In the *International Conference on Social Robotics* (ICSR), Tsukuba, Japan, November 22-24, 2017.

# Learning Affordances for Assistive Robots

Mohan Sridharan and Ben Meadows

Department of Electrical and Computer Engineering, The University of Auckland, NZ
`m.sridharan@auckland.ac.nz, bmea011@aucklanduni.ac.nz`

**Abstract.** This paper describes an architecture that enables a robot to represent, reason about, and learn affordances. Specifically, Answer Set Prolog is used to represent and reason with incomplete domain knowledge that includes affordances modeled as relations between attributes of the robot and the object(s) in the context of specific actions. The learning of affordance relations from observations obtained through reactive execution or active exploration is formulated as a reinforcement learning problem. A sampling-based approach and decision-tree regression with the underlying relational representation are used to obtain generic affordance relations that are added to the Answer Set Prolog program for subsequent reasoning. The capabilities of this architecture are illustrated and evaluated in the context of a simulated robot assisting humans in an indoor domain.

## 1 Introduction

Consider the robot[1] in Figure 1(b) assisting humans in an office by moving particular objects to particular locations or people. While it is difficult to perform such tasks in complex, dynamically changing domains with incomplete domain knowledge, it is difficult to equip the robot with a comprehensive and accurate domain model. The robot may possess rich commonsense knowledge in the form of statements such as "books are usually in the library", which holds in all but a few exceptional circumstances, and information obtained by processing sensor inputs. It may also have descriptions of action preconditions and effects, and the action capabilities of the robot (or other domain agents). For any given goal, reasoning with incomplete knowledge may not identify an existing plan, may provide a suboptimal plan, or may provide a plan whose execution results in unexpected outcomes. For instance, a robot moving to a particular location in a room with a newly carpeted surface without an accurate model of movement on this surface, may end up in an unexpected location. To truly assist humans in such domains, the robot thus needs to represent and reason with incomplete knowledge, and to revise its knowledge over time, which are open problems in robotics and AI.

The architecture described in this paper enables a robot to reason with incomplete knowledge and interactively learn previously unknown action capabilities, often called *affordances*. We build on the understanding that the robot's observations are a rich source of information, and that reasoning and learning can bootstrap off each other. The architecture is based on the following tenets:

- Knowledge elements include symbolic content encoding object constants, relations representing attributes and actions at different abstractions, and axioms composed of these relations.

[1] We use the terms "robot", "agent" and "learner" interchangeably in this paper.

- Affordances are axioms defined over attributes of the robot and/or objects with reference to a particular action; multiple affordances can be defined for each action.
- Knowledge elements are revised non-monotonically, adding to or deleting existing elements by reasoning with the existing knowledge and a history of observed outcomes of actions.
- Learning is coupled with interaction, revising the perceived utility of actions for any given goal through domain exploration performed actively and in response to unexpected transitions.

Our architecture incorporates these tenets by combining the principles of declarative programming, reinforcement learning, and probabilistic sequential decision making. It builds on our prior work that combined declarative programming with probabilistic graphical models for planning and diagnostics [15]. In this paper, we describe the following characteristics of our architecture:

- An action language is used to describe the incomplete knowledge of domain dynamics. This description is translated to a distributed representation in Answer Set Prolog (ASP), which is solved for planning and diagnostics.
- The uncertainty in perception is abstracted away, and the learning of affordances through reactive execution and active exploration is formulated as a Reinforcement Learning (RL) problem guided by ASP-based reasoning.
- Decision-tree regression and a sampling-based approach operate over the relational representation to identify candidate affordances, generalize across them, and include them in the ASP program for subsequent reasoning.

This architecture extends our prior work on discovering action preconditions, and action capabilities that the agent does not possess [16, 17]. We evaluate our architecture in the context of a simulated *Robot Assistant* delivering objects to particular locations or people. Section 2 discusses related work, and Section 3 describes our architecture. Experimental results are presented in Section 4, followed by conclusions in Section 5.

