Robotics and Autonomous Systems 166 (2023) 104452

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Symbolic representation of what robots are taught in one demonstration

Andrea Maria Zanchettin

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo Da Vinci 32, Milano, Italy

ARTICLE INFO

Article history:
Available online 22 May 2023

Keywords:
Programming by demonstration
Semantic skill representation
Semantic scene description

ABSTRACT

To facilitate the use of robots in small and medium-sized enterprises (SMEs), they have to be easily and quickly deployed by non-expert users. Programming by Demonstration (PbD) is considered a fast and intuitive approach to handle this requirement. However, one of the major drawbacks of pure PbD is that it may suffer from poor generalisation capabilities, as it is mainly capable of motion-level representations. This work proposes a method to semantically represent a demonstrated skill, so as to identify the elements of the workspace that are relevant for the characterisation of the skill itself, as well as its preconditions and effects. This way, the robot can automatically abstract from the demonstration and memorise the skill in a more general way. An experimental case study consisting in a manipulation task is reported to validate the approach.

1. Introduction

When it comes to instructing robots to execute certain operations, different methods exist. The first adopted solution was to program their motions through sets of machine-readable instructions [1,2]. Together with the widespread use of robots, especially in automotive industries, languages have become more and more powerful by adding new functionalities, thus allowing robots to be employed in an increasing variety of tasks. In face of this growing complexity, researchers proposed to organise code into libraries containing atomic, composable, and reusable parts. These atomic programs are usually referred to as *skills* [3–5]. According to [5], skills are atomic functionalities each one qualified by a set of pre- and postconditions. The use of pre- and postconditions allows the robot to have access to a formal description of how the skill will modify the state of the world, as well as when the state of the world allows the skill itself to be executed.

Nowadays, researchers rather prefer to program robots by physically demonstrating trajectories that are then encoded into a set of parameters, see [6] for a review. This trend has been further stimulated by the availability on the market of a new generation of robotics platforms allowed to be manually guided by a relatively non-expert technician [7].

Regardless the way robots have been instructed, they surely miss the awareness and the understanding of what they are taught. It follows that robots will not be able to *autonomously compose skills* anytime soon. The main reason is to be found in the way programs are memorised in robotics controllers. From the one hand, instructions have clear meanings for robotic programmers, thanks to their mnemonic names. On the other hand, robots are surely not able to automatically compose them, essentially because programs or instructions are not represented in a way that allows comprehension by the robot. Furthermore, while being intuitive and fast, Programming by Demonstration (PbD) does not allow the agent being thought to represent semantic knowledge of the underlying skill being demonstrated, thus preventing both the composition and the generalisation in other contexts.

The term *semantic memory* has been first proposed by Quillian in his doctoral dissertation [8]. According to Endel Tulving semantic memory is the

> *"[...] organised knowledge a person possesses about words and other verbal symbols, their meaning and references, about relations among them, and about rules, formulas, and algorithms for the manipulation of these symbols, concepts, and relations."*, [9].

Despite the use of semantics is definitely not a novel research field in computer science, recent works from the robotics community are gradually demonstrating the benefits of its adoption, especially for complex tasks. The combined use of sensorimotor information and exteroceptive perception has been proposed in [10] and then exploited in other works. For example in [11], the authors collect several demonstration to reinforce the model of preconditions and effects the robot has regarding the skill it has been taught. In [12] semantic annotations provided by the programmer through natural language are used for grounding the data acquired in one-shot kinaesthetic demonstrations. The

E-mail address: andreamaria.zanchettin@polimi.it.

https://doi.org/10.1016/j.robot.2023.104452

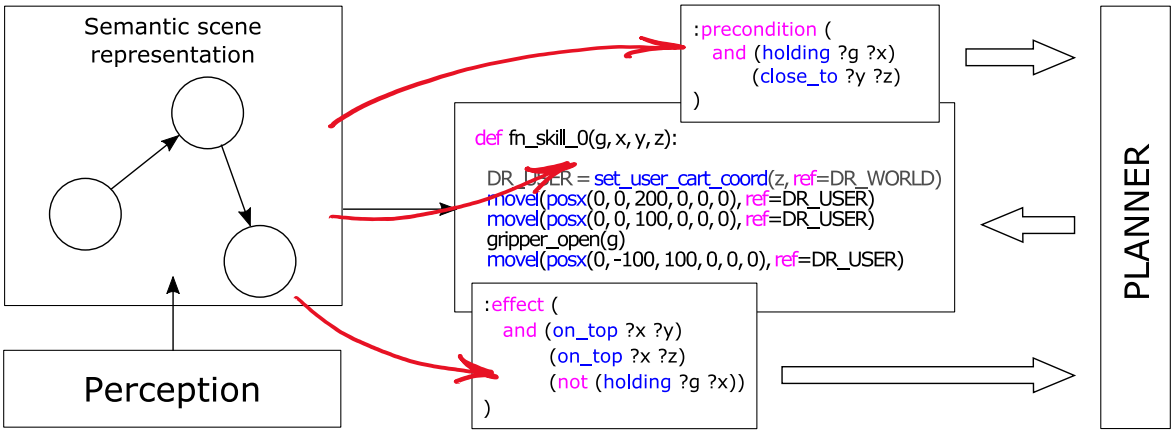

Fig. 1. Pictorial representation of the approach described in this paper: during a demonstration, a scene understanding engine is fed with the output of perception and derives the executable code describing a skill, as well as semantic annotations, specifying preconditions and effects.

problem of modelling the spatial relationships between objects to better understand the meaning of manipulation actions is presented in [13]. Despite the work is primarily focused on human actions, the approach is also relevant for robots to understand the meaning of what they are taught from demonstration, [14]. A similar approach has been also proposed by Ramirez-Amaro et al. in [15] to transfer the semantic representation of human activities to a humanoid robot. This method has been further exploited in [16] to automatically generate planning operators. The authors of [17] propose to combine DMPs (Dynamic Movement Primitives, [18]) with an attentional system that is based on a long term memory module, containing the procedural knowledge of the robot. In [19] an approach that combines PbD and semantics is presented. The authors developed an algorithm, named *semantic skill recogniser*, that classifies the demonstrated trajectory into a proper skill within a predefined set, as the skill portfolio in [5]. The method proposed in [20] adopts a time series of relational graphs to model the effects of the demonstrated skill on the environment. Semantic features are then extracted among those with high degree of invariance between different demonstrations. Finally, the work described in [21] presents a generative model of static and dynamic 3D spatial relations between multiple objects which is then use in combination with natural language to instruct a humanoid robot.

Research works focused on the characterisation of skills from demonstrations are particularly interesting for this work. The main challenges in learning actions models through demonstrations is that the robot is only provided with positive examples [11,22].

Once semantic grounding is available in the description of a basic skill, the program can be automatically constructed through compositions, as described in [23,24]. For example, Planning Domain Definition Language (PDDL), [25], can be used to specify the initial and goal configurations (in [24] they are derived from visual data), and to plan the sequence of skills to achieve the goal, using e.g. [26,27]. Finally, in [28] a neuro-symbolic task planner is fed with inputs coming from a two-level abstraction: the geometric scene graph spatial poses of objects, and a symbolic scene graph that represents topological relations among the perceived objects.

