IEEE International Conference on Robotics and Automation (ICRA 2016)
Stockholm, Sweden, 16-21 May 2016

# From Human Instructions to Robot Actions: Formulation of Goals, Affordances and Probabilistic Planning

Alexandre Antunes, Lorenzo Jamone, Giovanni Saponaro, Alexandre Bernardino, Rodrigo Ventura
Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
{aantunes, ljamone, gsaponaro, alex, rodrigo.ventura}@isr.tecnico.ulisboa.pt

***Abstract*— This paper addresses the problem of having a robot executing motor tasks requested by a human through spoken language. Verbal instructions do not typically have a one-to-one mapping to robot actions, due to various reasons: economy of spoken language, e.g., one short instruction might indeed correspond to a complex sequence of robot actions, and details about action execution might be omitted; grounding, e.g., some actions might need to be added or adapted due to environmental contingencies; embodiment, e.g., a robot might have different means than the human ones to obtain the goals that the instruction refers to. We propose a general cognitive architecture to deal with these issues, based on three steps: i) language-based semantic reasoning on the instruction (high-level), ii) formulation of goals in robot symbols and probabilistic planning to achieve them (mid-level), iii) action execution (low-level). The description of the mid-level is the main focus of this paper. The robot plans are adapted to the current scenario, perceived in real-time and continuously updated, taking in consideration the robot capabilities, modeled through the concept of affordances: this allows for flexibility and creativity in the task execution. We showcase the performance of the proposed architecture with real world experiments using the iCub humanoid robot, also in the presence of unexpected events and action failures.**

## I. Introduction

With the growing presence of robots in public spaces, human-robot interaction is becoming increasingly important. When communicating in natural language, humans omit many details that are left to the interpretation of other participants, since they all share common sense knowledge and have similar motor skills. This is not, however, the case of a robot, that has to reason on the natural language instruction, understand the goals, plan a sequence of actions to achieve them, and then execute the planned actions. This setup can be translated into three steps: semantic reasoning to understand the instruction (high-level), goal formulation and planning (mid-level), and motor execution (low-level). In this paper we present a general architecture that supports such behaviours in autonomous robots, with focus on the mid-level step, and the integration of different sources of knowledge: i) robot prior knowledge, in the form of learned affordances; ii) semantic knowledge, provided by semantic language-based reasoning on the verbal instructions; iii) perceptual knowledge, obtained from the robot perception about the current state of the world.

Since the early days of Artificial Intelligence (AI), planning techniques have been employed to allow agents to achieve complex tasks in closed and deterministic worlds.

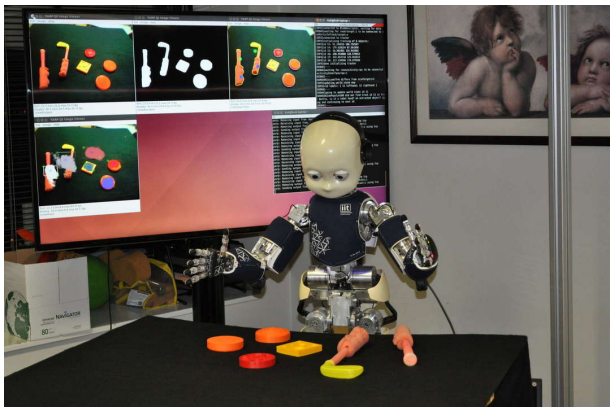


**Fig. 1:** The iCub robot looking at a table full of objects. In the back screen, images from the robot cameras showing object segmentation and tracking.

Every action has clearly defined pre-conditions and deterministic post-conditions; the world is assumed to be fully observable, and only the agent's actions can change the world state. These assumptions are not plausible for a real robot acting in real unstructured environments: the consequences of actions are not deterministic, the world is perceived through noisy sensing and it is constantly and dynamically changing due to unpredictable events.

In this paper we tackle the planning problem by resorting to *probabilistic planning*, in which actions with probabilistic outcomes are considered. These probabilities depend on the characteristics of the specific objects the robot has to act upon, and on the robot perceptual and motor apparatus. In our framework, these probabilities are extracted from the knowledge of affordances, that the robot has acquired through motor exploration. In this way, the set of actions available to the planner (i.e., the actions repertoire of the robot) can be grounded to the specific context when the robot looks at the objects around, enabling it to plan the sequence of actions that has the highest probability to achieve the goals. Additionally, the robot continuously monitors the state of the world, in order to be able to react to unexpected events (e.g., action failures, erroneous perception, modifications in the environment due to other agents).