## 2 Related Work

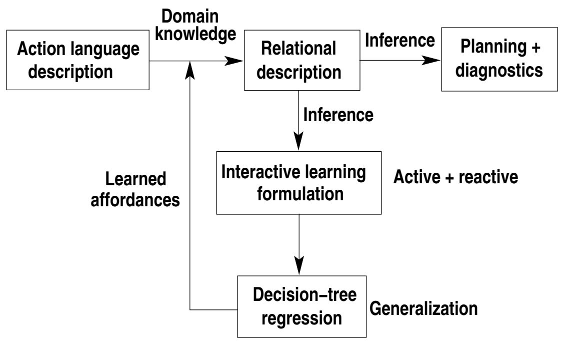
In the research literature, there are different definitions of affordances and methods to infer them. Building on initial work [5], existing formalizations consider three perspectives (agent, environment, observer) and different combinations of attributes of the agent and the environment [11]. Studies on how people judge the capabilities of others show that such judgments can be based on observed kinematics not associated with the desired action [18], and on simplistic representations, e.g., the movement of lights on some joints can be used to judge a person's gender and physical abilities [10]. Approaches for representing and reasoning about affordances have been inspired by these findings and can be grouped under two classes. Approaches in the first class assume a given model and apply probabilistic methods on perceptual descriptions to estimate the probabilities of transitions and observations [14]. Such systems have difficulty using relational and declarative knowledge, and find it difficult to scale to complex domains. Approaches in the second class use logics to encode domain knowledge as relational and hierarchical structures, and to infer activities and affordances [2]. These methods require detailed domain knowledge and do not support probabilistic models of uncertainty. Recent approaches have used a combination of probabilistic and logic-based representations, e.g., Dempster-Shafer theory, to reason about affordances [12]. These methods do not support all desired capabilities such as non-monotonic logical reasoning or incremental knowledge revision. The representation and inference of affordances thus continues to pose open questions.

Early research in learning from observations used first-order logic statements and the observed effects of actions to learn causal laws, but only the encoded conditions or effects were monitored [13]. Another approach refined first-order logic operators by constructing preconditions or effects from unexpected observations, but did not revise incorrect axioms or allow actions to have different outcomes under different contexts [6]. There has also been work on learning operators with contexts and probabilistic effects by searching for dependencies in streams of categorical, sensor data [8]. These approaches have the limitations of first-order logic or do not support generalization as described in this paper. In the logic programming community, inductive logic has been combined with ASP to monotonically learn causal rules [9]. More recently, interactive task learning has been proposed as a general approach for learning from observations of the domain, human demonstrations, or instructions [7]. Interactive learning is often posed as an RL problem, and relational RL (RRL) supports efficient RL in dynamic domains by using relational representations and regression for Q-function generalization [1, 20]. However, RRL algorithms limit generalization to a single planning task or do not support reasoning with commonsense knowledge. Our prior work combined ASP with RRL to discover conditions under which specific actions cannot be executed [16, 17]. The architecture described in this paper builds on complementary research in ecological psychology and AI to introduce a distributed definition of affordances in the context of specific actions. This architecture also supports reasoning with commonsense knowledge, and provides an efficient algorithm for interactively learning affordances.

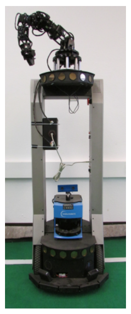
## 3 Proposed Architecture

Figure 1 depicts the overall architecture. For any given goal, ASP-based non-monotonic logical reasoning with a coarse-resolution domain description provides a sequence of abstract actions. Each abstract action is implemented as a sequence of concrete actions, reasoning probabilistically with the relevant part of the fine-resolution system description—see [15] for details. To demonstrate the ability to interactively learn affordances, we abstract away the uncertainty in perception and do not discuss probabilistic planning. We thus describe ASP-based reasoning with a distributed representation of domain knowledge, including affordances, for planning and diagnosis. We also describe the discovery of affordances using principles of relational RL and active learning. This approach can also be used to discover action preconditions and effects, and to discover the action capabilities that an agent lacks [16, 17]—see Section 4. We use the following simulated domain as a running example throughout the paper.

