cesar indicating that the result of the test is stored on the CGAmed server for validation.

This brief summary describes how the CLARC robot should administer a CGA test, in this case the Barthel one. To perform the required actions, the robot uses specific sensors and is endowed with a software framework that allows it to manage its own hardware and task-dependent functionalities, as well as the vast amount of data (videos included) associated with a session. This last point is essential, as the robot is a tool whose behaviour during the test should be subsequently analysed by a medical expert. It is then necessary to collect all the information required by the doctor for review and to end the evaluation. Figure 4 depicts a global overview of the CLARC solution. It has two major parts: the robot and the CGAmed server. The server stores the database with information about patients, doctors, rooms and the schedule of sessions designed by the clinicians. It also stores the application that Dr. Cesar used to launch the test. Finally, the CGAmed is responsible for connecting the whole CLARC framework with the Clinical Data Management System (CDMS).

Figure 4 also shows the presence of three human-computer interfaces in the architecture. One of them allows the medical expert to supervise the data stored in the CGAmed server. For instance, it can be used to check and edit the scores assigned by the robot to a CGA test, to watch the video associated with the session, or to compare two administrations of the same test performed at different times. The clinician can also use a second interface to connect with the robot. Among other functionalities, this interface allows the doctor to watch the session online or to pause the test. However, in this proposal, the most relevant interface

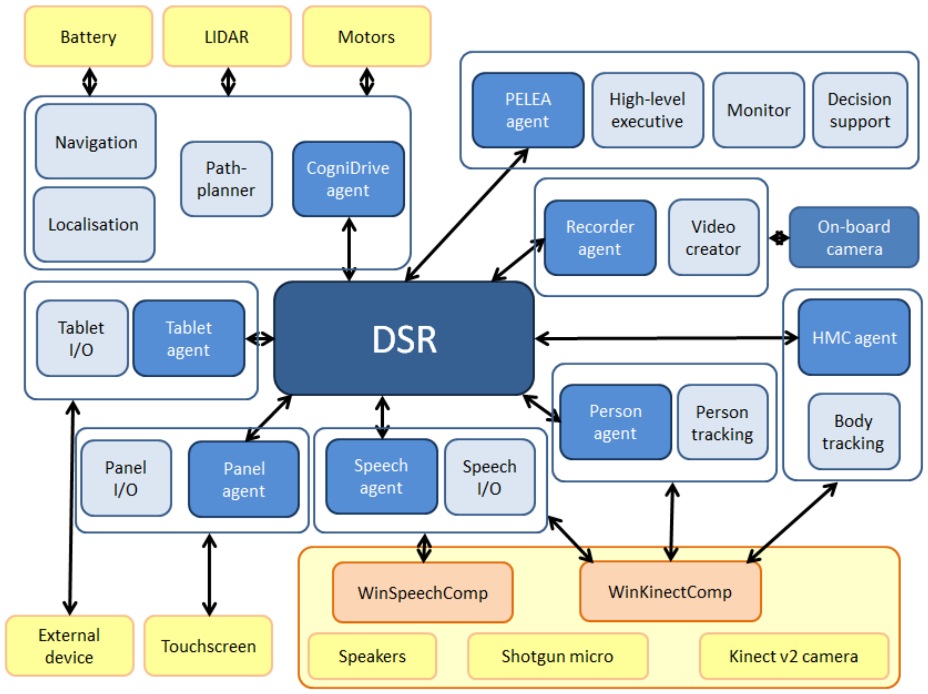


Fig. 5 Overview of the software architecture within the CLARC robot.

is the patient-robot interface. This interface is fully managed by the robot and, therefore, is linked to the Deep State Representation that drives its behaviour. The next sections provide a detailed description of the software architecture in the robot and the agents involved in the patient-robot interface.

4.2 Overview of the architecture

Figure 5 shows an overview of the whole architecture in charge of administering the CGA tests within the CLARC project. There are eight agents surrounding the inner world provided by the DSR: `PELEA`, `Speech`, `Panel`, `Tablet`, `CogniDrive`, `Recorder`, `HMC` and `Person`. The `PELEA` agent is in charge of providing the deliberative skills to the architecture. It is an instantiation of the Planning, Learning and Execution Architecture (PELEA) [28], which maintains its own internal memory and the software modules for monitoring the course of action. It interacts with the rest of agents using the same procedure, changing the DSR. The `Recorder` agent manages an IP camera, which provides a stream of video for online supervision and also records the session for offline visualization.

The channels for patient-robot interaction are provided by the `Speech`, `Panel` and `Tablet` agents. The `Speech` agent is in charge of interpreting the answers of the patient or guardian and translating text into speech, generating the voice of CLARC. It is internally connected to `WinSpeechComp`, a module in charge of generating the voice from text using the Text-To-Speech (TTS) software provided

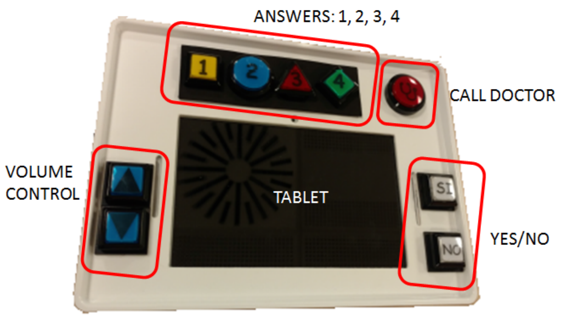


Fig. 6 Augmented tablet used for interacting with the robot.