Another important aspect of our approach is to exploit the semantic reasoning performed by the high-level to help the probabilistic planner find a solution. In our implementation, the high-level is represented by a language-based semantic

network, the PRAXICON [1]. Indeed, if the planner had to compute a sequence of actions that goes from the initial conditions to the final goal, it might not converge to a solution, or it might take too long to do it, due to the huge amount of possibilities it has to consider. Instead, the high-level step suggests a sequence of actions that lead to the goal, based on semantic reasoning about i) the human verbal request and ii) the names of the objects available; the expected outcomes of these actions are expressed as states of the world that can be perceived by the robot (e.g., "grasp bread with left hand" becomes *bread_inhand_left*), which constitute the sub-goals of the planner. Planning toward these sub-goals reduces the search space of the probabilistic planner, making complex problems solvable. In short, our system (mid-level) merges semantic knowledge with robot prior knowledge coming from the affordances and perceptual knowledge about the current context by using probabilistic planning; the output of the mid-level is the motor actions to be executed by the low-level, which in our case is represented by the iCub robot motor control libraries [2], [3].

The rest of the paper is organized as follows. In Sec. II we review the related work in the fields of planning and affordances models. Then in Sec. III we describe each component of our system, separated in domain-independent and domain-dependent. In Sec. IV we present experimental results that showcase a number of situations that our approach allows to deal with. Finally, in Sec. V we report our conclusions and sketch the future work.

## II. Related Work

A vast amount of work has been done in the area of combined task and motion planning [4]. Hierarchical planning was also studied in detail, with some algorithms available that incorporate Hierarchical Task Networks (HTN), like SHOPS2 [5], [6]. These methods, however, usually consist of one plan, which could be followed and completed. Some work has been done on simultaneous plan and execution on Hierarchy Planning in the Now (HPN) [7], which is oriented towards the execution of geometrical problems. Below we will go into detail on the points of contact.

**From Natural Language to Action Execution** One of the aspects we are working with in this system is the translation of natural language instructions into robot knowledge and actions. This field has seen some recent studies, as shown in PRAXICON [1] and DIARC [8]. In DIARC, a similar architecture was developed, where Natural Language instructions would be translated and executed by the robot, but while the system allowed for a failure detection, it did not re-plan nor use robot prior knowledge in the form of affordances. In PRAXICON, a human instruction is decomposed into a set of deterministic, human-like actions, that do not take into account the world around the robot, thus requiring planning on these instructions.

**Hierarchy Planning in the Now** The problem of real-time planning and execution requires an algorithm that can adapt to changes in the state. HPN introduces this by updating the world state at each step, and planning from there. Using hierarchy on the actions, it transforms a big, hard-to-solve problem into several smaller problems that are more easily solved by a planner. This approach was initially suggested by Nourbakhsh [9], where a big problem would be turned into smaller problems by completing sub-goals and re-planning. Our proposal uses Nourbakhsh's concept applied to a probabilistic planner, where we merge semantic knowledge with robot prior knowledge and the geometry of the world state.

**Combining Symbolic and Geometrical Planning** In the works of Kaelbling and Lozano-Pérez [7], a depth-first approach is used, where a plan-branch is recursively expanded until all that is left is a primitive task that can be executed. The problem to this approach is the feasibility of the plan, and the solution is to evaluate the serializability of the sub-goals, guaranteeing that no step in the plan makes it impossible for the other sub-goals to be achieved. This is a very different approach from the one we are using, which could be considered breadth-first, in the sense that we have the abstract plan, with all the sub-goals found, before we proceed with the execution.

**Multi-Level Planners** One of the concepts discussed in this paper is the use of different types of planners to solve a problem. This kind of work comes up in some papers [5], [7], [10], but is usually secondary or only somewhat present, in the sense that a set of instructions already exists, or the actions for the robots are predetermined. Our approach tries to create a full chain, from a human instruction, very abstract, to very specific motions at the lower level, using PRADA [11] for the mid-level probabilistic planner: we choose this planner because it allows to seamlessly incorporate the prior robot knowledge encoded in probabilistic terms (in our case, using the Bayesian Network of affordances we previously developed [12]) and because it is quite fast and accurate in planning the first best action of the plan, permitting real robot operations.