This paper addresses the problem of generating modular and executable motion instructions from kinaesthetic demonstrations, allowing the robot to autonomously combine these parts in goal-driven tasks. Semantic annotations are automatically generated for a skill that is taught by demonstration. The meaning associated to a certain skill is then memorised in order to allow the robot to autonomously reuse such a skill in similar or

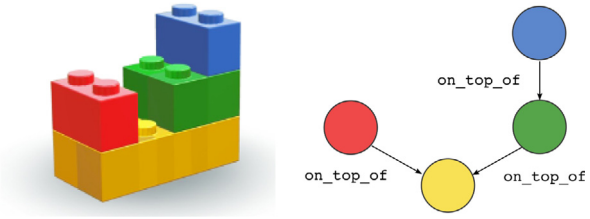

Fig. 2. Example of an image and the corresponding Semantic Network.

even different tasks. The approach proposes the adoption of a dynamic *Semantic Network* that is constantly updated throughout the demonstration. Upon the completion of a demonstration, the method automatically generates executable code for future utilisation of the skill by the robot. In particular, the approach (sketched in Fig. 1) is capable of

1. providing a basis for the semantic representation of preconditions and effects of a demonstrated skill, for the automatic composition by e.g. automatic planners;
2. identifying the elements of the scene that are relevant for the characterisation of the skill, thus to abstract from the (grounded) demonstration;
3. producing executable and reusable code from a single demonstration.

2. Methodology

The first mathematical tools adopted to represent relations between concepts, and therefore their semantic relationships, are the so-called *Semantic Networks* or *Labelled Graphs*, see [29]. Semantic Networks are essentially graphs consisting of nodes, denoting objects or concepts, directed arcs, denoting the existence of relationship between them, and arc labels that further specify these relations. As an illustrative example, Fig. 2 reports an image, together with the corresponding Semantic Network.

In the context of programming by demonstration, the following definition of Semantic Network is considered.

Definition 1 (*Semantic Network*)**.** A Semantic Network (SN) is a dynamic oriented graph $G_k = \langle N_k, A_k, f_k \rangle$, where, at every discrete time instant k, N_k is the set of nodes, $A_k \subseteq N_k^2$ is the set of arcs, and f_k is a function $f_k : A_k \rightarrow L$ that defines labels, from a predefined set L, associated to a certain arc.

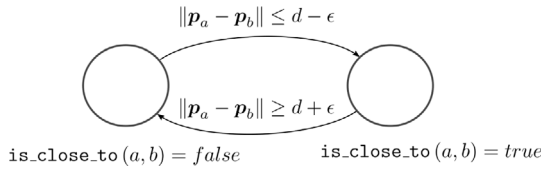

Fig. 3. Example of the dynamics to be adopted to update the grounding of the predicate `is_close_to`(a, b).

We further introduce the set

$$\Pi_k = \{\langle \eta, \mu, f\left((\eta, \mu)\right) | \eta, \mu \in N_k, (\eta, \mu) \in A_k \rangle\}$$

as the set of triplets in the SN at discrete time k.

The SN just introduced is a generalisation of the Semantic Networks described in [29], allowing for both spatial, and temporal inference on the state of the workspace. In fact, nodes, arcs and their labels are not necessarily static, as they might change according to the dynamics of the environment.

Definition 2 (*Update Function*)**.** At any discrete time instant k, the network G_k is updated through a suitable *Update function* $h : G \times \mathcal{Y} \rightarrow G$, that maps the current network G_k to the one at the next time instant G_{k+1} based on current sensing data $y_k \in \mathcal{Y}$, i.e. $G_{k+1} = h\left(G_k, y_k\right)$.

The evolution of the workspace can be then described by means of the dynamical system $G_{k+1} = h\left(G_k, y_k\right)$. At any time instant, the arcs present in the SN can be used for the grounding of predicates. As an example, consider the relationship between two objects a and b. The automaton in Fig. 3 represents the update function $h\left(\cdot, \cdot\right)$ that can be adopted to modify the Semantic Network G_k based on sensed data y_k, thus to derive G_{k+1}.

With the aim of characterising how a robotic agent is capable of applying changes to the environment, we first need to introduce the concept of skill.

Definition 3 (*Skill*)**.** A skill is an atomic time-limited action that an intelligent agent can perform. The skill has unique entry and exit points, at least in case of success, and cannot be interrupted in favour of another skill.

In order to semantically characterise the skill from one demonstration, three major issues arise: (a) how to characterise the effects on the environment to be attributed to the execution of the skill, (b) how to extract preconditions of the skill, and, (c) which reference frame is the most suitable to abstract the demonstrated motion. The three problems are further addressed in the following.

The first two are needed in order to allow the robot to autonomously reuse the skills taught by demonstration, [30], while the latter is essential to correctly report the demonstrated motion to the most suitable reference frame.

2.1. Representing the effects of a skill

The first problem to be addressed is to let the robotic agent represent the effects on the environment caused by the execution of a certain skill. Therefore, consider the internal representations of the environment the robotic agent has, at discrete time k, before starting the demonstration of a certain skill s, say G_k, and after its completion, $G_{k+\lambda}, \lambda > 0$, see Fig. 4 for an example. Furthermore, define the two sets

$$\Pi^{+}_{k+\lambda|k,s} = \{\tau | \in \Pi_{k+\lambda}, \notin \Pi_k\}$$

$$\Pi^{-}_{k+\lambda|k,s} = \{\tau | \tau \in \Pi_k, \tau \notin \Pi_{k+\lambda}\}$$

Table 1
Predicates changing their values throughout the demonstration of the skill will be either in $\Pi^{-}_{k+\lambda|k,s}$ or in $\Pi^{+}_{k+\lambda|k,s}$, $\Pi^{=}_{k+\lambda|k,s}$ in turn will contain the invariants.

		$G_{k+\lambda}$	
		$\tau \in \Pi_{k+\lambda}$	$\tau \notin \Pi_{k+\lambda}$
G_k	$\tau \in \Pi_k$	$\tau \in \Pi^{=}_{k+\lambda|k,s}$	$\tau \in \Pi^{-}_{k+\lambda|k,s}$
	$\tau \notin \Pi_k$	$\tau \in \Pi^{+}_{k+\lambda|k,s}$	–

The two sets $\Pi^{+}_{k+\lambda|k,s}$ and $\Pi^{-}_{k+\lambda|k,s}$ contain the triplets that are in $\Pi_{k+\lambda}$ (i.e. in the SN $G_{k+\lambda}$) and were not in Π_k (i.e. in the SN G_k), and those that were in Π_k but are no longer in $\Pi_{k+\lambda}$, respectively. For completeness, we also introduce $\Pi^{=}_{k+\lambda|k,s}$ which contains the triplets that were both in G_k and in $G_{k+\lambda}$, i.e.

$$\Pi^{=}_{k+\lambda|k,s} = \{\tau | \tau \in \Pi_k, \tau \in \Pi_{k+\lambda}\}$$

Notice that the three sets Π^{+}, Π^{-}, and $\Pi^{=}$ form a partition of all the available triplets (see Table 1).