by Microsoft Speech Platform SDK. This software is also used for voice recognition with the help of specific grammars that are loaded for each question in order to maximize recognition rates. The transcription system of the ASR system achieved a 5.1% error rate, the same rate measured for humans [32]. Furthermore, the use of the Microsoft SDK made the implementation of a multi-language interface easier. The robot is currently able to interact with the patient in French, English or Spanish. The verbal channel is enhanced by using a touchscreen on the torso of the robot [27]. The `Panel` agent manages the tactile interaction and the information shown in this touchscreen, which has been carefully designed for dealing with elderly people [34,33]. The position of the touchscreen on the robot is shown in Figure 4(right). Intense use of this quasi-vertical touchscreen forces the patient to adopt an uncomfortable position, not only because the patient must keep his/her arm extended but also because the robot is prevented from being too close to the patient, resulting in the patient having to continuously approach/move away from the screen to touch it. To avoid this discomfort, we add a third element to the interface: an external device whose core is a tablet with large buttons used to answer typical questions in a Barthel test (see Figure 6).

The CLARC robot is built over the MetraLabs SCITOS G3 platform[2]. This base uses a LIDAR sensor for localization, navigation and obstacle avoidance, functionalities that are provided by the CogniDrive software running over MIRA middleware[3]. The `CogniDrive` agent implements a bridge to connect these software modules to our cognitive architecture.

Finally, the `Person` and `HMC` agents are in charge of detecting, tracking and capturing the motion of the people sharing the environment with the robot. The `Person` agent is responsible for detecting and tracking the upper body of the interviewed person, while the `HMC` agent captures the motion of the full body of the patient. To solve these tasks, both agents are connected to the `WinKinectComp` module. This module is in charge of capturing the preprocessed data provided by a Kinect sensor v2 (i.e. joints and face of the person).

`PELEA` is the most complex compoNet within the architecture. The use of `PELEA` and, in particular, of Automated Planning in the proposed cognitive architecture is motivated by its ability to react to unexpected or unforeseen situations. Automated

[2] http://www.metralabs.com/en/
[3] http://www.mira-project.org/joomla-mira/

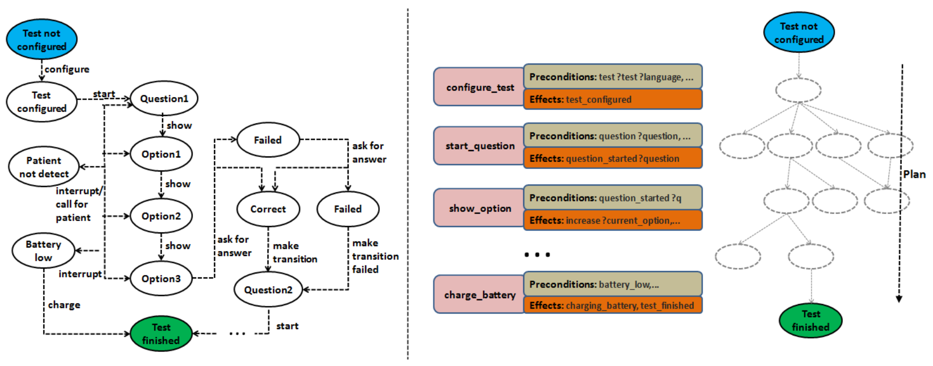


Fig. 7 (left) An example of modelization of the Barthel test as a FSM; (right) Modelization of the Barthel test as a planning problem.

Planning gives a domain description (in terms of available actions), an initial state and a set of goals, and generates a plan that allows achieving those goals from the initial state by executing the generated plan of actions. This form of deliberative reasoning is being increasingly used in robotic systems because it contains its own explicitly represented, symbolic and high-level model of the world, which is used to perform a forward projection (reasoning in depth about goals, preconditions, resources, and timing constraints). Systems with this type of reasoning generate plans to accomplish their goals and are therefore better suited to coping with uncertainty, reacting to unforeseen situations and recovering from bad decisions. In contrast, reactive systems, such as Finite State Machines (FSMs) or rule-based systems, do not hold a symbolic view of their environment, and designing them can be a complex and time-consuming endeavour because of the need to pre-code all of the behaviours of the system for all foreseeable circumstances.

By way of example, Figure 7 shows an example of a Barthel test modelled as an FSM (left) and using automated planning (right). The use of an FSM requires explicitly enumerating the states, then for each state, specifying the actions that are possible in that state, and finally, for each state-action pair, specifying the new state that results from carrying out the action in the prior state. Therefore, it requires explicitly defining the sequence of actions that lead from the initial state (blue node in Figure 7 (left)) to the goal state (green node). Obviously, this is an unpractical approach for a domain with a large number of states and actions. In fact, Figure 7 (left) presents only part of the full FSM required to perform a Barthel test. Instead, the modelization of this test as a planning problem (right) replaces the explicit enumeration of states and transitions with a description of the initial state (blue node), the goals (green node) and available actions (with its preconditions and effects), and lets the planner decide the sequence of actions to satisfy those goals. Therefore, automated planning makes the code more modular and easier to maintain. By decoupling the states from each other, each action can be worked on independently of the others. Additionally, if `PELEA` detects an unexpected state during the execution of a plan, the planner is invoked again to obtain a new plan that takes the system from that unexpected state to a state where the goals are met, even though the unexpected state has not been explicitly modelled. For instance, if at any time during a test the patient is outside the field of view of the robot, PELEA generates a new plan: it calls out to the patient and then continues with the rest of the test. This type of deliberative reasoning perfectly suits complex robotic tasks.

