18th IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 25-27 2018, Torres Vedras, Portugal

Generating Symbolic Representation from Sensor Data: Inferring knowledge in Robotics Competitions

Rodríguez-Lera, Francisco J.
CSC Research Unit
University of Luxembourg
Esch-sur-Alzette, Luxembourg
francisco.lera@uni.lu

Martín-Rico, Francisco
GSyC Dept.
Universidad Rey Juan Carlos
Fuenlabrada, Spain
fmartin@gsyc.es

Matellán-Olivera, Vicente
Mech., Computer and Aerospace Eng. Dept.
Universidad de León
León, Spain
vicente.matellan@unileon.es

***Abstract*—Transformation of data gathered by sensors into symbolic knowledge is a recurrent problem in control systems, artificial systems, and robotics in particular. The control architecture of the robot reflects this process. This paper proposes an inference component devoted to transform sensor information from the sub-symbolic layer to a symbolic layer. Its operation has been illustrated using a navigation task on simulated and real scenario based on @home competition from the European Robotics League.**

***Index Terms*—grounding, representation, octomap, ROS**

I. Introduction

Transformation of data gathered by sensors into symbolic knowledge is a recurrent problem in control systems, artificial intelligence, and robotics in particular. Robotic control architectures reflect this problem. Two basic trends have driven the research in this strand of AI: the cognitive and the reactive approaches.

In the early days of electronic controlled robotics, sensors directly drive the behavior of robots. Controllers were based on the feedback loop cybernetic theory [1]. Some of them achieved interesting results considering the stage of the electronic technologies in that moment, such as Walter's turtle [2].

During the seventies, most approaches were based on the symbolic AI paradigms (abstraction, planning, heuristic search, etc.) [3]. These type of controllers were based on the advances achieved in the automatic generation of "plans", that is, a sequence of robot actions that have to be executed in order to achieve a goal. Automatic generation of these plans operates under the "closed-world" assumption [4]:

- The state of the real world can be formally and correctly observed and defined.
- The robot is the only agent which can modify the world.
- The robot actions have only the effects specified in its formal definition.

Under these assumptions, planners are given a complete description of the initial state of the world, the potential *actions* that the robot can perform, and a set of desired *goals*. The role of the planner is making a search into the tree of possible operators combinations to produce the sequence of robot actions (plan) that leads from the current situation to a situation on which the goals are fulfilled.

In the nineties, a strong criticism [5] to this approach emerged. It was based on the limitations of planners to cope with real-world uncertainties and dynamics. Authors supporting this argument claimed that intelligence results from the interaction with the environment, which gave the name to the trend: "reaction". In this way, autonomous robots intelligence would be a result of its interaction with the real world through its sensors and actuators. Systems built using these ideas, such as the Herbert robot [6], were able to deal with the real world successfully. Reactive systems can be considered sub-symbolic because they do not compile abstract knowledge to generate behavior.

In the 21st century, robots gather a lot of data from their sensors (Lidar, cameras, odometers, etc.) and face symbolic problems (interacting with humans), and have to cope with all of in real time. Hybrid architectures emerged as a result of a research process joining deliberative and reactive subsystems. As a collateral effect, a hybrid solution needs to join two different ways of managing robot environment: sub-symbolic and symbolic. The reason is that the reactive layer works at low level, which means that it manages information from the sensor and generates immediate robot behaviors. Notwithstanding, the information should be more elaborated in the deliberative layer (symbolic).

In this paper, we are focusing on the problem of generating symbolic knowledge from sensor data. It is presented in the literature as the anchoring problem [7]. It can be also enunciated converting sub-symbolic information into an internal symbolic representation capable of being used in a planner. This problem is also known as *Symbol grounding* or *anchoring*.

The anchoring problem has been addressed from the first attempts to implement cognitive skills in Artificial Intelligent systems. For instance, Soar architecture [8] faced this scenario more than 30 years ago. To cope with robot memory and actions, they used "productions", which is a piece of memory of the robot defined as a symbol structure that comprises the twofold condition-action. If a production is running because its conditions were accomplished, actions associated are instantiated, if there are not actions, the system generates new actions, therefore, new symbols.