**Example 1** *[Robot Assistant (RA) Domain]*
Consider the robot in Figure 1(b) delivering particular objects to particular rooms or people. Some attributes of this domain include:



(a) Architecture for Learning Affordances.



(b) Robot.

**Fig. 1.** Architecture combines non-monotonic logical reasoning and relational reinforcement learning for learning affordances.

- Sorts such as *entity*, *person*, *robot*, *object*, *book*, *desk* etc.
- Static attributes such as a human's *role*, which can be $\{engineer, manager, sales\}$; the robot's *armtype*, which can be $\{electromagnetic, pneumatic\}$; and an object's *surface*, which can be $\{hard, brittle\}$.
- Fluents such as the location (*loc*) of humans and the robot, which can be *office*, *kitchen*, *library* or *workshop*; *status* of an *object*, which can be $\{damaged, intact\}$; and whether an *object* has been *labeled*.

In this domain, some unknown information of interest may include:

- A heavy object cannot be picked up by a robot with an electromagnetic arm.
- A damaged object cannot be labeled by a robot with a pneumatic arm.
- A damaged object can only be delivered to an engineer, except if it is labeled.
- A brittle object cannot be labeled, except if the arm is electromagnetic.

These statements correspond to different affordances (more details below). The objective is to discover these affordances and include suitable axioms in the ASP program.

## 3.1 Knowledge Representation

Action languages are formal models of parts of natural language used to describe transition diagrams. We use action language $AL_d$ [3], which has a sorted signature with *actions*, *statics*, which are domain attributes whose truth values cannot be changed, and *fluents*, which are domain attributes whose truth values can be changed by actions. Basic fluents obey inertia laws and are changed directly by actions. Defined fluents do not obey inertia laws and are not changed directly by actions. A domain attribute or its negation is a *literal*. $AL_d$ supports three types of statements: *causal laws*, *state constraints* or *executability conditions*.

***Relational Domain Description:*** The domain description includes a system description $\mathscr{D}$ in the form of statements of $AL_d$, and a history $\mathscr{H}$. $\mathscr{D}$ has a sorted signature $\Sigma$ and axioms describing transition diagram $\tau$. $\Sigma$ for the RA domain has basic sorts such as *place*, *robot*, *person*, *object*, *surface*, and *cup*, which are arranged hierarchically, and specific instances, e.g., robot $rob_1$ and places $\{office, workshop, kitchen, library\}$.

Domain attributes and actions are described in terms of their arguments' sorts. In the RA domain, $\Sigma$ includes fluents such as $loc(entity, place)$, and $obj\_status(object, status)$; statics such as $obj\_surface(object, surface)$; and actions such as $move(robot, place)$ and $serve(robot, object, person)$. $\Sigma$ includes the sort $step$ for temporal reasoning, and the relation $holds(fluent, step)$ implies that a particular fluent holds at a particular timestep. System description $\mathscr{D}$ includes axioms such as:

$$\begin{aligned}&move(rob_1, Pl) \ \textbf{causes} \ loc(rob_1, Pl)\\&\neg in\_hand(E, O_2) \ \textbf{if} \ in\_hand(E, O_1), \ O_1 \neq O_2\\&\textbf{impossible} \ pickup(rob_1, O) \ \textbf{if} \ loc(rob_1, L_1), loc(O, L_2), \ L_1 \neq L_2\end{aligned}$$

which correspond to a causal law, state constraint, and executability condition respectively. The recorded history $\mathscr{H}$ of a dynamic domain is a record of fluents observed to be true or false at a time step, i.e., $obs(fluent, boolean, step)$, and the occurrence of an action at a time step, i.e., $hpd(action, step)$. Our model of history also includes defaults describing the values of fluents in their initial state and exceptions, e.g.,"computers are usually in the office but damaged computers are in the workshop".