Furthermore, PELEA allows us to model the available actions, the initial state and the goals to be reached in a declarative form using a Planning Domain Definition Language (PDDL), regardless of the robotic platform or the specific programming language used in the low-level modules of the architecture. Such declarative language makes it easier to integrate new low-level modules in the architecture. The integration of such modules usually results in new robot skills (e.g., manipulation); hence, in the PDDL, the definitions in the domain description are updated with the new available actions along with their preconditions and effects. Such integration also requires from PELEA the translation of these high-level actions into low-level commands that are annotated in the DSR and interpreted by the corresponding new low-level modules, as well as the translation of the annotations of these modules in the DSR to a high level of abstraction to verify that the course of action is correct. Finally, this integration also requires the definition of the new module itself, with its corresponding agent able to annotate and read from the DSR.

In this project, PELEA includes the following components:

- The *High Level Executive* (HLE) module manages the whole compoNet. It receives the global goals and invokes the Monitoring module to obtain a plan that can achieve these goals. Then, the HLE module takes the first action of the plan and invokes the HighToLow module to decompose the action into low-level actions. These actions are then inserted into the DSR as changes in the model. The HLE looks at the changes in the DSR and, after a conversion to high level knowledge performed by LowToHigh, sends them to Monitoring, which checks whether the plan is executing properly.
- The *Monitoring* module is in charge of maintaining a high level model of the environment and of invoking the Decision Support module when any deviation in the execution of the plan arises. It detects, for example, that the user has not answered a question or is not facing the robot and tries to find alternate plans to solve these problems.
- The *Decision Support* module creates a plan starting from the current state, the goals to be achieved, the possible states of the world and the description of the changes the actions produce in the world state. To create this plan, the module invokes an automated planner that returns the sequence of actions that can achieve the desired goals.
- The *HighToLow* module converts the high-level actions of the plan created by the Decision support module into low-level actions that can be included in the inner world.
- The *LowToHigh* module converts the information contained in the inner world, which represents knowledge in the form of binary predicates, into n-ary predicates that the Monitoring module can use to reason about the correctness of the execution of the plan.

4.3 Encoding the grammar rules within the agents

One important idea of the proposed scheme is the distribution of the control, which is encoded within the agents in the architecture (see Figure 5). Thus, each compoNet gains its own set of rules that, as the example in Figure 3 shows, consists of triplets representing the states of the DSR after, during and before the execution of a specific action. This Grammar is local to the compoNet and is encoded at that moment within the code of the agent. In an architecture that is mainly driven by how the internalized world changes, the coherence of the encoding of each change and its global synchronization are fundamental aspects. Although we have briefly described how the world is internalized in the DSR in Section 3.2, we will provide here a more detailed description of how a Grammar is encoded within an agent.

Algorithm 2 illustrates how the specific grammar of the `Speech` compoNet is encoded in the agent. The `Speech` compoNet is in charge of translating the chosen sentence from text to speech and of recognizing the responses from the patient (via the Automatic Speech Recognition (ASR) set on the `WinKinectComp`). There are three main modules within Algorithm 2. finishSpeaking is launched by the `TTS` module to the `Speech` agent to inform the agent that a sentence has been said. In the DSR, this implies that **robot** is **speaking** must change to **robot** finish **speaking**. setAnswer is launched by the `Answer` module to the agent to inform the agent that a response has been captured. The DSR is changed from **person** waiting **answer** to **person** got **answer**. The DSR also provides the specific *answer*. Thus, these functions encode the final state of the two rules driving the responsibilities of the `Speech` compoNet (e.g., Figure 3(right) shows the result of launching finishSpeaking). The main loop of the (compute) agent searches for the changes mentioned in Algorithm 1 . In this case, we document the two situations that are launched: the waiting for a new response from the patient (waitingAnswer) and the saying of a new sentence (startSpeaking). In the first case, the change in the DSR is done without evaluating any additional constraint. In the second case, we will evaluate if the *label*, i.e., the sentence to say, has been changed (see Figure 3). The procedures to be addressed (i.e., the **process** { *action* } of Algorithm 1) are also launched (**canAnswer**() and **setText**(*label*), respectively). However, as described in Section 3.2, just before launching one of these procedures, the agent notifies the rest of agents that the process is under execution (publishing the new DSR model).

5 Experimental results

5.1 Comprehensive Geriatric Assessment (CGA)

The experimental setting was briefly presented in Section 4.1. This section provides additional details. Comprehensive Geriatric Assessment (CGA) is a clinical procedure for the evaluation of the frailty of older people and adequate treatment prescriptions. CGA is an interdisciplinary effort that requires the coordination of several clinical professionals with the aim of increasing both the quality and quantity of life of elderly people. Some of its benefits include improving diagnoses, creating correct, customized and proportional therapeutic plans, increasing functional autonomy, and reducing complications during hospitalizations and the

```
finishSpeaking
if getEdge(robot,speaking) == is then
    removeEdge(robot,speaking, is);
    addEdge(robot,speaking, finish);
    publishModification();
end
setAnswer
if getEdge(person,answer) == waiting then
    removeEdge(person,answer, waiting);
    addEdge(person,answer, got [answer]);
    publishModification();
end
compute
if worldChanged then
    waitingAnswer
    if getEdge(person,answer) == can then
        removeEdge(person,answer, can);
        addEdge(person,answer, waiting);
        canAnswer();
        model_modified = true;
    end
    startSpeaking
    if getEdge(test,test_part) == is_in then
        {q, label} = getEdgeAttribute(test,test_part);
        if label != label_back then
            removeEdge(robot,speaking, is_not);
            addEdge(robot,speaking, is);
            setText(label);
            model_modified = true;
        end
    end
    changeConfig
    if ... then
        ...
    end
    if model_modified then
        publishModification();
    end
end
```

Algorithm 2: Example of the Grammar encoded within the Speech agent

incidents of mortality. Considering the ageing of the world population, the importance of CGA and its associated costs will no doubt continue to increase.