This work has been partially funded by Junta de Castilla y León grant LE-028P17, Comunidad de Madrid grant RoboCity2030-Fase 3, and by Ministerio de Economía and Competitidad of the Kingdom of Spain under RETOGAR project (TIN2016-76515-R).

This research is carried out using ROSPlan Framework [9] for task planning. It is supported by Robot Operating System (ROS) [10] and it uses Planning Domain Definition Language (PDDL) which tries to standardize [11] Artificial Intelligence (AI) planning languages.

The anchoring problem continues open, and researchers continue applying different computational models for solving it. For instance, PELEA architecture [12] proposes a module called LowToHigh Module whose concept presents a mapping function to translate from a low-level state that contains the sensory information to a high-level state which presents the generalization of that information. Other authors have proposed more complex solutions inspired by models of the human mind [13]. Cognitive architectures as such as the Atomic Components of Thought-Rational (ACT-R) [14] or Symbolic and Sub-Symbolic Robotic Intelligence Control System (SS-RICS) [15] aim to represent information following human models of brain and associating them to a software component [16].

The solution that we propose is based on motivational principles supported on a previous research [17], rather than the previous brain models. Specifically, our solution is guided by the robot role, which identifies a set of goals attending robot goal. The role will guide our Knowledge Inference Component (KnIC) to generate new PDDL symbols using propositional logic.

We have chosen robotics competitions to illustrate how the system works because they are well-known scenarios built under strict rules and requirements but at the same time, these scenarios are complex and dynamic environments for the robots: objects moving, humans walking, long-term tasks. Thus, every time that the environment suffers a change, the robot has to update its plans accordingly.

The remainder of this paper proceeds as follows. Section II presents the control architecture we use in our robots, focusing on the knowledge inference component. Section III describes the implementation based on ROS and PDDL. Its use in robotic competitions is described in Section IV. Finally, Section V outlines major contributions and future works envisioned.

II. Knowledge Inference Component

We have included the Knowledge Inference Component (KnIC) in our architecture in order to generate symbolic information for the World Knowledge Database. It gathers information from two sources: low-level information from components managing sensors; and sub-symbolic information extracted from the Knowledge Database. The KnIC is integrated into a hybrid architecture, described in [17], that is depicted in Fig. 1.

The architecture divides the robot control into three subsystems: deliberative, reactive and motivational. The motivational system favors non-monotonous behaviors in the robot. The reactive subsystem aims to provide behaviors just-in-time to solve real and non-expected situations. The deliberative subsystem is the one that manages symbolic information to generate long-term behaviors and is the one that needs to be fed. The design of KnIC copes with information complexity, processing the information obtained from the Reactive Subsystem in two steps.

A. Information Complexity

From robotics point of view, symbolic information could have different levels and scales (considering General Theory of Information [18]) in order to manage the planes involved in a robot task. We propose three different levels of symbolic information useful for the deliberative layer. This is different from reviewed approaches as PELEA or SS-RISC, which propose only two: sub-symbolic, closer to reactive layers, and symbolic, closer to planners.

The proposal divides the information into three levels:

1) Low-level knowledge unit (LLKU): this information identifies an element in the robot scenario such as objects, sounds, humans, or locations.
2) Cognitive Knowledge Unit (CKU): it is a piece of information that defines an internal representation of knowledge where some level of cognition and abstraction has been applied. In essence, the information elaborated here comes from LLKU and knowledge defined previously, based on short-term and long-term memories approach proposed in Rastegar and Ahmadabadi work [19]).
3) High-level knowledge unit (HLKU): this unit identifies a complex piece of information. It links and mixes different LLKUs or CKUs in order to define a concept for the robot.

These three levels of information identify three different scales of complexity.

The motivation of this multiple-level approach is that in dynamic environments, under real-time restrictions, it is necessary to maintain the world model updated continuously to get the planner working on accurate information. However, it is also necessary to cope with some complex and long-term tasks as well as generate new goals. This process needs some level of transformation, which could not be managed in real time if the real world presents too many changes.