**Affordances and Robot Knowledge** A number of computational models have been investigated in the robotics literature to learn object affordances and use them for prediction [13], tool use [3], [12], [14], [15] and planning [16]–[19]. In the framework presented by Montesano et al. [13], objects affordances are modeled with a Bayesian Network [20], a general probabilistic representation of dependencies between actions, objects and effects. We recently extended this model to deal with actions that involve two objects: a held object (tool) and an affected one [12]. In [17] the relational affordances between objects pairs are exploited to plan a sequence of actions to achieve a desired goal, using probabilistic reasoning; however, how these interactions are affected by different geometrical properties of the objects is not investigated. In [18] the robot learns first affordance categories and then logical high-level rules, which are eventually encoded in Planning Domain Definition Language (PDDL), enabling symbolic planning with off-the-shelf AI planners. In a follow-up work [19] the generated plans are used in a real-world object stacking task and new affordances that appear during plan execution are discovered. The robot is able to

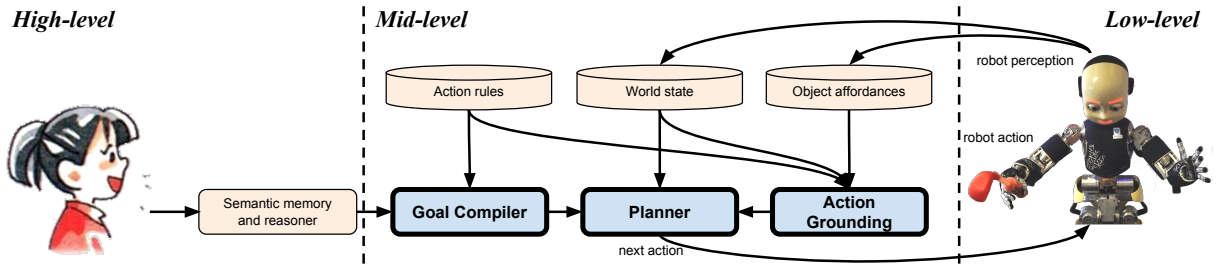



**Fig. 2:** Proposed architecture. The high-level side deals with interpreting human instructions provided in Natural Language; the mid-level part formulates robot goals and a probabilistic plan, also taking into account the state of the world (robot environment) and object affordances; the low-level block manages motor execution and control. Blocks in the upper row are specific to the assigned task (e.g., they would be different in an assembling scenario and in a cleaning scenario), while those in the lower row are general. This paper describes the mid-level part, in particular the goals, the actions grounding and the planner components – highlighted with dark background and bold text.

build stable towers exhibiting some interesting reasoning capabilities, such as stacking larger objects before smaller ones.

## III. System Architecture

The proposed system merges three different sources of knowledge: i) language-based semantic knowledge about the task, ii) perceptual information about the current context, and iii) robot prior motor experience. This is realized through four domain-independent modules: Semantic memory/reasoner, Goal Compiler, Action Grounding and Planner; one domain-dependent module: World State Monitor; and two databases: Rules (possible robot actions) and Object Affordances (see Sec. III-B). This paper focuses on the domain-independent modules, in particular the three highlighted blocks in Fig. 2. The system composed by these modules is very generic, and can be adapted to most situations.

### A. Action Rules

A rule (ungrounded action) is a symbolic template for each action. Each rule is defined by: i) Symbol, e.g., *grasp__obj_with__hand*; ii) Context, defining the pre-conditions necessary to execute said action, e.g., *_hand_clearhand, _obj_isreachable_with__hand*; iii) Outcome, consisting of a list of possible outcomes, organized from most to least likely, e.g., *_obj_inhand__hand*. Objects are indicated by *_obj* and hands *_hand*, when grounding. This is explained in detail in Sec. III-F.

### B. Affordances

Affordances, defined by J. J. Gibson as action possibilities offered by the environment [21], depend both on the object properties and on the sensorimotor capabilities of the agent. In a previous work we proposed a computational model of affordances which represents relationships between actions, visual features of objects (both a held object, or *tool*, and an affected one) and the resulting effects [12], [15]. The model is learned by the robot through sensorimotor experience. In this work, we refer to such model as *inter-object affordances*. The model represents the robot prior knowledge about what consequences its own actions can generate in the world. The Action Grounding module queries the affordance model to obtain the probabilities of the action effects given the visual properties of the objects involved: $P(E|A, O, T)$, where $E$ are the effects, $A$ is the action, $O$ are visual features of the affected object, and $T$ are visual features of the held object (or *tool*) used by the robot. For further details about the affordance model, the reader can refer to [12], [15].