Consider again the example of Fig. 4, the three sets are

$$\Pi^{+}_{k+\lambda|k,s} = \left\{\langle n_5, n_4, \texttt{on\_top}\rangle, \langle n_5, n_7, \texttt{on\_top}\rangle\right\}$$
$$\Pi^{-}_{k+\lambda|k,s} = \left\{\langle n_3, n_5, \texttt{holding}\rangle\right\},$$

and

$$\Pi^{=}_{k+\lambda|k,s} = \{\langle n_1, n_2, \texttt{on\_top}\rangle, \langle n_7, n_4, \texttt{close\_to}\rangle, \langle n_6, n_3, \texttt{has}\rangle\},$$

respectively. In this case, the robotic agent may easily infer that executing skill s would result in having placed the green cylinder n_5 on top of both the yellow cube n_4 and the magenta cube n_7 (as per the two triplets contained in $\Pi^{+}_{k+\lambda|k,s}$), while the gripper n_3 will be no longer holding the cylinder n_5 (triplet in $\Pi^{-}_{k+\lambda|k,s}$).

It should be clear that the two sets $\Pi^{+}_{k+\lambda|k,s}$ and $\Pi^{-}_{k+\lambda|k,s}$ can be used to characterise the effects of skill s, playing the role of the *add list* and of the *delete list* of STRIPS (STanford Research Institute Problem Solver, [26]), respectively.

Back to the example of stacking the green cylinder in Fig. 4, after one demonstration of the skill, the corresponding effect would be represented as follows

```
:effect (
 and (on_top ?x ?y)
     (on_top ?x ?z)
     (not (holding ?g ?x))
)
```

With the aim of generalising any further autonomous execution of the skill, notice that the effect is referred to generic nodes `g`, `x`, `y`, and `z` (specific types, i.e. `g` begin a gripper, `x` being a cylindrical object, `y` and `z` being cubes will be specified within the precondition as it will discussed next).

As the environment is open, i.e. the robotic agent is not the only one capable of modifying it, the triplets contained in the two sets are not necessarily *caused by* the execution of the skill s. Unfortunately, the only way to refine the knowledge of the effects a certain skill produces on the environment is to rely on additional information, coming from either new demonstrations, from human explicit inputs [12], or as an output of a learning strategy run throughout the autonomous execution of the skill by the robot, [31,32].

2.2. Representing the preconditions of a skill

Differently from effects, preconditions are properties of the environment that must hold true prior to the execution of the skill, since only in this case the skill itself might produce the

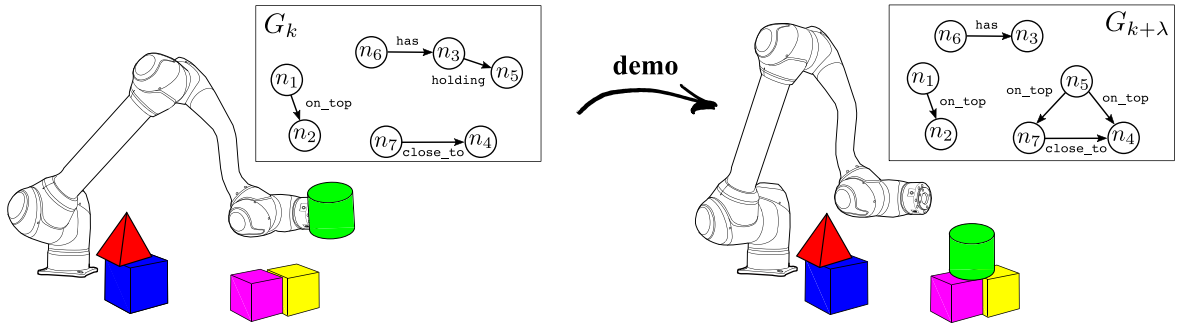

Fig. 4. Example of demonstration of a skill together with the SN before (G_k) and after ($G_{k+\lambda}$) the demonstration. Node n_1 represents the red pyramid, n_2 the blue cube, n_3 the gripper, n_4 the yellow cube, n_5 the green cylinder, n_6 the robot, while node n_7 represents the magenta cube. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

desired effects. Understanding whether or not the current state of the environment allows the robot to execute a certain skill can be seen as a classification problem to be learnt [22]. However, the main issue in learning a model of preconditions through demonstrations is that the robot is only provided with positive examples, i.e. successful demonstrations of skills [11]. Facing this intrinsic limitation of learning from demonstration would require interaction with the demonstrator, either directly, [11,12,33], or indirectly through negative examples to learn from, [22], or in simulation [34]. Finally, it is worth reminding that based on the assumption of an open world, approaches like skill chaining, [35,36], are not practical e.g. in collaborative executions or in case of demonstrations taken place without a meaningful order, as they assume the environment can be only modified by one agent.

With the aim of extracting the largest amount of cues from a single demonstration, without requiring a direct interaction with the teacher, a first guess regarding possible preconditions of a skill taught by demonstration is contained as triplets in the set $\Pi^{-}_{k+\lambda|k,s}$, i.e. triplets that were in Π_k but are no longer in $\Pi_{k+\lambda}$. The rationale behind this attempt is straightforward. If the execution of the skill has the effect of invalidating a certain relationship between objects in the scene (i.e. putting a triplet in the delete list), such a relationship should be present before the execution. In the example of Fig. 4, the set $\Pi^{-}_{k+\lambda|k,s}$ was represented by the single triplet $\langle n_3, n_5, \texttt{holding}\rangle$, which clearly constitutes a precondition for the execution of the skill.

There are conditions, i.e. triplets, that *have* to be considered as preconditions though their grounding is not effected by the execution. In the example of Fig. 4, the vicinity of the yellow and magenta cubes has to be somehow represented within the preconditions. A possible way to handle this case consists in using $\Pi^{=}_{k+\lambda|k,s}$, i.e. the set of (invariant) triplets that are present *both* at the beginning and at the end of the demonstration. However, without further pruning of unnecessary conditions, this would lead to an over-specified set of preconditions. In fact, while triplet $\langle n_6, n_3, \texttt{has}\rangle$ would be now considered as a precondition, the same would apply to $\langle n_1, n_2, \texttt{on\_top}\rangle$, limiting the possibility to execute the demonstrated (stacking a cylinder on top of two cubes) skill only in situations where a pyramid is stacked onto a cube. To avoid these possibilities, we define a set of nodes, called *nodes of interest* (*NoI*), as the nodes that appear in some triplets of $\Pi^{+}_{k+\lambda|k,s}$ or $\Pi^{-}_{k+\lambda|k,s}$, i.e.

$$NoI = \{\ \eta \,|\, \langle \eta, \mu, f(\eta,\mu)\rangle \in \Pi^{+}_{k+\lambda|k,s} \cup \Pi^{-}_{k+\lambda|k,s} \vee \langle \mu, \eta, f(\mu,\eta)\rangle \in \Pi^{+}_{k+\lambda|k,s} \cup \Pi^{-}_{k+\lambda|k,s}\ \}$$

Based on this definition, only triplets in $\Pi^{=}_{k+\lambda|k,s}$ referring to elements in *NoI*, say $\epsilon_{NoI}\left(\Pi^{=}_{k+\lambda|k,s}\right)$, will be considered as preconditions. In other words, if $\langle \eta, \mu, f(\eta,\mu)\rangle$ and $\langle \eta, \nu, f(\eta,\nu)\rangle$ are either in $\Pi^{-}_{k+\lambda|k,s}$ or $\Pi^{+}_{k+\lambda|k,s}$, then $\langle \mu, \nu, f(\mu,\nu)\rangle \in \Pi^{=}_{k+\lambda|k,s}$ will be listed as precondition.