B. Knowledge Inference Process

KnIC defines three mechanisms for inferring the three types knowledge units. Fig.1 presents graphically the connections among the three mechanisms:

a) Knowledge Generation Mechanism 1 (KGM-1): This layer comprises a set of components called *Knowledge Providers - Level 1* or KP. These components define a model that links concrete information to symbolic information. In particular, a KP manages the information gathered from only one sensor. As a result, a KP processes the information to generate low-level symbolic information associated with an object captured by that sensor.

b) Knowledge Generation Mechanism 2 (KGM-2): This layer could be also defined as the ”Cognitive Model Layer - Level 2” (CML). It is an instrument that applies some level of cognition to the knowledge units generated by KGM-1.

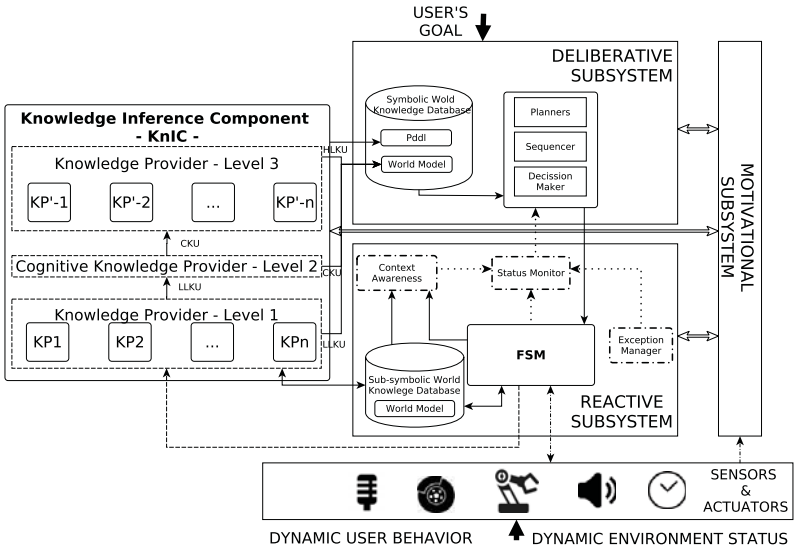

Fig. 1. Hybrid architecture containing the Knowledge Inference Component.

In particular, it correlates LLKUs and cognitive models such as ideas, abstractions, motivations, or beliefs. For instance, in our case we apply two cognitive models, motivational principles and spacial abstractions based on octomaps. The motivational principles define the robot role that could be: the companion mode, the service mode (separated in autonomous or semi-autonomous), and teleoperated mode. In each role, the information perceived and stored is processed at different rates and priorities. As a result, the knowledge generated (CKUs) is stored in the database to be consumed by the next layer.

c) Knowledge Generation Mechanism 3 (KGM-3): This layer comprises a set of components called *Knowledge Providers - Level 3* or $KP'm$. This layer is composed of a number m of KP' that can consume one or multiple CKUs from the previous layer. These components generate or eliminate High-level KU predicates in the knowledge database that will be used by the deliberative layer.

III. Implementation

As explained above, the hybrid architecture used in this research is composed by Deliberative, Reactive and Motivational subsystems. These subsystems are running on top of ROS. Particularly, each subsystem has specific components: the Reactive Subsystem runs Finite State Machines based on BICA [20]; the Deliberative Subsystem runs ROSPlan; and the Motivational Subsystem, which is a component developed on ROS, adds a certain degree of entropy to the system in order to avoid repetitive behaviors during robotics competitions.