CLARC is currently able to perform three tests: a functional test (Barthel [35]), a cognitive test (MMSE [36]) and a motion analysis test (Get Up & Go [37]). To evaluate how the proposed scheme works, we first summarize that these tests include closed-answer questions ("Select option 1, 2 or 3"), open-answer questions ("What day is it today?") and monitoring of simple ("Close your eyes") or complex ("Get up from the chair and walk three metres") patient movements. CLARC is intended to work with real patients in real-life hospital environments; thus, it needs to be much more than a simple survey tool. Before starting the tests, CLARC needs to introduce itself as an accessible and helpful *assistant* (or, at least, tool). Elderly people undergoing CGA tests are often not familiar with robotic technologies. It is crucial for CLARC to make the patients feel comfortable and reassured and to offer them natural and intuitive ways to interact with it. The hypothesis driving the design of CLARC is that the proposed architecture allows management of the interaction with the patient and adaptation of the course of action to exogenous events, such as the patient not answering a question, asking for help or leaving the room. This hypothesis was confirmed by the results of user tests performed in Seville with real patients. We evaluated the interaction through questionnaires. Furthermore, the scores provided by the robot were assessed and compared with those provided by medical experts. Section 5.3 summarizes these evaluations. The evolution of the DSR when the robot is performing a test is presented in Section 5.2.

Finally, during the tests, CLARC collects, saves and displays the responses. Using the CGAmed interface, the physician can either monitor the tests online or access and edit the results once the test finished. As the development of these abilities is not related to the organization and management of the DSR or the cognitive architecture built around it, they will not be evaluated in this work.

5.2 Qualitative analysis of our proposal

This section provides a qualitative analysis of how the DSR interacts with the agents to administer a Barthel test. Briefly, the Barthel test consists of ten items that measure a person's daily functioning, specifically the activities of daily living and mobility: Bladder and bowel function, transfers and mobility, grooming and dressing, bathing and toilet use, and feeding and stairs use. Because actions and perceptions are internalized within the DSR , the monitoring of its evolution allows us to track the whole execution of the use case. To reduce the complexity of the representation, we did not use the external tablet in this example. Thus, the patient-robot interface only involves the `Speech` and `Panel` agents.

The Barthel test requires that the robot speak and show on the touchscreen specific sentences that either ask test questions or provide assistance to the patient . The touchscreen is also used for showing introductory videos to the patient. The robot must hear or capture from the touchscreen the answers from the patient or relative. Finally, the robot needs to track the presence of the person, as s/he must remain seated in front of the robot. As mentioned above, the use case includes a first introduction, where the robot is introduced and the test is explained to the user. Then, the robot asks the user for ten items. Each item requires that the user chose the option that best fits his/her condition from a list. The presentation of each item follows the same scheme:

1. The robot introduces the item through voice and a message on the touchscreen
2. The robot describes each possible option (voice and text)
3. The robot asks the user to give an answer (voice or touchscreen interaction)
4. If after a specific time there is no response
   (a) The robot describes each possible option via voice again
   (b) The robot shows all options on the touchscreen

If after this second try the patient still does not answer, the robot will go to the next item in the test. Two non-answered items will cause the robot to ask the clinician to enter the room. Figure 8 shows the evolution of the DSR during the management of one question of the Barthel test. The figure shows how the state evolves when the person is lost and then is detected again. It can be noted that