### C. World State Monitor

One of the sources of information for the planner is what we call World State, as shown in Fig. 2. This block consists of a *short-term memory of the symbols* needed by the planner. These symbols pertain to characteristics of the objects present in the environment as well as robot proprioception, for instance: spatial position of an object, whether it is currently grasped by a hand or "free", in which hand it is, a set of shape descriptors (used for affordance inference, see Sec. III-B), etc. This memory, or database, is kept consistent by fusing the information that comes from different sensory modules in real time (visual routines such as object feature extractor and tracking, shown in the background of Fig. 1), applying some task-specific constraints (e.g., whether far objects are in-reach with the help of a tool, by using an optimization procedure based on an internal simulation of the robot actions [3]), maintaining symbols of occluded or disappeared objects in memory and applying temporal filtering to remove noise.

### D. Semantic Memory and Reasoner

Natural language instructions are usually very abstract in details, often assuming the other agent is seeing the same objects, can execute the same actions, and understands what is being asked. This is not true for robots, which means such an instruction has to be processed. This is done by semantic memory and reasoner, PRAXICON [1], which decomposes an abstract instruction into an action sequence using the objects the robot can see, e.g., "make a sandwich" would return "hand grasp cheese, cheese reach bun-bottom, hand put cheese…". While the semantic reasoner provides a sequence of human-like actions, it does not take into account if the objects are robot-usable (e.g., if an object is reachable), or if the robot can execute the actions (e.g., self- or environment collisions).

### E. Goal Compiler

In order to translate the instructions provided by the semantic reasoner, from human-language to robot-language, some common ground has to be found. A way to achieve this would be to use robot symbols to describe human actions and their results. This translation is performed by the Goal Compiler. Instructions provided by the semantic reasoner come in the form *(object action object)*, allowing for easy identification of the action. The algorithm searches for a matching action, with a similar symbol, e.g.: for *(hand grasp cheese)*, it finds *grasp__obj_with__hand*. Finally, it creates a sub-goal from the first (most likely) outcome of this action. The process is cyclic, using the previous sub-goals and changing them with the outcomes of the following actions, until the final goal is created.

During this translation, the Goal Compiler evaluates if these instructions can be executed, given the previous state, in order to detect inconsistencies. This is useful since the semantic reasoner lives in the conceptual space, without any knowledge of the real world. This allows the compiler to detect when an object that cannot be used appears in an instruction, which would be impossible to solve, providing some early-detection of mistakes.

The Goal Compiler works independently of the rules, since it only finds a match between the rules the robot knows and the instructions from the semantic reasoner. If no match is found, it announces the unknown instruction and awaits new instructions.

### F. Action Grounding

While the Goal Compiler provides the planner with the goals to be achieved, to achieve those goals the robot motor experience needs to be considered. The Action Grounding serves this function, matching the repertoire of robot rules with the learned affordances of the objects around.

The outcome of the grounding is a list of possible actions, given robot knowledge and what surrounds it. For each rule, the grounding module will find the possible objects match, creating several possible grounded actions, e.g., *grasp__obj_with__hand* becomes *grasp_spoon_with_left*, *grasp_spoon_with_right*, *grasp_tomato_with_left*, etc. Then, for each of these actions, we predict the effects by querying the Bayesian Network: the effects are encoded as probability distributions, and therefore the actions become probabilistic rules that can be used by the planner. The list of possible outcomes is now associated to their respective probability:

| Grounded action example |
|---|
| ACTION: |
| grasp_spoon_with_left |
| CONTEXT: |
| ¬spoon_ishand left_clearhand left_ishand |
| spoon_isreachable_with_left ¬ALL_on_spoon |
| OUTCOMES: |
| 0.85 spoon_inhand_left ¬left_clearhand |
| 0.15 <noise> |

All actions have a noise outcome, representing unforeseen results from a certain action, and usually with a small probability associated with it. This outcome is used by PRADA [11] in predicting the best next action, as we will see in the next sub-section. We use this outcome to represent the failure of the action for simplification, but more outcomes can be encoded if needed. The rules can be changed and adapted to any problem, as long as the structure is kept as above (Action; Context; Outcome).