With reference again to the example of Fig. 4, as already previously mentioned the set $\Pi^{=}_{k+\lambda|k,s}$ contains $\langle n_6, n_3, \texttt{has}\rangle$ as well as $\langle n_1, n_2, \texttt{on\_top}\rangle$, $\langle n_7, n_4, \texttt{close\_to}\rangle$. While the first two triplets will not be considered as preconditions, the latter will, in fact

$$NoI = \{n_3, n_4, n_5, n_7\}$$

does not contain n_6 and n_3 nor n_1 and n_2, but contains *both* n_7 and n_4. Hence, after one demonstration, the preconditions of the skill will be modelled as follows

```
:paramaters (?g ?x ?y ?z)
:precondition (
 and (is_a_gripper ?g)
      (is_a_cylinder ?x)
      (is_a_cube ?y)
      (is_a_cube ?z)
      (close_to ?y ?z)
      (holding ?g ?x)
)
```

Notice that, though the skill has been demonstrated on specific instances of objects (the green cylinder, the yellow and the magenta cubes), for generalisation, the skill is represented with respect to generic classes of objects `x`, `y`, and `z`, etc., with the preconditions of `x` being a cylinder, `y` and `z` being cubes, etc.

2.3. Representing skills taught by demonstration in suitable coordinates

After a skill has been demonstrated once, the corresponding motion has to be encoded into a reduced number of parameters. The next problem to face regards the identification of the object the skill refers to, and consequently selection of the most suitable coordinate system to represent the demonstrated motion. Before proceeding, we assume that, at every discrete time instant k, each node $n \in N_k$ of the SN is further specified by its pose ${}^{w}\boldsymbol{T}_{n_k} \in SE(3)$ relative to the world coordinate system. One first issue regards the space where the skill has to be represented, i.e. either the joint/actuation space, or the operational space. The natural choice is towards the operational space, so as to decouple the particular robot kinematic structure from the representation of the skill. We then assume that the raw demonstration of a skill s is preliminarily stored in terms of the state of the gripper $\sigma_{g_i} \in \{0, 1\}$ (e.g. open or closed) as well as by a set of waypoints in ${}^{w}\boldsymbol{T}_{g_i} \in SE(3)$ specified in the world coordinate system:

$$\mathcal{D}_s = \left\{\left(\sigma_{g_i}, {}^{w}\boldsymbol{T}_{g_i}\right) | k \le i \le k+\lambda,\ \sigma_{g_i} \in \{0,1\},\ {}^{w}\boldsymbol{T}_{g_i} \in SE(3)\right\}$$

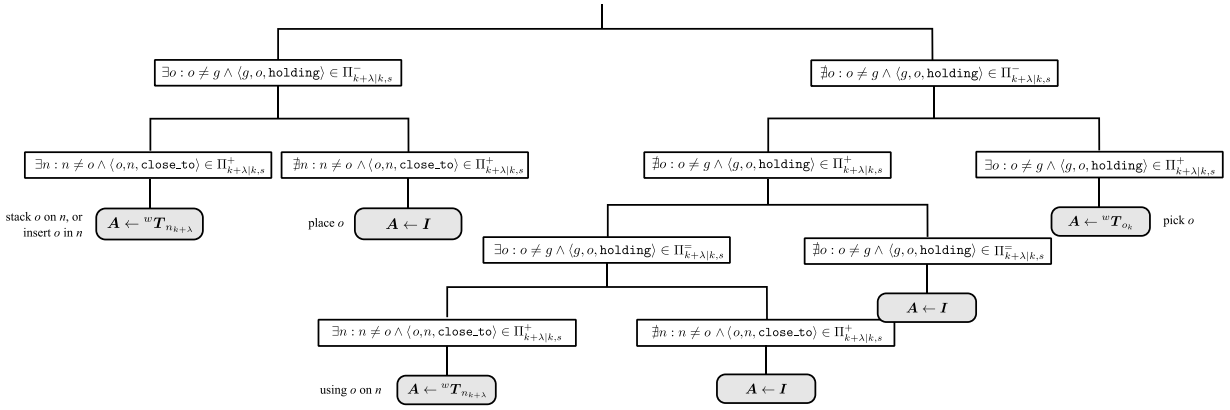

Fig. 5. Decision-tree for the frame selection procedure: it returns the most suitable reference frame $\boldsymbol{A} \in SE(3)$ to represent the waypoints ${}^w\boldsymbol{T}_{g_i}$ that are collected in the world reference frame (here predicate `close_to` is representative of any possible spatial relation symbol [13], as e.g. `on_top`, `next_to`, etc.).

where subscript g stands for the node representing the gripper in the SN, i is a discrete time instant, while w is the world frame.

Since the world coordinate system is not necessarily the best one to allow for the reusability of the skill, a more convenient frame to represent the waypoints in $\mathcal{D}_s$ has to be selected. The following options are available for the transformation in a suitable reference frame of a generic waypoint ${}^w\boldsymbol{T}_{g_i}$ captured at discrete time instant i:

(a) the pose of a generic node n of the SN at time instant $k+\lambda$ (${}^w\boldsymbol{T}_{n_{k+\lambda}}$), i.e. at the end of the demonstration, or
(b) the pose of a generic node n of the SN at time instant k (${}^w\boldsymbol{T}_{n_k}$), i.e. immediately before the demonstration.

The problem at hand actually entails two subproblems: (1) which node n has to be accounted as relevant for the skill, and (2) at which time instant, either k or $k+\lambda$ the position of node n has to be considered.

As already stated, the execution of a skill s can be characterised in terms of the two sets $\Pi^+_{k+\lambda|k,s}$ and $\Pi^-_{k+\lambda|k,s}$, containing triplets that deserve attention, as they are somehow related to the skills. Therefore, it is natural to select the reference node n among those listed in one of the two sets. This approach differs substantially from the one described in [19] where all possible pairs of nodes are considered, while somehow recalls the attentional system presented in [17].

The approach adopted in this work is inspired by the classification of human actions introduced in [15] which is based on the human hand, the tool, and the object the tool is possibly acting on.