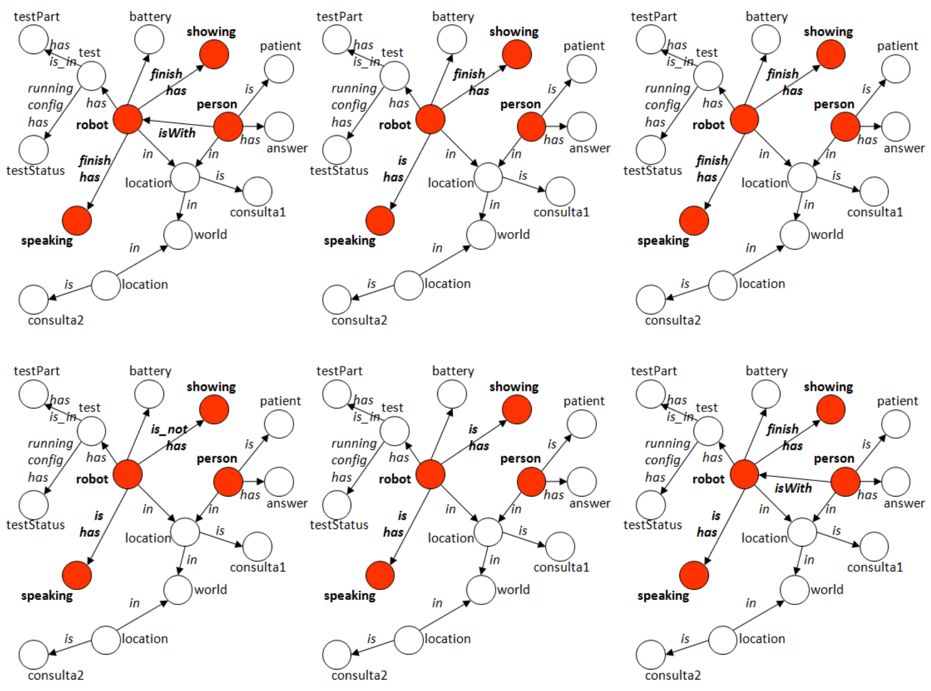


Fig. 8 Evolution of the DSR: During the execution of the Barthel test, the robot loses sight of the person for several frames and then asks the patient to sit in front of it again. More-related nodes and edges are coloured and in bold, respectively. The figure shows a sequence of views of the DSR: (top-left) before losing the person, the robot has just finished asking a question and is showing a message on the touchscreen; (top-center) the **robot** is **speaking** a new sentence but then it loses sight of the person. The situation is reported to PELEA; (top-right) the Speech agent immediately stops speaking; (bottom-left) the Speech and Panel agents return the DSR to its original state (**robot** is_not **speaking** and **robot** is_not **showing**) and PELEA changes the world, prompting the robot to say a specific sentence ('Please, sit down in front of me'). This sentence is encoded as a label on the is_in attribute between **test** and **testPart**; (bottom-centre) the Panel also shows this sentence on the touchscreen; and (down-right) the **person** is_with **robot** again and PELEA determines that the test can continue

the DSR practically provides a semantically annotated view of the scene. We have not measured response times, but currently, all the tests can be run online and without remarkable latencies. Section 5.3 captures the opinions of the patients about the interaction with the robot.

5.3 Quantitative evaluation

The CLARC robot was evaluated at a retirement home in Seville in November 2017. Following a voluntary interview with a professional physiotherapist working at the residence, eight patients were recruited. They formed a representative sample of the residence population in terms of IT (Information Technology) use and functional and cognitive abilities:

– Inclusion criteria: patients should be volunteers, over 55 years old and with a Mini-Mental State Examination (MMSE) score greater than 20 [36].

Table 2 Characteristics of the patients who participated in the evaluation of CLARC at a retirement home (Seville)

User ID	Gender	Age	Hearing Imp.	Visual Imp.	Cognitive Imp.
#1	Woman	86	no	no	no
#2	Woman	75	light	no	no
#3	Woman	84	light	medium	light due to the age
#4	Woman	55	no	light	light
#5	Man	93	medium	light	no
#6	Woman	84	no	light	light
#7	Woman	82	no	no	no
#8	Woman	92	light	no	no

– Exclusion criteria: patients were excluded if they had severe vision and/or serious hearing impairments.

Table 2 describes the main characteristics of the patients who participated in this evaluation. Some of them had serious motor impairments (patients #1 and #4 needed a wheelchair, patients #2 and #3 a walker). None of the patients had previously interacted with another robot. All patients had a mobile phone, but most used it only for phone calls or for taking photographs. Five of the patients also sporadically used tablets or computers with some simple computer applications. Only one of the patients (#4) used her smartphone and tablet continuously and extensively (using chat applications, navigating the Internet, etc.). The mean age of the patients was 81.37 ± 12.07 years. There were seven women (87.5%) and one man (12.5%).

Before starting the session with the robot, each patient was interviewed. There were three parts to the pre-test interview. The first was informed consent: the patient was given an explanation of the study protocol in detail and of his/her rights before s/he signed the informed consent document. Then, sociodemographic variables were collected: the patients age and IT tool use and proficiency. Finally, the user was asked his/her opinion about his/her perception of robots. The objective of this question was to acquire knowledge about a possible link between familiarity with technological interfaces and better performance in interacting with the robot and whether an a priori positive/negative attitude towards robots influenced performance.

Then, the patient was taken to a second room where he/she was introduced to the robot. The patient was invited to sit comfortably, and the robot's main features were explained. The patient answered the Barthel test with the robot. The patients were previously informed that the CLARC robot would explain to them at any times how to interact with it and that unless a technical problem should arise, they were not to communicate with the engineer in the room. The physiotherapist that designed the interaction with the robot for the Barthel test was present in the room and proceeded with his own Barthel Test evaluation while he observed the interaction of each patient with the CLARC robot. He coded his usual grid based on his observations and the patient's answers to the robots questions. However, in order to reflecting the patients current functional state as objectively as possible, after the sessions, the clinician compared and completed his scores (as necessary) with a previous Barthel Test administered a few days before at the retirement home as part of the medical care facilities of the institution. Once the test was finished, the patients were accompanied back to the first room.

Table 3 Patient's responses to specific questions from the post-test interview

Question ID	Description	Mean	SD
q1	I could hear and understand the robot clearly	4.14	1.21
q18	The robot was engaging and made me feel at ease interacting with it	4.43	0.53
q23	I had the impression that my oral answers were well taken into account	2.5	1.49
q24	It was easy to answer using the touchscreen or remote control when the robot 'did not hear'	5	0
q25	I had the feeling that it was 'natural' to speak with, the robot or use the touchscreen or the remote control	4.85	0.37
q28	I felt confident during the interaction with the robot	4.86	0.37

A post-test interview and satisfaction questionnaire were administered. Table 3 summarizes the mean and standard deviation of the patients answers on a 5-point Likert scale from 1 (strongly disagree) to 5 (strongly agree). The table provides some of the more significant questions evaluating the patient-robot interaction. From these responses, we can conclude that the proposed scheme allows the robot to interact with the patient in a fluent and natural manner.