### G. Planner

The objective of this architecture is the execution of a task specified by a human in the form of a verbal instruction. In order to achieve this, the robot has to react to changes in the environment, and problems not foreseen by the semantic reasoner. In this paper we propose a planning cycle, where in each loop the robot will update its perception of the world, check to see if the goal has been met, plan the next action using PRADA [11], and execute the planned action. The introduction of this method allows the implementation of some *heuristics*, as well as the exploitation of some features of the previous modules. We have labeled these features into three different categories:

1. **Goal Consistency Check**. Since all the sub-goals are stored in memory, it is then possible to manage them in order to fix some problem, like outside interference. At each planning step, a consistency check is performed on the symbols that are common to the current and previous sub-goals. If one of those symbols fails the check, the planner backtracks on its sub-goal list until this check passes, and re-plans from there.

2. **Adaptability**. If an action fails, and nothing changes in the world state, the robot would keep trying to use that action endlessly. Since the grounded actions are independent from each other, we can *adapt* an action that fails by lowering its probability of success and increasing its <noise>, making its outcome less certain, and as such, less useful for planning, forcing the robot to try a different approach.

3. **Creativity**. When no action is found for a certain sub-goal, and the planning horizon is already too large, the planner jumps to the next sub-goal, and tries for that instead. This is possible because all relevant information is present in the final goal, and no information is lost in jumping sub-goals. This can solve some problems with action-specific sub-goals, like having an object on a specific hand, e.g., *spoon_inhand_left*.

## IV. Examples and Results

The proposed planning architecture is not only able to solve the geometric problems not generally considered in natural language instructions, but also deal with other situations, like external and robot-related problems. To showcase these capabilities we report here experiments in which an iCub robot [22] is instructed to make a sandwich, with cheese, tomato and bread. These objects are on top of a table, together with two tools: a rake and a stick.

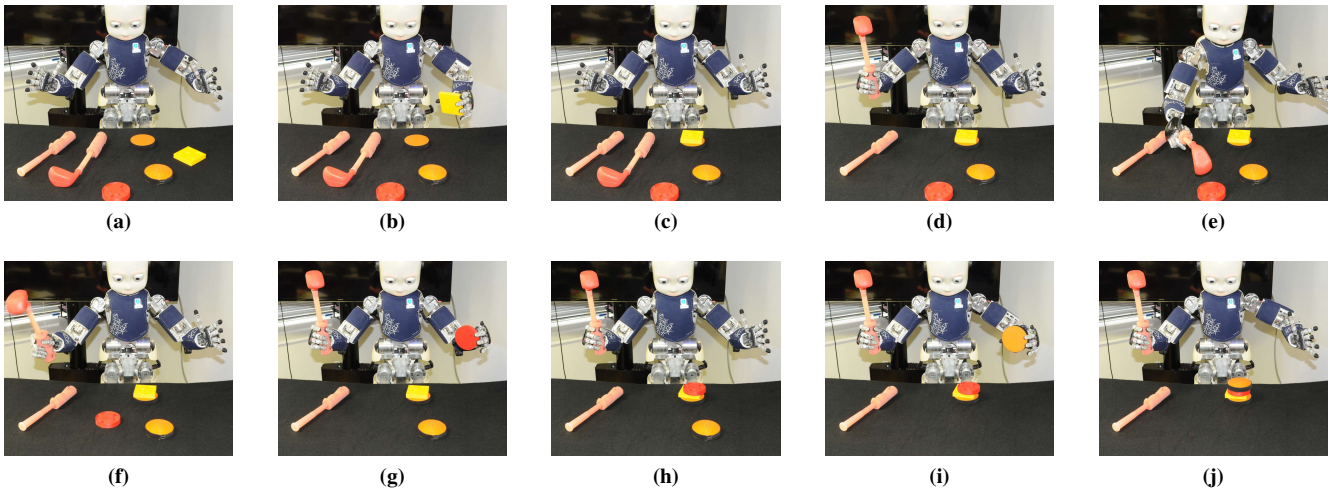



**Fig. 3:** Temporal snapshots of the robot during the first example (Object Out of Reach), see Sec. IV-A. Best seen in color.

### A. First Example: Object Out of Reach

In the first experiment, shown in Fig. 3, the tomato is far from robot reach, but reachable by the tools. The instructions provided by the semantic reasoner are: "hand grasp cheese; cheese reach bun-bottom; hand put cheese; hand grasp tomato; tomato reach cheese; hand put tomato; hand grasp bun-top; bun-top reach tomato; hand put bun-top". These are translated by the Goal Compiler into a list of sub-goals, where the final goal is *cheese_on_bun-bottom; tomato_on_cheese; bun-top_on_tomato*. The initial world state can be seen on Fig. 3a.