Assume the gripper is holding an object o at the beginning of the demonstration. Then, if the same object will turn out to be no longer held by the robot but close enough to any other object n at the end of the skill, then the node n and its pose at the end of the skill is considered as significant for the execution of the skill itself, like in "stack o on n" or "insert o in n". The waypoints ${}^w\boldsymbol{T}_{g_i}$ will be then referred to frame ${}^w\boldsymbol{T}_{n_{k+\lambda}}$ (as in option (a)). Otherwise, all waypoints are referred to world frame, like in "place o" (option (a) or option (b), as the world frame does not move).

In turn, if the demonstration terminates with the robot holding object o not held at the beginning, the pose of the same object o at the beginning of the execution is considered (as in "grasp o"), and the waypoints in $\mathcal{D}_s$ will be referred to ${}^w\boldsymbol{T}_{o_k}$ (as in option (b)). Finally, if none of the options above apply, the skill either begins and terminates with the gripper holding an object o, or no object is interacting with the gripper all along the demonstration. In the former case, if the object o is close to another object n at the end of the skill, like in "using o on n", the waypoints ${}^w\boldsymbol{T}_{g_i}$ will be then referred to frame ${}^w\boldsymbol{T}_{n_{k+\lambda}}$ (as in option (a)), or to world frame otherwise. In the latter case, no specific meaning for this skill can be inferred, and all waypoints are referred to world frame.

The overall procedure is formalised in the decision-tree reported in Fig. 5. The outcome of the decision-tree consists in a matrix $\boldsymbol{A} \in SE(3)$ to be used to transform all waypoints in $\mathcal{D}_s$. First, all waypoints are computed in frame $\boldsymbol{A}$, i.e. ${}^\eta\boldsymbol{T}_{g_i} = \left(\boldsymbol{A}^{-1}\right){}^w\boldsymbol{T}_{g_i}, k \leq i \leq k+\lambda$. Algorithm 1 can be finally adopted to generate reusable code (Doosan Robotics Language, DRL, is assumed here) for the execution of the skill. Referring to the example of Fig. 4, i.e. the stacking of the green cylinder on top of the yellow and magenta cubes, the following code will be generated automatically:

```
def fn_skill_0(g, x, y, z):
  DR_USER=set_use_cart_coord(z, ref=DR_WORLD)
  movel(posx(0,0,200,0,0,0), ref=DR_USER)
  movel(posx(0,0,100,0,0,0), ref=DR_USER)
  gripper_open(g)
  movel(posx(0,0,-100,0,0,0), ref=DR_USER)
```

Consistently with the decision-tree in Fig. 5, the motion will be referred to yellow cube `z`, as $\langle g, x, \texttt{holding} \rangle \in \Pi^-_{k+\lambda|k,s}$ and $\langle x, z, \texttt{on\_top} \rangle \in \Pi^+_{k+\lambda|k,s}$.

By collecting preconditions, effects, and robot executable code, one would finally obtain the representation of the demonstrated skill as in Fig. 1.

3. Comparison with related methods and possible limitations

This Section discusses related methods from the state of the art, highlighting the main differences with the approach proposed in this work.

First of all, the approach adopted to describe the scene by means of a Semantic Network (SN) together with the *Update function*, $G_{k+1} = h(G_k, y_k)$, differs substantially from the ones in the literature that propose to update the graph based on measurements only, $G_k = h(y_k)$, [13]. In the example of Fig. 3, a simple hysteretic (hence dynamic) behaviour has been introduced to prevent chattering of the grounding of predicate `is_close_to`(a, b). The presence of this dynamics allows the method to be more robust in face of sensing noise.

The problem of automatically deriving preconditions and effects is addressed in several different ways in the literature, ranging from supervised [22] and unsupervised learning [11,37], to skill chaining [17]. The work that mostly resembles the one described in this paper is the one from Liang et al. [33]. Triplets

Algorithm 1 Pseudocode to derive DRL code for the execution of the demonstrated skill. Notice that waypoints saved by the demonstrator will be referred to a suitable reference frame $\boldsymbol{A} \in SE(3)$ automatically selected according to the decision-tree in Fig. 5.

Require: $\mathcal{D}_s, \boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n, \eta$
out ← "def fn_skill_0("
for each $i \in [1, n-1]$
  out.append ("A_" + i + ", ")
out.append ("A_n) : \n")
if $\eta > 0$ ▷ skill relative to an object in position A_η (will be known online)
  $\boldsymbol{A} = \boldsymbol{A}_\eta$
  out.append (" DR_USER=set_user_cart_coord(A_ + η + ", ref=DR_WORLD)\n")
else
  $\boldsymbol{A} = \boldsymbol{I}$
for each $p \in \mathcal{D}_s$
  $\sigma_{g_i} \leftarrow p[0]$
  ${}^{w}\boldsymbol{T}_{g_i} \leftarrow p[1]$
  ${}^{\eta}\boldsymbol{T}_{g_i} \leftarrow \left(\boldsymbol{A}^{-1}\right) {}^{w}\boldsymbol{T}_{g_i}$
  $e \leftarrow euler\_angles\left({}^{\eta}\boldsymbol{T}_{g_i}\right)$
  out.append (" movel(posx(")
  out.append (${}^{\eta}\boldsymbol{T}_{g_i}[0][3]$ + "," + ${}^{\eta}\boldsymbol{T}_{g_i}[1][3]$ + "," + ${}^{\eta}\boldsymbol{T}_{g_i}[2][3]$ + ",")
  out.append ($e[0]$ + "," + $e[1]$ + "," + $e[2]$)
  if $\eta > 0$
    out.append (") ,ref=DR_USER)\n")
  else
    out.append (") ,ref=DR_WORLD)\n")
  if $\sigma_{g_i} \wedge \neg\sigma_{g_{i-1}}$
    out.append (" gripper_open()\n")
  else if $\neg\sigma_{g_i} \wedge \sigma_{g_{i-1}}$
    out.append (" gripper_close()\n")

before and after the demonstration are compared to select preconditions and effects. However, differently from the present work, in [38] the effects are only characterised by triplets that appear after the demonstration. A similar approach is adopted also in [16], where preconditions and effects are taken from the state of the environment before and after the demonstration, respectively, after being pruned from irrelevant triplets.

The present work proposes to characterise the effects using both $\Pi^{+}_{k+\lambda|k,s}$ and $\Pi^{-}_{k+\lambda|k,s}$. Despite this might seem a marginal difference, consider once again the demonstration reported in Fig. 4. As already mentioned, after the single demonstration, the two sets would be $\Pi^{+}_{k+\lambda|k,s} = \{\langle n_5, n_4, \texttt{on\_top}\rangle, \langle n_5, n_7, \texttt{on\_top}\rangle\}$, and $\Pi^{-}_{k+\lambda|k,s} = \{\langle n_5, n_7, \texttt{holding}\rangle\}$, respectively. According to the method in [38], the effects of the execution of the skill would be characterised only through the triplet in $\Pi^{+}_{k+\lambda|k,s}$, i.e.

```
:effect (
 (on_top ?x ?y)
 (on_top ?x ?z)
)
```

In turn, consistently with the method described in this paper, the effects of the skill will be characterised as follows:

```
:effect (
 and (on_top ?x ?y)
     (on_top ?x ?z)
     (not (holding ?g ?x))
)
```

The proposed characterisation is able to capture the capability of the skill of nullifying (some of) the preconditions. We believe that this feature will be particularly relevant in planning, and especially in those methods using backward skills chaining, [35,36], as the skill will be automatically discarded (thus reducing the branching factor) if the state to be reached is characterised by $\langle n_3, n_5, \texttt{holding}\rangle$.