Finally, although this is out of the scope of this proposal, we note that the robot and the clinician had complete agreement in the scores assigned to two of the questions of the Barthel test. In the other eight questions, a maximum of two users per question answered differently between the robot and the clinician. After revising the recorded videos and taking into account the clinician's notes taken while observing the session, four situations were detected:

1. the patient accidentally pressed the wrong button on the remote control (1 case);
2. the patients did not understand the question (2 cases);
3. the patients did not answer during the allocated time (2 cases);
4. the patients got confused with the remote device buttons: instead of pressing one of the option buttons (physical buttons labelled 1-4) after the question, they pressed the "yes"/"no" buttons immediately after listening to each option (they did not wait for the response period following the presentation of the whole question) (3 cases).

Only in one case could a patient not answer due to a technical issue.

6 Discussion

As Section 5.3 shows, the proposed scheme has been evaluated with real patients at a retirement home in Seville. Although the number of patients was very small, the initial tests were nonetheless successful. The agents interact using the DSR to allow the robot to follow the correct course of action but also to act against exogenous events such as the sudden absence of the interviewee (as shown in Figure 8). However, the distribution of global control comes with a cost: the major complexity of the DSR and the effort of the software developers to synchronize the interactions of the agents without discharging this responsibility onto a specific manager. This section briefly outlines the main advantages and disadvantages of this proposal based on our experience with CLARC.

Cons. It is clear that the complexity of the software components in charge of monitoring the evolution of the DSR (our agents) is higher than that of these same agents when they must only wait for a command to act. Although the architecture may give the impression of being modular, this is not so in its current version: the definition of the agents demands a high degree of coupling among all developers in charge of the implementation of each one of the agents. The reason for this dependence can be the need to work with the same collection of symbols/tokens, which was unfortunately not defined in advance. Thus, each change requires that all agents know how the change will be 'written' in the DSR. Agents must also carefully manage how they will advise the rest of the agents that they are working on a specific part of the DSR. The current messages in the DSR (such as **robot** is **speaking**) must be correctly interpreted by other agents that could be assigned to use the robots speakers. We must also endow the Grammars with priority levels to access a shared resource. This problem was noted by Hartley [29], as *similar states of the world could mean different things depending on the context*. Thus, *this would result in a behavior being activated when another behavior accidentally allowed the world to satisfy its preconditions*. Therefore, we again require a very close interaction among the whole team of programmers.

Pros. On the one hand, we have been able to encode the action and perception possibilities using the same scheme and collection of tokens, more or less elaborated by the domain-dependent modules, but always coming from the sensors. It is interesting to note that this provides a natural mechanism for merging top-down and bottom-up procedures for determining the course of action. In this proposal, the domain-dependent modules can respond in the same way to both kinds of processes. For instance, when the robot loses sight of a person, the `Speech` agent can immediately stop talking, and `PELEA` can change the global course of action. The local behaviour of the `Speech` module does not need to wait to receive a specific command from `PELEA` (or from any other *goal manager*). Thus, these modules can learn how to react to specific stimuli. In the scenario where CLARC works, this implies a fluent interaction with the patients.

However, the major advantage of this proposal is that the DSR stores a complete view of the activities that the robot is performing, joining perceptions and actions. Although the knowledge of the Grammars is defined by hand, the evolution of the DSR is driven by the agents without supervision. The sequence of *sentences* that any agent can read in the DSR (e.g., the robot is_not speaking) allows the agents to have at their disposal a detailed source of information for improving the management of a given situation. The context information is annotated in the DSR (e.g., the robot knows the questions in the test, the room, the name of the next patient, etc.) and can be augmented by adding new modules (e.g., an agent could monitor the face of the patient and easily annotate the expression on the graph) if required. The different leaves of the DSR evolve independently as there are no bounds. The coupling of several planners (the path planner for tracing new routes in a populated environment included on the `CogniDrive` agent and the conversational ability originating from the `PELEA` planner) can be synchronized in the DSR, as it is unnecessary to design a complex manager for determining what the current, and unique, task to be achieved is. The filtering required to prevent several agents from simultaneously accessing a resource, such as the speakers, can be set by joining the resource to a unique agent. This was the case for the `Speech` agent.

7 Conclusions and future work

This paper proposes a mechanism for bridging the perceptual and acting perspectives within a software architecture for robotics. The idea can resemble the blackboard model or scene graphs, as the agents in our proposal annotate on a short-term memory, the DSR, all the perceptual information they need to share with other agents for solving a task. Instead of maintaining a specific agent for determining the next goal/s to address, agents are now able to capture this information from the shared memory space. To correctly unfold this behaviour, each agent must manage its own knowledge base, where it maps configurations of the DSR (subgraphs) with actions. This constitutes an augmented mechanism for defining the affordance concept. In simpler cases, the existence of an object in the DSR can allow an agent to launch a specific action (welcoming a person). In most cases, the agent will need to verify a more global context. For instance, when the robot is navigating from one resident to another, a chair is simply an obstacle to avoid. When the robot is administering a Get up & Go test in a room, the chair is a key item, as the robot should ask the patient to stand up from this chair, walk in a straight line for approximately three metres, turn back, return to the chair and sit down. The contextual information should then be present in the representation. Additionally, when an agent is performing an action (e.g., navigating, speaking), it is consuming a significant resource (e.g., motors, speakers). We suggest that the agents inform their counterparts in the representation that an action has been launched and that, therefore, the resource is occupied. The current actions can be stopped or modified by any agent in the architecture. They are responsible for avoiding undesirable modifications of a course of actions. It is important to emphasize that the proposed scheme is currently running within a robot that is working with real patients. The preliminary results provided in this paper will be extended to significantly increase the number of tests. We have recently tested the ability of the robot to conduct the Get Up & Go test in the Rehabilitation Unit of the Hospital Civil at Malaga. There, nineteen patients with physical and/or neurological issues performed this test, while the CLARC robot autonomously evaluated their gait. This application opens the discussion about future work. Currently, the knowledge stored on the agents is set by our developers. This is applicable to the `PELEA` agent, which maintains a specific domain, but also to the rest of agents in the architecture. The idea of improving this knowledge by learning from experience is probably the most interesting work to be addressed. However, we will not conduct this research on CLARC. Interaction with the elderly in a real scenario is currently well defined and not open to exploration. We should instead instantiate the architecture on other scenarios where we can have the freedom to make mistakes.