The first two actions are the same as in the instructions sent by the semantic reasoner: *grasp_cheese_with_left; put_cheese_on_bun-bottom_with_left* (Figs. 3b–3c). However, when the robot tries to grasp the tomato, it detects that the tomato is out of reach. The probabilistic planner then plans a way to grasp the tomato. From the affordances database (see Sec. III-B), it knows that the action of pulling the tomato with a rake has a higher success probability than pulling it with the stick, and it plans the following sequence: *grasp_rake_with_right; pull_tomato_with_rake_on_right (Fig. 3d–3f); grasp_tomato_with_left*. With this sub-goal met (Fig. 3g), it continues the planning cycle (explained in Sec. III-G) until it gets to the final goal, shown in Figs. 3h–3j.

### B. Second Example: Sabotaged Plan

In the previous example, the robot is considered to be the only agent able to act on the surrounding environment. In most situations, however, and in the case of human-robot shared tasks in particular, this is not true.

In this second example, shown in Fig. 4, a human *sabotages* the robot plan execution, by removing some ingredients from the sandwich when the robot is almost ready to complete the recipe plan, already with the bun-top in its hand. This external disturbance can be seen in Figs. 4a–4b. When the world state is updated, the Goal Consistency Check fails for *tomato_on_cheese*. This check fails two more times, jumping back on the sub-goal list for each failure, until the sub-goal is *cheese_on_bun-bottom*. This time, the check passes, and the planning cycle is resumed. The resulting action sequence is: *grasp_cheese_with_right; put_cheese_on_bun-bottom_with_right*, and so on, leading to the final goal shown in Fig. 4e.

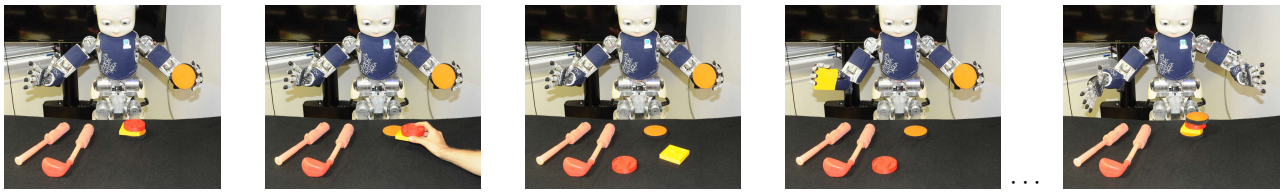
This example shows how the Goal Consistency Check heuristic (see Sec. III-G) helps the planner in solving complex problems, by receding in the plan to fix them first, instead of trying to achieve the final goal despite the mistake.

### C. Third Example: Action Failure

In the previous example, the robot solved a sabotage problem, but sometimes the robot itself might be the problem. In this example, the robot has a faulty left hand, making it impossible for the robot to grasp with it.

Since the first sub-goal requires the robot to have *cheese_inhand_left*, and since this is an impossible goal, the robot would be stuck here forever, retrying *grasp_cheese_with_left* indefinitely. Due to the Adaptability heuristic (see Sec. III-G), each failure reduces the probability of success of that action. However, with no alternative action to achieve this sub-goal, the planner would still fail. This problem is solved by the Creativity heuristic. In this particular case, it jumps the *cheese_inhand_left* sub-goal, and instead tries to achieve *cheese_on_bun-bottom*. With the probability of success of the *grasp_cheese_with_left* action so low, the robot chooses the right hand instead: *grasp_cheese_with_right; put_cheese_on_bun-bottom_with_right*.

The robot uses a different approach to achieve the sub-goal compared to the natural language instructions it received, showing some creativity in fixing the problems it faces. There is a limit, however, as to how many sub-goals the robot can jump, since given the probabilistic nature of the planner, if too many steps are considered, the planner might not find a solution. The number of steps that can be considered depends on the probability of effects of the actions, but in this case



(a) (b) (c) (d) (e)

**Fig. 4:** Temporal snapshots of the robot during the second example (Sabotaged Plan), see Sec. IV-B. Best seen in color.

(for a probability of success around 0.80) we found the planner could find a solution for a problem 5 steps deep.