The KnIC component converts information obtained at Reactive subsystem to the symbolic model in the Deliberative subsystem. This interchange of information is formalized using PDDL 2.1 standard. When the information is generated, the ROSPlan is fed with this information in order to generate new plans and take decisions ROSPlan is composed of two well-differentiated parts: the Knowledge Database and the planning system.

a) The Knowledge Database: Contains the information about the robot world. It can be considered as the robot memory and presents Symbolic and Syb-symbolic information. It consists of two components: the PDDL model, and a world model that links symbolic and sub-symbolic information. The PDDL model is initialized from a file with the domain, the instances of the problem, the initial predicates and the goals. The world model, defines the anchoring, if the PDDL model contains a location instance called *kitchen*, the model will contain the coordinates where the kitchen is in the real world.

b) The planning system: This system reads the information from the knowledge database, creates a plan, and dispatches the actions to carry it out. The planning system delivers the action and waits for it to be completed. If the action is completed successfully, add the effects of the action to the knowledge database and continue with the next action in the plan. If the action fails, the system creates another plan and executes again.

The actions implemented to manage the output of the Planning System must be coherent with the actions declared in the PDDL Domain File. Any change in PDDL domain implies changes in the implementation of actions.

ROSPlan provides an interface to implement the actions that have been defined in the PDDL model. These actions run in parallel with the responsive components running at the reactive subsystem.

A. KGM-1

We have used concepts of object orientation and multiple inheritances to facilitate communication with basic knowledge. The actions or any other software component can be a client of the knowledge database, accessing it through a simple

interface. This allows developers to avoid the complexity of this communication. Four operations provide interaction with the database:

- Add instance: Updates the database with a new instance of a selected type. For example: `add_instance("people", "Jack");`
- Add fact: Updates the knowledge with a new fact. For instance: `add_fact("robot_at", "kitchen")`
- Remove fact: brings up to date the knowledge removing an existing fact. For instance: `remove_fact("person_at", "kitchen");`
- Add goal: Updates the database with a new goal. For example: add_goal("robot_at", "bedroom");

Using ROSPlan the action delivered by the planning system was managed by the component responsible for its implementation. The programmer of the action was responsible for the entire implementation, returning true or false depending on the success of the action.

We have added interfaces to check the conditions available in the definition of durative actions in PDDL (*AtStart*, *AtEnd* and *OverAll*) used in ROSPlan.

Algorithm 1: Action Interface.

```
if checkAtStartConditions(action) then
    return false
if not activation(action) then
    return false
retval = true
finished=false
while not finished and retval do
    if not checkOverAllConditions(action) then
        retval = false
    else
        retval = step(action, finished)
if not deactivation(action) then
    return false
if ret and not checkAtEndConditions(action) then
    return false
return retval
```

These interfaces check the conditions, the activations/deactivations, and the execution of the action. In summary, this is presented as:

- `activation()`: This method is called at the beginning of the execution of the action. If it returns false, it means a failure in the activation.
- `step()`: This method is iteratively called during the execution of the action and returns if the action has finished.
- `deactivation()`: This method is iteratively called when the execution of the action finished. If it returns false, it means a failure in the deactivation.

If the algorithm returns false, it indicates that the execution of the plan has failed, and the planning system stops until it is called again. If the algorithm returns true, the action successfully finishes, the knowledge database is updated with the effects of the actions, and the plan continues executing, as shown in Algorithm 1.

B. KGM-2 + KGM-3

At the implementation level, the KGM-2 defines two main cognitive models. The robot role, set in companion mode, proposes direct relationships to those objects that help to improve the interaction: glasses, remote, tablet,... and habitual spaces of interaction such as living room, bedroom and kitchen.

In addition, a model supported in octomaps is applied to the spatial knowledge. An octomap [21] is a compact structure for storing in three-dimensional spaces occupation/probability values in a range $[0, 1]$. In each case, a location is encoded as an octomap with respect to a global reference axis.

The KGM-3 manages and mixes previous information for updating the outdated information available at runtime in the deliberative knowledge database.

C. Example

The implementation can be illustrated as follows. Initially, a KP dedicated to detecting and recognizing people would be constantly processing information from the camera, and when it finds *Jack*, a particular person, it adds an instance to the knowledge database (only once at the first detection) and publishes an octomap in the topic `people_jack` that represents in real time the space he occupies in the environment.