***Affordance Representation:*** Positive (or enabling) affordances describe permissible uses of objects in actions, whereas negative affordances (or disaffordances) describe unsuitable combinations of objects, agents, and actions. In this paper, we introduce the following generic definition of positive and negative affordances:

$$\begin{aligned}&aff\_forbids(ID, A) \ \textbf{if} \ not \ fails(ID, A), forbidding\_aff(ID, A)\\&\textbf{impossible} \, A \ \textbf{if} \ aff\_forbids(ID, A)\\&aff\_permits(ID, A) \ \textbf{if} \ \ldots\\&\textbf{impossible} \, A \ \textbf{if} \ \ldots, \ not \ aff\_permits(ID, A)\end{aligned}$$

where the "not" represents default negation (explained below). The second statement implies that action $A$ cannot occur if it is not afforded, which depends on whether suitable conditions (defined by first statement) hold true. The fourth statement implies that $A$ cannot occur unless it is permitted by an affordance relation, which can be defined (as in the third statement) jointly over attributes of the robot and/or objects. Any action can have one or more such relations defined with unique $ID$s. For instance:

$$\begin{aligned}\textbf{impossible} \ label(rob_1, Ob) \ &\textbf{if} \ obj\_surface(Ob, brittle),\\&\quad not \ aff\_permits(ID, label(rob_1, Ob))\\aff\_permits(id_1, label(rob_1, Ob)) \ &\textbf{if} \ arm\_type(rob_1, electromagnetic),\\&\quad obj\_surface(Ob, brittle)\end{aligned}$$

where a brittle object cannot usually be labeled, but an electromagnetic arm and a brittle object jointly afford labeling. This distributed representation of affordances (and knowledge) improves generalization, and can simplify inference and information reuse.

***Reasoning with Knowledge:*** The domain representation is translated into program $\Pi(\mathscr{D}, \mathscr{H})$ in CR-Prolog, a variant of ASP that reasons with exceptions to defaults using consistency-restoring (CR) rules [4]. ASP is based on non-monotonic logics, and supports *default negation* and *epistemic disjunction*. Unlike "$\neg a$" that states *a is believed to be false*, "*not a*" only implies *a is not believed to be true*, i.e., the value of a literal can be *unknown*. Also, unlike "$p \lor \neg p$", "$p$ *or* $\neg p$" is not tautologous. ASP can represent recursive definitions, defaults, causal relations, and constructs that are difficult to express in classical logic. The ground literals in an *answer set* obtained by solving $\Pi$ represent beliefs of an agent associated with $\Pi$. Planning and diagnostics are reduced to computing answer sets of $\Pi$. For any given goal, if the robot reasons with incomplete knowledge of affordances, it may not find an existing plan, or the plan may be incorrect or suboptimal, e.g., it may take longer to achieve the goal. We seek to discover such affordances to improve the quality of plans computed.

## 3.2 Axiom Discovery

We describe the steps of axiom discovery in the context of affordances, but the algorithm is applicable to other kinds of axioms too. Missing knowledge can be acquired from different sources, e.g., a human or a repository. However, it is difficult to obtain labeled training samples in complex domains, and access to humans may be limited. Also, any observed transition may be influenced by one or more past (or future) states and actions. We thus enable the robot to interactively acquire labeled samples to learn the axioms. However, learning generic axioms may take many interactions and running all these trials on a robot may be intractable. To address these challenges and mimic the experiences acquired by a robot over a period of time, we formulate interactive axiom discovery as a reinforcement learning (RL) problem in a simulated domain. Unlike previous work on axiom discovery, including our own, our current approach supports both active exploration and exploration in response to unexpected transitions. For the latter, the state described by the action's expected effects becomes the goal state in an RL problem to identify state-action pairs likely to lead to similar states. For active exploration, the robot probabilistically chooses actions to explore, including actions not relevant to a given goal and actions whose preconditions are not satisfied.