### D. Fourth Example: Missing Objects

In some situations, due to robot or external influence, some objects might go missing. As long as this object is not a part of the final goal (e.g.: rake), the planner can still find a solution, otherwise it reports the failure back to the high-level planning, and awaits new instructions.

## V. Conclusions and Future Work

We presented a general cognitive architecture to interpret verbal instructions that involve motor tasks, and translate them into robot actions, adapted to the robot sensorimotor capabilities and to the contingencies of the current state of the environment. We provide a number of real world examples in which the humanoid robot iCub successfully executes complex motor tasks requested by a human user through spoken language, showing flexible and creative behaviours that allow to cope with i) details of the motor execution not specified by the human user, ii) unexpected events caused by external interventions, and iii) action failures.

These robot behaviours are made possible by the integration of different components: the definition of the goals in robot-symbols (Goal Compiler), the inclusion of the robot prior sensorimotor knowledge in the plans (Actions Grounding through the Affordances, for the Probabilistic Planner), the continuous monitoring of the state of the world (World State). The three main modules (Goal Compiler, Action Grounding and Planner) can be used with any robot, provided the information is structured as described in Sec. III (rules, symbols, instructions sequence). Moreover, the use of a hierarchical reasoning process (high–mid–low levels) allows to solve large, complex problems, by transforming them into several smaller problems, each one with its own small plan. All the software that implements the proposed architecture is publicly available (`https://github.com/robotology/poeticon`).

Currently, symbols and rules are defined by the programmer; a possible route for future work would be to have the robot learning the symbols involved with each rule automatically.

## Acknowledgments

This work was partially supported by the Portuguese Government (Fundação para a Ciência e a Tecnologia, UID/EEA/50009/2013) and by the European Commission under projects POETICON++ (FP7-ICT-288382) and LIMOMAN (PIEF-GA-2013-628315).

## References

[1] K. Pastra, P. Dimitrakis, E. Balta, and G. Karakatsiotis. PRAXICON and its language-related modules. In *SETN*, 2010.
[2] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *IROS*, 2010.
[3] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta. Exploring affordances and tool use on the icub. In *Humanoids*, 2013.
[4] T. Lozano-Pérez, J. L. Jones, E. Mazer, P. A. O'Donnell, W. E. L. Grimson, P. Tournassoud, and A. Lanusse. HANDEY: A Robot System that Recognizes, Plans, and Manipulates. In *ICRA*, 1987.
[5] J. Wolfe, B. Marthi, and S. J. Russell. Combined Task and Motion Planning for Mobile Manipulation. In *ICAPS*, 2010.
[6] R. P. Goldman. A Semantics for HTN Methods. In *ICAPS*, 2009.
[7] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical Planning in the Now. In *ICRA*, 2011.
[8] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *ICRA*, 2009.
[9] I. R. Nourbakhsh. Using Astraction to Interleave Planning and Execution. In *Third Biannual World Automation Congress*, 1998.
[10] T. Lozano-Pérez and L. P. Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *IROS*, 2014.
[11] T. Lang and M. Toussaint. Planning with Noisy Probabilistic Relational Rules. *Journal of Artificial Intelligence Research*, 39, 2010.
[12] A. Gonçalves, J. Abrantes, G. Saponaro, L. Jamone, and A. Bernardino. Learning Intermediate Object Affordances: Towards the Development of a Tool Concept. In *ICDL-EpiRob*, 2014.
[13] L. Montesano, M. Lopes, A. Bernardino, , and J. Santos-Victor. Learning Object Affordances: From Sensory–Motor Coordination to Imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.
[14] A. Stoytchev. Learning the affordances of tools using a behavior-grounded approach. *Springer LNAI*, pages 140–158, 2008.
[15] A. Gonçalves, G. Saponaro, L. Jamone, and A. Bernardino. Learning Visual Affordances of Objects and Tools through Autonomous Robot Exploration. In *ICARSC*, 2014.
[16] N. Krüger et al. Object–action complexes: Grounded abstractions of sensory-motor processes. *Robotics and Autonomous Systems*, 59(10), 2011.
[17] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *ICRA*, 2012.
[18] E. Ugur and J. Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *ICRA*, 2015.
[19] E. Ugur and J. Piater. Refining discovered symbols with multi-step interaction experience. In *Humanoids*, 2015.
[20] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
[21] J. J. Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, USA, 1986. Original work published in 1979.
[22] G. Metta et al. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, 2010.