The cognitive layer receives this information and associates it with the octomap associated this location and the priority list associated with the robot role. As a result, these associations are converted into CKUs, updating the information in the database, and providing 3D information of the person.

Kp' components are responsible for extracting relationships from CKUs to insert predicates in the knowledge database. A Kp' component can subscribe to several types of Intermediate Knowledge units. Let's see some examples:

- A Kp' could subscribe to the information of people to generate predicates such as `(person_near Jack Nick)`.
- A Kp' could subscribe to the information of the people and to the locations to generate predicates such as `(person_at Jack Kitchen)`. To do so, I would only have to check the spatial intersection of each octomap of people with each octomap of locations and when they overlap, generate the knowledge.
- A Kp' could subscribe to the information of the sound events and to the locations to generate predicates such as `(sound_at Doorbell corridor)`.
- A Kp' could subscribe to the information of the locations and the information, external to the architecture, of the position of the robot generated by its self-localization module to generate predicates in real time such as `(robot_at kitchen)`.

These components are also responsible for removing from the knowledge database those predicates that detect false.

If detected `(robot_at kitchen)`, any other predicate `(robot_at X)` is removed, if `X` is not `kitchen`.

IV. Application: Robot Competitions

Robotic Competitions [22], [23] present two types of scenarios where our component could be tested: Functionality Benchmarks and Task Benchmarks. The Functionality Benchmarks are used to evaluate individual robot functionalities. Task Benchmarks evaluate robot success during the execution of a given task.

In this section, we will describe the application of KnIC in the task benchmark named "Catering for Granny Annie's Comfort" defined by the European Robotics League Rulebook.

A. A Case Study: Real-Time Knowledge Generation in the Granny Annie's Comfort Domain

In this task, the robot helps an individual called Granny Annie with some of her daily tasks.

There are different inputs to be recognized that are important to take high-level decisions:

1) Objects: the list of possible elements involved during the task execution
2) Locations: there are three different locations associated with this task: human, robot or an object.
3) Spoken commands: they define the goals to be fulfilled by the robot

On the 2D map of the surroundings, each of the rooms has been marked. The producer component of the spatial knowledge system in charge of the locations uses this information to obtain which regions of the space belong to each room. This component produces, for each room, a different octomap labeled with each room, as shown in Fig. 2.

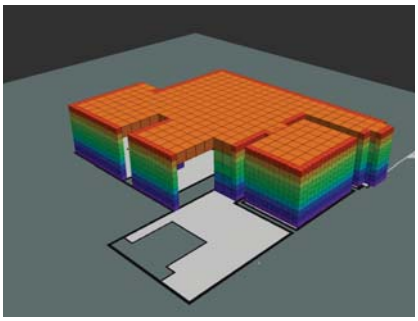

Fig. 2. The 2D location of each room defines a space (kitchen, living room) at the symbolic level (a 3D container). All the rooms are selected but the bedroom (closest room)

The producer component of the spatial knowledge system is in charge of localizing the objects. It starts with an empty octomap for each type of object. As soon as it detects an object, it inserts its 3D position into an octomap. This is done using the robot's self-locating position and the 3D sensor information.

A consumer of the spatial knowledge system reads the information from both producers to insert in the knowledge base predicates such as:

```
(object_found glasses)
(object_at glasses bedroom)
```