Moreover, differently from other methods in the literature, the proposed approach is focusing on what is changed in the environment with respect to when the skill is started being demonstrated, without the need of pruning redundant triplets. Thanks to the definition of ϵ_{NoI}, preconditions are extended from considering removed triplets, only. The definition of ϵ_{NoI}, that is used to pick relevant invariant triplets to be regarded as preconditions, can be regarded as the neighbourhood nodes of triplets whose grounding has changed during the demonstration. Notice that, differently from other works using the Euclidean distance or spatial relations, [13,21], between nodes as a vicinity metrics, the definition of ϵ_{NoI} is based on (generic) semantic relationships between nodes.

Finally, some of the methods in the literature, [11,14,17], adopts DMPs to encode the demonstrated motion, while other methods provide directly executable code in the native programming language, [12,33]. In our opinion, the native programming language of the robot and the interpolation capabilities of its controller are the natural choice. To support this claim, consider a skill in which the robot is supposed to align an object to another one. We observed that technicians tend to demonstrate this operation by first bringing the robot to the final position, then retracting the robot along the approach direction to save a waypoint, and eventually bringing back the robot to the final position to save the final point. The use of DMPs in this situation would result in an unnecessary back-and-forth motion, whilst the use of waypoints to be specified by the demonstration would definitely avoid this problem.

As per the selection of the reference frame in which waypoints or trajectories are represented, the object closest to the end-effector is typically adopted, [11,33,38], or alternatively the decision is left to the user [12]. We believe that our approach

Table 2
Summary of the comparison with existing methods in terms of how preconditions and effects are extracted from the demonstration, how motion is interpolated, and which landmark in the scene is used to represent waypoints.

Paper	Preconditions	Effects	Motion	Landmark
Abdo et al. [11]	Based on clustering		DMP, [18]	Closest object
Stenmark et al. [12]	*Not covered*	*Not covered*	Programming language	User defined
Savarimuthu et al. [14]	Learning, from [37]		DMP, [18]	*Not specified*
Diehl et al. [16]	Extracted from Π_k	Extracted from $\Pi_{k+\lambda}$	*Not covered*	*Not covered*
Caccavale et al. [17]	Chaining, similar to [35,36]		DMP, [18]	*Not specified*
Kroemer at al., [22]	Binary classification (random forests)	*Not covered*	*Not covered*	*Not covered*
Liang et al. [33]	$\Pi^{-}_{k+\lambda\|k,s}$	$\Pi^{+}_{k+\lambda\|k,s}$, $\Pi^{-}_{k+\lambda\|k,s}$	Programming language	Closest object, from [38]
Alexandrova et al. [38]	$\Pi^{-}_{k+\lambda\|k,s}$	$\Pi^{+}_{k+\lambda\|k,s}$ only	Just waypoints	Closest object
This work	$\Pi^{-}_{k+\lambda\|k,s}$, portion of $\Pi^{=}_{k+\lambda\|k,s}$	$\Pi^{+}_{k+\lambda\|k,s}$, $\Pi^{-}_{k+\lambda\|k,s}$	Programming language	Decision tree, see Fig. 5

Table 3
Set of arc labels L (PDDL predicates), and corresponding meaning.

Label	From	To	Meaning
`is_occupying_spot`	Object	Spot	Object is occupying spot
`is_spot_empty`	Spot, s	–	$\nexists obj$: `is_occupying_spot`(obj, s)
`is_holding`	Gripper	Object	Object is held by the gripper
`is_gripper_empty`	Gripper, g	–	$\nexists obj$: `is_holding`(g, obj)
`is_stacked_on`	Object	Object	The first object is above the other (as in [13])
`is_clear_from_top`	Object	–	The object can be grasped
`is_an_object`	*any*	–	Node is an object
`is_a_gripper`	*any*	–	Node is a gripper
`is_a_spot`	*any*	–	Node is a spot

based on the decision-tree of Fig. 5 is striving for a more informative selection, being also based on other semantic predicates, rather than just on ones describing vicinity of objects.

Overall, the method proposed in this work is the sole simultaneously capable of deriving executable code from a single demonstration that is semantically annotated with preconditions and effects. A more schematic comparison with related methods is reported in Table 2.

Limitations of the presented method are primarily related to which extent a single demonstration is informative to completely characterise the skill in terms of effects and preconditions. Although the introduction of the *nodes of interest* that is used to selectively add preconditions from invariant triplets in $\Pi^{=}_{k+\lambda|k,s}$, the method may be still not capable of correctly interpreting relevant preconditions as $\langle n_6, n_3, \texttt{has}\rangle$ in Fig. 4.

Though the characterisation of the effects also through $\Pi^{-}_{k+\lambda|k,s}$ can be beneficial, the same set may contain triplets that *one would not* consider as preconditions, at least if the aim is to generalise as much as possible from a single demonstration. For example, consider once again the demonstration of Fig. 4. As after being placed the green cylinder (n_5) would be on top of the yellow cube (n_4), the triplet $\langle n_5, n_4, \texttt{on\_top}\rangle$ would be considered as an effect, as already discussed. This is definitely a suitable representation if one meant to demonstrate the *stacking* of a cylinder onto a cube, but poorly generalisable as a more generic *placing* skill. Therefore, either the skill will be planned/executed just to stack a cylinder onto a cube, or additional demonstrations, starting from different conditions, have to be provided to characterise the generic placing capability. Despite this seems to be a limitation in the generalisation capabilities of the method, the stacking skill substantially differs from a more generic placing skill in the effects it has on the object below within the stack, if present. A picking skill will be saved to represent the placing action for objects that will be alone in the scene, while the stacking skill will further characterise the specific effects the action has on the object that will result below within the stack. On the other hand, relaxing the definition of $\epsilon_{NoI}\left(\Pi^{=}_{k+\lambda|k,s}\right)$ so to consider triplets having *at least one* element *NoI* would list unnecessary triplets as precondition.