Future work will focus on dealing with the main problems detected from these trials. We need to define a common dictionary of terms and predicates that cover all the knowledge bases of our agents. We also need to develop a mechanism that allows the agents to access specific parts of the DSR (e.g., the person) and not the whole representation. It is mandatory to set and manage priority levels for using resources, a complex task that should be addressed by mid-level planners. It is also mandatory to change the way agents are currently encoded, providing a mechanism that can allow an easier way to map subgraphs to actions. Finally, the collection of symbols must be *generalized* as a way to achieve the desired modularity of the agents. The current encoding, which allows *reading* the state of the outer world by a simple observation of the DSR, can be a good option for achieving this generalization. However, we must also be open to the possibility that while this encoding can ease our monitoring of the evolution of the DSR (and thus, the possibilities for sharing the information with humans), it might not be the best option for achieving autonomous learning by the task-dependent modules. Other encodings (e.g., low-level features or outcomes generated/employed by these modules) could be employed by the graph items that are closer to the sensors. This will increase the data volume represented in the DSR but also open the architecture to techniques such as deep learning, which could be applied by specific agents to generate the current symbols encoded using natural language. It is also essential to improve the ability of the robot to interact with users. Specifically, we need to work on speech entrainment [38], i.e., the capacity of interlocutors to become similar to each other during spoken interaction. Speech entrainment plays a relevant social role, since humans perceive people who entrain to their speaking style as more socially attractive and competent. Because of the significance of the verbal channel, this is clearly a necessary skill for a robot that needs to engage with elderly patients.

Acknowledgements The authors warmly thank the members of the "Amis du Living Lab" community and the patients and clinicians of Hospital Virgen del Rocío (Seville) for their participation in this research.

Compliance with Ethical Standards

Funding

This work has been partially funded by the EU ECHORD++ project (FP7-ICT-601116) and the TIN2015-65686-C5-1-R (MINECO and FEDER funds).

Conflict of Interest

Rebeca Marfil, Adrián Romero-Garcés, Juan P. Bandera, Luis J. Manso, Luis V. Calderita, Pablo Bustos, Antonio Bandera, Fernando Fernández and Dimitri Voilmy declare that they have no conflict of interest. Javier García is partially supported by funds from the Comunidad de Madrid (Spain) under research project 2016-T2/TIC-1712.

Ethical Approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Informed Consent

Informed consent was obtained from all individual participants included in the study.

References

1. Coradeschi S, Loutfi A, Wrede, B. A Short Review of Symbol Grounding in Robotic and Intelligent Systems. Künstliche Intelligenz 27 (2), 129–136, 2013
2. L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater and J. Santos-Victor, Affordances in psychology, neuroscience and robotics: a survey, IEEE Transactions on Cognitive and Developmental Systems, 99, 2017
3. Minsky, ML. The society of mind. Simon and Schuster, 1986
4. Krüger N, Piater J, Wörgötter F, Geib C, Petrick R, Steedman M, Ude A, Asfour T, Kraft D, Omrcen D, Hommel B, Agostino A, Kragic D, Eklundh J, Krüger V, Dillmann R. A Formal Definition of Object Action Complexes and Examples at Different Levels of the Process Hierarchy. EU project PACO-PLUS, 2009
5. Hoyningen-Huene NV, Kirchlechner B, Beetz M. GrAM: reasoning with grounded action models by combining knowledge representation and data mining. In International conference on Towards affordance-based robot control, E. Rome, J. Hertzberg, and G. Dorffner (Eds.). Springer-Verlag, Berlin, Heidelberg, 47-62, 2008
6. Blumenthal S, Bruyninckx H, Nowak W, Prassler E. A scene graph based shared 3d world model for robotic applications. IEEE International Conference on Robotics and Automation, 453-460, 2013
7. Hawes N, Wyatt J, Sloman A. An architecture schema for embodied cognitive systems. Tech. Rep. CSR-06-12, University of Birmingham, School of Computer Science, 2006
8. Manso LJ, Calderita LV, Bustos P, García J, Martínez M, Fernández F, Romero-Garcés A, Bandera A. A genera-purpose architecture for control mobile robots. Workshop on Physical Agents (WAF), 105-116, 2014
9. Mohan V, Morasso P, Sandini G, Kasderidis S. Inference through embodied simulation in cognitive robots. Cognitive Computation 5(3), 355–382, 2013
10. Magill K, Erden Y. Autonomy and desire in machines and cognitive agent systems. Cognitive Computation 4(3), 354–364, 2012
11. Newell A. Some problems of basic organization in problem solving programs. In M.C. Yovits, GT Jacobi & GD Goldstein (Eds.), Conference on Self-Organizing Systems. Washington D.C.: Spartan Books, 393-423, 1962
12. Laird JE. The Soar cognitive architecture. Cambridge, MA: The MIT Press, 2012
13. Anderson JR. How can the human mind occur in the physical universe? New York: Oxford University Press, 2007
14. Hayes-Roth B. A domain-specific software architecture for a class of intelligent patient monitoring agents. J. Exp. Theor. Artif. Intell., 8(2), 1996
15. Haber A, Sammut C. A cognitive architecture for autonomous robots. Advances in Cognitive Systems 2, 257–275, 2013
16. Lemaignan S, Warnier M, Sisbot EA, Clodic A, Alami R. Artificial cognition for social human-robot interaction: An implementation. Artificial Intelligence 247, 45–69, 2017
17. Beetz M, Mösenlechner L, Tenorth M. CRAM a cognitive robot abstract machine for everyday manipulation in human environments. IEEE/RSJ International Conference on Intelligent Robots and Systems 2010
18. Arrabales R, Ledezma A, Sanchis A. Simulating visual qualia in the cera-cranium cognitive architecture. In From Brains to Systems: Brain-Inspired Cognitive Systems 2010, Advances in Experimental Medicine and Biology, pages 223–238. Springer, 2011
19. Bello P, Bridewell W, Wasylyshyn C. Attentive and pre-attentive processes in multiple object tracking: A computational investigation. In Proc. 38th Annual Meeting of the Cognitive Science Society, pages 1517–1522, Philadelphia, PA, USA, 2016
20. Conforth M, Meng Y. Embodied intelligent agents with cognitive conscious and unconscious reasoning. In Proc. of the Int. Conf. on Brain-Mind, pages 15–20, Brain-Mind Institute, 2012