B. Trace of execution

Next, we are going to show how the situation is planned in which Granny Annie asks the robot to look for her glasses and take them away. The relevant parts of the model in PDDL are:

```
(define (domain erl)

(:types location robot object)
(:predicates
    (kitchen_location ?l - location)
    (bedroom_location ?l - location)
    (entrance_location ?l - location)
    (living_room_location ?l - location)
    (robot_at ?r - robot ?l - location)
    (object_at ?o - object ?l - location)
    (object_found ?o - object)
    (object_not_found ?o - object)
    (explored ?l - location)
    (not_explored ?l - location))
(:durative-action move_to
    :parameters (?from ?to - location ?r - robot)
    :duration ( = ?duration 10)
    :condition (and (at start (robot_at ?r ?from)))
    :effect(and
        (at start (not (robot_at ?r ?from)))
        (at end (robot_at ?r ?to))))
(:durative-action explore_location
    :parameters (?r - robot ?l - location ?o - object)
    :duration ( = ?duration 10)
    :condition (and
        (at start (not_explored ?l))
        (at start (robot_at ?r ?l)))
    :effect(and
        (at end (explored ?l))
        (at end (not(not_explored ?l)))))
(:durative-action explore_house_for_object
    :parameters (?lk ?lb ?le ?lv - location ?o - object)
    :duration ( = ?duration 10)
    :condition (and
        (over all(object_not_found ?o))
        (at start(explored ?lk))
        (at start(explored ?lb))
        (at start(explored ?le))
        (at start(explored ?lv))
        (at start(kitchen_location ?lk))
        (at start(bedroom_location ?lb))
        (at start(entrance_location ?le))
        (at start(living_room_location ?lv)))
    :effect(at end (object_found ?o))
(:durative-action carry_object
     :parameters (?r - robot ?o - object ?from ?to - location)
      :duration ( = ?duration 10)
      :condition (and
        (at start (object_found ?o))
        (at start (robot_at ?r ?from))
        (at start (object_at ?o ?from)))
      :effect(and
        (at end (not (robot_at ?r ?to)))
        (at end (not (object_at ?o ?to)))
        (at end (robot_at ?r ?to))
        (at end (object_at ?o ?to)))))
```

As soon as Granny Annie asks the robot to look for her glasses, a "problem" is generated with the information obtained from the producers of the spatial knowledge system, and the goal set:

```
(define (problem erl_task)
(:domain erl)
(:objects
    kitchen bedroom entrance living_room - location
    glasses - object
    r2d2 - robot
)
(:init
    (robot_at r2d2 bedroom)
    (kitchen_location kitchen)
    (bedroom_location bedroom)
    (entrance_location entrance)
    (living_room_location living_room)
    (not_explored kitchen)
    (not_explored bedroom)
    (not_explored entrance)
    (not_explored living_room)
    (object_not_found glasses)
    (object_at glasses living_room))
(:goal (object_at glasses bedroom)))
```

Note, that in PDDL, the action of exploring the house produces the predicate `(object_found glasses)` at the end. It is clear that if the robot explores the house and can not find them, this predicate can not be added. But we

have to design the model in PDDL in this way because otherwise a correct plan could not be calculated. In the same way, it is necessary to start the problem with the predicate `(object_at glasses living_room)`, which is not true, because otherwise the plan can not be calculated. This predicate will be deleted at the start of the robot operation, and when the object is found, the real predicate will be added. PDDL only adds or removes predicates as the effect of the execution of actions. In a real system, the predicates can be added or eliminated depending on the perception of the robot. If the house is explored and a predicate is added as a result of this action, the spatial knowledge system would remove this predicate, as it determines to be false.

The plan is generated based on the model and the problem, establishing a sequence of actions which first part is:

```
(move_to r2d2 bedroom kitchen)
(explore_location r2d2 kitchen glasses)
(move_to r2d2 kitchen entrance)
(explore_location r2d2 entrance glasses)
(move_to r2d2 entrance living_room)
(explore_location r2d2 living_room glasses)
```

If the glasses were detected in the entrance, after the spatial knowledge system updates, the action `(explore_location r2d2 entrance glasses)` would fail and a replanning would occur. The plan finally executed is:

```
(move_to r2d2 bedroom kitchen)
(explore_location r2d2 kitchen glasses)
(move_to r2d2 kitchen entrance)
(explore_location r2d2 entrance glasses) <-- FAIL!!
[replanning]
(carry_object glasses entrance bedroom)
```

V. Conclusions

Current scenarios in robotics competitions present long-term tasks developed in highly dynamic environments. As a result, decision making based on finite state machines are no longer enough and it is necessary to add a deliberative component to robot architecture. In this scenario, it is a requirement to maintain a knowledge database with high-level and low-level information in order to manage deliberative approaches.