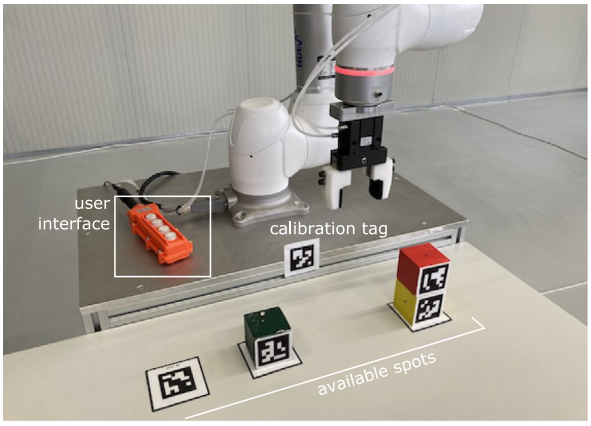

Fig. 6. Layout for the experimental validation showing the robot, its gripper, the three cubes, the three available spots (two of them occupied), and the user interface. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4. Experimental verification

The verification experiments consist in demonstrating to a Doosan M0609 robot some manipulation skills. The manipulator is equipped with a Camozzi CGPS-L-32 pneumatic parallel gripper. The task consists in manipulating three cubes (yellow, red, and green), starting from an arbitrary configuration to reach an arbitrary goal. The workspace of the robot is monitored through a Microsoft LifeCam USB camera (with a resolution of 1280x720 pixels) that is used to locate the parts through Aruco tags. The experimental setup[1] is shown in Fig. 6, showing the robot together with its gripper, the three cubes, and the three spots. The PDDL domain consists in the predicates listed in Table 3. The grounding of predicates is performed using a SN, see Definition 1, and a corresponding *Update function*, see Definition 2, fed with the positions of objects (including the gripper). As already mentioned

[1] A video explaining the proposed method is also available at https://youtu.be/vq-9mIuBUss.

Table 4
Domains collected during the validation campaign, demonstrated arrangements (to be used as initial condition or goal), classification of skills, and success rates (among the 56 collected problems) for each domain. For interpretation of the references to colour in this table, the reader is referred to the web version of this article.

Volunteer/Domain	#1	#2	#3	#4	#5	#6	#7	#8
Condition (init or goal)								
# skills demonstrated	4	6	9	6	6	6	6	4
# pick demonstrations	1	0	2	2	0	1	0	1
# place demonstrations	1	0	2	0	0	1	1	1
# stack demonstrations	2	0	2	1	0	2	0	1
# unstack demonstrations	0	0	1	0	0	2	1	1
Total demonstration time (mm:ss)	3:54	5:03	7:24	6:06	5:38	7:38	8:31	2:14
Success rate	48%	100%	100%	100%	100%	100%	100%	100%

in Section 2, presence of arcs directly determines the grounding of corresponding predicates.

Volunteers ($N = 8$ in total, aged 24−42) were selected among MSc and Ph.D. students of Politecnico di Milano as well as Camozzi employees. They were instructed to demonstrate skills, subdivided into pick, place, stack, unstack, etc., from arbitrary initial conditions, *without being informed about the goal*. They are only told to demonstrate a sufficient number of skills the robot may need to solve arbitrary manipulation task. They interact physically with the robot, as well as with three buttons: one to actuate (open/close) the gripper, one to add the current position, as well as the corresponding state of the gripper, to the list of waypoints $\left(\sigma_i, {}^{w}\boldsymbol{T}_{g_i}\right)$ in $\mathcal{D}_s$, and the last one to start/stop the demonstration of one skill. Volunteers are also asked to avoid manually moving the objects while recording. Volunteers were just informed about the functionalities provided by each button. The minimal training received by the volunteers is consistent with the goal of making the method viable in contexts where operators have minimal or no experience in robotics. After the demonstrations, volunteers are asked to rearrange the parts on the table. The corresponding arrangement will be regarded both as an initial condition/goal. Each pair initial condition/goal will define a specific PDDL problem, while the set of demonstrated skills constitute the PDDL domain. After all the demonstration were collected, each problem is solved through an online planner, [39]. A total number of $N^2 - N = 56$ problems and $N = 8$ possible domains have been collected and used to validate the approach, with particular focus on its generalisation capabilities. As a matter of fact, all possible problems can be solved by means of four PDDL actions: pick, place, stack, and unstack. Therefore, all the demonstrated skills have been also manually classified accordingly.

The outcomes of the experiments are summarised in Table 4. In the 93.5% of the 488 given problems, the planner has been capable of finding a solution starting from the demonstrated set of skills (domain). The average demonstration time has been of 248 ± 125 s, corresponding to approximately 1 min per demonstration. It is worth noticing that in 7 out of 8 demonstrated domains, the success rate has been of 100%, while the reduced success rate corresponding to domain #1 is clearly due to the lack of the demonstration of the unstack skill, that is responsible of all failures.

These results confirm the good generalisation capabilities of the proposed method, as well as the reduced amount of time requested to demonstrate a meaningful set of skills.

As a possible drawback, we noticed that, for some of the volunteers, the segmentation of the demonstrations (that follows immediately from the interaction with the start/stop button present within the user interface) was different from the one expected. For example, some of them were demonstrating non-atomic skills, e.g. complete pick-and-place tasks, rather than dividing them into single pick and place demonstrations. This would be probably solved adopting an automatic segmentation method, which is however out of scope in the present work. In other cases, in turn, the same skill has been demonstrated multiple times (see for example domains #3 and #6) but starting from different conditions, e.g. picking from a stack of 2 objects, and picking from a stack of 3 objects. While the presence of redundant skills is surely not a problem for finding a solution to a given problem, it clearly increases the branching factor that characterises the domain, possibly making slower the search for a plan. Also in this case, a simple solution would be to employ a post-processing of the domain to automatically prune identical skills in terms of preconditions and effects.

5. Conclusions

This work proposes a method to automatically derive planning domains from a minimum number of demonstrations, together with the corresponding motion profile for the execution. For a single skill, the motion profile is generated directly in the programming language of the robot and expressed with respect to the most suitable reference frame that is selected as a function of the derived preconditions and effects.

Results have shown a good generalisation capability despite the reduced time spent during the demonstration.

We can conclude that the approach presented in this paper constitutes a promising direction to ease the use of robots, especially for non-skilled employees.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

This work has been partly supported by Progetto PRIN 2017, Italy "TIGHT: Tactile InteGration for Humans and arTificial systems", prot. 2017SB48FP. The author would also like to thank Camozzi Group SpA for hosting the experimental part of this work.

References

[1] T. Lozano-Perez, Robot programming, Proc. IEEE 71 (7) (1983) 821–841.
[2] W.A. Gruver, B.I. Soroka, J. Craig, T. Turner, Industrial robot programming languages: a comparative evaluation, IEEE Trans. Syst. Man Cybern. (4) (1984) 565–570.
[3] T. Hasegawa, T. Suehiro, K. Takase, A model-based manipulation system with skill-based execution, IEEE Trans. Robot. Autom. 8 (5) (1992) 535–544.