21. Hawes N, Zillich M, Wyatt J. BALT & CAST: middleware for cognitive robotics. Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication, ROMAN 2007, pp. 998-1003, 2007
22. Tenorth M, Beetz M. KNOWROB knowledge processing for autonomous personal robots. IEEE/RSJ International Conference on Intelligent Robots and Systems 2009
23. Zhang S, Sridharan M, Gelfond M, Wyatt J. Towards an architecture for knowledge representation and reasoning in robotics. In International Conference on Social Robotics (ICSR) 2014
24. Wintermute S, Laird JE. Imagery as compensation for an imperfect abstract problem representation. In N. A. Taatgen & H. van Rijn (eds.), Proceedings of the 31st Annual Conference of the Cognitive Science Society, 2009
25. Manso LJ, Bustos P, Bachiller P, Núñez P. A Perception-aware Architecture for Autonomous Robots. International Journal of Advanced Robotic Systems 12(174), pp.13. DOI: 10.5772/61742. 2015
26. Calderita LV, Manso LJ, Bustos P, Suárez-Mejías C, Fernández F, Bandera A. THERAPIST: Towards an Autonomous Socially Interactive Robot for Motor and Neurorehabilitation Therapies for Children. JMIR Rehabil Assist Technol. 1(1), DOI: 10.2196/rehab.3151, 2014
27. Romero-Garcés A, Calderita LV, Martínez-Gómez J, Bandera JP, Marfil R, Manso LJ, Bandera A, Bustos P. Testing a fully autonomous robotic salesman in real scenarios. IEEE International Conference on Autonomous Robots Systems and Competitions 2015
28. Alcázar V, Guzmán C, Prior D, Borrajo D, Castillo L, Onaindia E. Pelea: Planning, learning and execution architecture. PlanSIG10, 17-24, 2010
29. Hartley R, Pipitone F. Experiments with the subsumption architecture. Proceedings of the International Conference on Robotics and Automation (ICRA) 1991
30. E. Gat, On Three-Layer Architectures, *Artificial Intelligence and Mobile Robots*, 195-210, Cambridge: MIT Press, 1998
31. V. Katter and N. Mahrt. Reduced representations of rooted trees. *Journal of Algebra* 412, 41-49, 2014
32. Xiong W, Wu L, Alleva F, Droppo J, Huang X, Stolcke A (2017) The microsoft 2017 conversational speech recognition system. CoRR abs/1708.06073
33. Tsui KM, Dalphond JM, Brooks DJ, Medvedev MS, McCann E, Allspaw J, Kontak D, Yanco HA (2015) Accessible human-robot interaction for telepresence robots: A case study. Paladyn 6(1)
34. Kurniawan S, Zaphiris P (2005) Research-derived web design guidelines for older people. In: Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility, ACM, New York, NY, USA, Assets '05, pp 129–135, DOI 10.1145/1090785.1090810
35. F.I. Mahoney, D. Barthel (1965) Functional evaluation: the Barthel index. Maryland State Med Journal 14, 56-61.
36. M. Folstein, S. Folstein and P. McHugh (1975) Mini-mental state: a practical method for grading the cognitive state of patients for the clinician. Journal of Psychiatric Research, 12, 189-198.
37. S. Mathias, USL Nayak, B. Isaacs (1986) Balance in elderly patients: the get-up and go test. Arch Phys Med Rehabil. 67, 387-389.
38. Benus S. Social aspects of entrainment in spoken interaction. Cognitive Computation 6(4), 802–813, 2014
39. Kotseruba I, Avella Gonzalez O, Tsotsos JK (2016) A Review of 40 Years of Cognitive Architecture Research: Focus on Perception, Attention, Learning and Applications. CoRR abs/1610.08602
40. Manso L, Bustos P, Bachiller P, Gutierrez M (2012) Graph Grammars for Active Perception 12th International Conference on Autonomous Robot Systems and Competitions, 4–8, Guimaraes, Portugal