This paper has highlighted the importance of a process to generate and maintain symbolic information in world-model database. Conceptually, we have devised a component for generating high-level premises using a three layers component. These three layers generate three types of knowledge units. First layer generates symbolic information "Low-level knowledge unit". Second layer generates elaborated symbolic information called Cognitive Knowledge Units according to pre-defined cognitive models. Finally, a third level mixes 1 or multiple CKUs obtaining "High-level Knowledge Units".

In addition, this research provides a real example using technologies from the state of the art: ROS, ROSPlan and PDDL models. Despite the fact that all of these tools have an outstanding value in their respective fields, the effort to integrate and customize them to solve real tasks during robot competitions is considerable. These initials results are encouraging and we plan to use this model in the next RoboCup2018.

References

[1] N. Wiener, *Cybernetics: Control and communication in the animal and the machine*. Wiley, 1948.

[2] G. Walter, *The Living Brain*. Duckworth, 1953.

[3] N. J. Nilsson, "Shakey the robot," SRI INTERNATIONAL MENLO PARK CA, Tech. Rep., 1984.

[4] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[5] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, no. 47, pp. 139–159, 1991.

[6] J. H. Connell, *Minimalist Mobile Robotics: A Colony-style Architecture fo a Mobile Robot*. Cambridge, MA: Academic Press, 1990, latas de Coca-Cola.

[7] S. Coradeschi and A. Saffiotti, "An introduction to the anchoring problem," *Robotics and Autonomous Systems*, vol. 43, no. 2-3, pp. 85–96, 2003.

[8] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial intelligence*, vol. 33, no. 1, pp. 1–64, 1987.

[9] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system," in *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15)*, 2015.

[10] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[11] M. Fox and D. Long, "pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[12] A. Ordonez, V. Alcázar, D. Borrajo, P. Falcarin, and J. C. Corrales, "An automated user-centered planning framework for decision support in environmental early warnings," in *Ibero-American Conference on Artificial Intelligence*. Springer, 2012, pp. 591–600.

[13] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind." *Psychological review*, vol. 111, no. 4, p. 1036, 2004.

[14] J. R. Anderson and C. J. Lebiere, *The atomic components of thought*. Psychology Press, 2014.

[15] T. D. Kelley, "Developing a psychologically inspired cognitive architecture for robotic control: The symbolic and subsymbolic robotic intelligence control system (ss-rics)," *International Journal of Advanced Robotic Systems*, vol. 3, no. 3, p. 32, 2006.

[16] E. Avery, T. Kelley, and D. Davani, "Using cognitive architectures to improve robot control: Integrating production systems, semantic networks, and sub-symbolic processing," in *15th Annual Conference on Behavioral Representation in Modeling and Simulation (BRIMS)*, 2006.

[17] Authors list removed for review, "Title blinded for review," *Cognitive processing*, 2018.

[18] M. Burgin and R. Feistel, "Structural and symbolic information in the context of the general theory of information," *Information*, vol. 8, no. 4, p. 139, 2017.

[19] F. Rastegar and M. N. Ahmadabadi, "Grounding abstraction in sensory experience," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*. IEEE, 2007.

[20] C. Aguero, J. M. Canas, F. Martin, and E. Perdices, "Behavior-based iterative component architecture for soccer applications with the nao humanoid," in *Proceedings of the 5th Workshop on Humanoid Soccer Robots @ Humanoids 2010*, 2010, pp. 29–34.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[22] S. Schneider, F. Hegger, A. Ahmad, I. Awaad, F. Amigoni, J. Berghofer, R. Bischoff, A. Bonarini, R. Dwiputra, G. Fontana *et al.*, "The rockin@ home challenge," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*. VDE, 2014, pp. 1–7.

[23] F. Amigoni, E. Bastianelli, J. Berghofer, A. Bonarini, G. Fontana, N. Hochgeschwender, L. Iocchi, G. Kraetzschmar, P. Lima, M. Matteucci *et al.*, "Competitions for benchmarking: Task and functionality scoring complete performance assessment," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 53–61, 2015.