[4] H. Mosemann, F. Wahl, Automatic decomposition of planned assembly sequences into skill primitives, IEEE Trans. Robot. Autom. 17 (5) (2001) 709–718.
[5] S. Bøgh, O.S. Nielsen, M.R. Pedersen, V. Krüger, O. Madsen, Does your robot have skills? in: Proceedings of the 43rd International Symposium on Robotics, VDE Verlag GmbH, 2012.
[6] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Survey: Robot Programming by Demonstration, Tech. rep., Springrer, 2008.
[7] V. Villani, F. Pini, F. Leali, C. Secchi, Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications, Mechatronics 55 (2018) 248–266.
[8] M. Quillan, Semantic Mnemory, Tech. rep., Bolt Beranek and Newman Inc., Cambridge, Ma., 1966.
[9] E. Tulving, in: E. Tulving, W. Donaldson (Eds.), Episodic and Semantic Memory, Organization of Memory, Academic Press, NY, 1972, pp. 381–403.
[10] Y. Kuniyoshi, M. Inaba, H. Inoue, Learning by watching: Extracting reusable task knowledge from visual observation of human performance, IEEE Trans. Robot. Autom. 10 (6) (1994) 799–822.
[11] N. Abdo, H. Kretzschmar, L. Spinello, C. Stachniss, Learning manipulation actions from a few demonstrations, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1268–1275.
[12] M. Stenmark, M. Haage, E.A. Topp, J. Malec, Supporting semantic capture during kinesthetic teaching of collaborative industrial robots, Int. J. Semant. Comput. 12 (01) (2018) 167–186.
[13] F. Ziaeetabar, E. Aksoy, F. Wörgötter, M. Tamosiunaite, Semantic analysis of manipulation actions using spatial relations, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 4612–4619.
[14] T. Savarimuthu, A. Buch, C. Schlette, N. Wantia, J. Roßmann, D. Martínez, G. Alenyà, C. Torras, A. Ude, B. Nemec, et al., Teaching a robot the semantics of assembly tasks, IEEE Trans. Syst. Man Cybern. Syst. 48 (5) (2017) 670–692.
[15] K. Ramirez-Amaro, M. Beetz, G. Cheng, Transferring skills to humanoid robots by extracting semantic representations from observations of human activities, Artificial Intelligence 247 (2017) 95–118.
[16] M. Diehl, C. Paxton, K. Ramirez-Amaro, Automated generation of robotic planning domains from observations, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2021.
[17] R. Caccavale, M. Saveriano, A. Finzi, D. Lee, Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction, Auton. Robots 43 (6) (2019) 1291–1307.
[18] A.J. Ijspeert, J. Nakanishi, S. Schaal, Trajectory formation for imitation with nonlinear dynamical systems, in: Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 2, IEEE, 2001, pp. 752–757.
[19] F. Steinmetz, V. Nitsch, F. Stulp, Intuitive task-level programming by demonstration through semantic skill recognition, IEEE Robot. Autom. Lett. 4 (4) (2019) 3742–3749.
[20] M.J. Aein, E.E. Aksoy, F. Wörgötter, Library of actions: Implementing a generic robot execution framework by using manipulation action semantics, Int. J. Robot. Res. 38 (8) (2019) 910–934.
[21] R. Kartmann, D. Liu, T. Asfour, Semantic scene manipulation based on 3D spatial object relations and language instructions, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), IEEE, 2021, pp. 306–313.
[22] O. Kroemer, G.S. Sukhatme, Learning spatial preconditions of manipulation skills using random forests, in: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), IEEE, 2016, pp. 676–683.
[23] Z. Sui, L. Xiang, O.C. Jenkins, K. Desingh, Goal-directed robot manipulation through axiomatic scene estimation, Int. J. Robot. Res. 36 (1) (2017) 86–104.
[24] Z. Zeng, Z. Zhou, Z. Sui, O.C. Jenkins, Semantic robot programming for goal-directed manipulation in cluttered scenes, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 7462–7469.
[25] M. Ghallab, A. Howe, C. Knoblock, I.D. McDermott, A. Ram, M. Veloso, D. Weld, D.W. SRI, A. Barrett, D. Christianson, et al., PDDL: The Planning Domain Definition Language, Tech. rep., 1998.
[26] R.E. Fikes, N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3–4) (1971) 189–208.
[27] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurts, M. Carreras, ROSPlan: Planning in the robot operating system, in: Proceedings International Conference on Automated Planning and Scheduling, ICAPS, 2015, pp. 333–341.
[28] Y. Zhu, J. Tremblay, S. Birchfield, Y. Zhu, Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 6541–6548.
[29] F. Lehmann, Semantic networks, Comput. Math. Appl. 23 (2–5) (1992) 1–50.
[30] O. Kroemer, S. Niekum, G. Konidaris, A review of robot learning for manipulation: Challenges, representations, and algorithms, J. Mach. Learn. Res. 22 (2021) 30–31.
[31] E. Ugur, J. Piater, Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning, in: 2015 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2015, pp. 2627–2633.
[32] G. Konidaris, L. Kaelbling, T. Lozano-Perez, Symbol acquisition for probabilistic high-level planning, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[33] Y.S. Liang, D. Pellier, H. Fiorino, S. Pesty, Iropro: An interactive robot programming framework, Int. J. Soc. Robot. (2021) 1–15.
[34] M.K. Sharma, O. Kroemer, Relational learning for skill preconditions, in: CoRL, 2020.
[35] G. Konidaris, A. Barto, Skill discovery in continuous reinforcement learning domains using skill chaining, Adv. Neural Inf. Process. Syst. 22 (2009) 1015–1023.
[36] G. Konidaris, S. Kuindersma, R. Grupen, A. Barto, Robot learning from demonstration by constructing skill trees, Int. J. Robot. Res. 31 (3) (2012) 360–375.
[37] H.M. Pasula, L.S. Zettlemoyer, L.P. Kaelbling, Learning symbolic models of stochastic domains, J. Artificial Intelligence Res. 29 (2007) 309–352.
[38] S. Alexandrova, M. Cakmak, K. Hsiao, L. Takayama, Robot programming by demonstration with interactive action visualizations, in: Robotics: Science and Systems (R:SS), 2014, pp. 48–56.
[39] M.C. Magnaguagno, R.F. Pereira, M.D. Móre, F. Meneguzzi, WEB PLANNER: A tool to develop classical planning domains and visualize heuristic state-space search, in: Proceedings of the Workshop on User Interfaces and Scheduling and Planning, UISP, 2017, pp. 32–38.

Andrea M. Zanchettin was born in Cremona (Italy) in 1983. He received his Ph.D. in Information Technology, with honour, from Politecnico di Milano in 2012. From January 2012 until February 2014 he has been a temporary research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). From March 2014 until September 2019 he has been a fixed-term assistant professor at DEIB where he is now a tenured Associate Professor. His research interests are about mechatronic systems, automatic control, and intelligent human–robot interaction. Andrea Zanchettin has been member of the IEEE Robotics and Automation Society since 2009. Since 2017, he has been co-founder and co-chair of the IEEE RAS Technical Committee on Collaborative Automation for Flexible Manufacturing (CAFM). Dr. Zanchettin is also chair of the Italian Chapter of the IEEE RAS (IRAS), as well as vice president for industrial activities of the Italian Institute of Robotics and Intelligent Machines (I-RIM). He is also cofounder and member of the Board of Directors of Smart Robots, a spin-off company of Politecnico di Milano.