***RL and Relational Representation:*** An RL formulation has an underlying Markov decision process (MDP) defined by a set of states ($S$), set of actions ($A$), a state transition function ($T_f$), and a reward function ($R_f$)—functions $T_f$ and $R_f$ are unknown to the robot. Each element in $S$ grounds the domain attributes and whether the expected outcomes of the target action were observed. The values of state-action combinations can then be determined using an RL algorithm—we use Q-learning algorithm [19]. We use ASP-based reasoning to automatically compute the states and actions relevant to a given transition, eliminating parts of the search space irrelevant to the discovery of the desired knowledge. This is equivalent to identifying object constants relevant to the transition of interest $T$, and constructing the system description $\mathscr{D}(T)$, which is the part of $\mathscr{D}$ relevant to $T$. We do so based on the definitions in [15, 16].

In domains with complex relationships between objects, the state space may still be large, making it difficult for Q-learning to converge. After one or more episodes of Q-learning, our approach uses the visited state-action pairs and their estimated Q-values to incrementally update a binary decision tree (BDT) that relationally represents the robot's experiences. The path from the root to a leaf node corresponds to a partial state description. Each internal node corresponds to a boolean test of a specific domain attribute or action, and determines the node's descendants. The remainder of the state description is stored at the leaf. The revised tree is used to compute a new policy, eliminating the need to completely rebuild the tree after each episode. Our approach to update the BDT extends prior work [1]. In each Q-learning episode, the agent stochastically decides to attempt a random action or the one preferred by the current policy, ignoring actions currently invalidated by known axioms. Each action's execution also updates the information stored at a relevant leaf. Over time, a higher value is assigned to outcomes perceived to be similar to the transition of interest. Since this transition may appear with different combinations of static domain attributes, these combinations are varied during RL trials, and the single BDT summarizes experiences from multiple, similar MDPs. Although Q-learning episodes typically terminate when the Q-values converge, for large, complex domains, learning is stopped when a specified fraction of the possible attribute-value combinations are explored.

***Constructing Candidate Axioms:*** To construct candidate axioms, the known structure of axioms is used to extract a partial state-action description from the path from each leaf to the root of the BDT, and the domain attribute information from this description is aggregated. Candidates with low Q-values or corresponding to an action that did not result in the observed transition (for reactive exploration) are eliminated. The resulting structures include information on the mean and variance of the stored Q-values based on different samples. Each structure's literals are partitioned into subsets that contain positive or negative literals. Each unique pairwise combination of those subsets is the basis of a candidate that stores the total Q-value, variance and number of training samples that influenced the candidate. The quality of each candidate is estimated from the Q-values of samples it has experienced. A number of random sample are drawn from the BDT (without replacement) proportional to the size of the tree and the number of literals not used as tests. Each sample is a full state description of the information at the leaf and along the path to the root. Each such state description that matches a candidate axiom adds to its Q-value, variance, and count.

***Validating Candidate Axioms:*** The final step generalizes over the candidate axioms. Candidates not refined by additional training samples after construction are removed. Candidates are ranked by the number of samples used to adjust them, and candidates that elaborate other, higher-ranked ones are removed. The remaining candidates undergo validation tests, e.g., a disaffordance, if true, should describe conditions under which an action will not provide the desired outcomes. If we can find a case that should imply an unexpected transition based on this axiom, but only produces an expected transition, the candidate is incorrect. These validation tests are guaranteed *not* to retract any correct axioms, but may fail to retract some incorrect ones. The remaining candidate axioms, after suitably replacing constants with variables, are included in the ASP program. We refer to this learning algorithm as "Q-RRL".

## 4 Experimental Setup and Results

We evaluate five claims about our architecture's capabilities. We report results in a simulated version of the RA domain, primarily when the robot must learn two positive

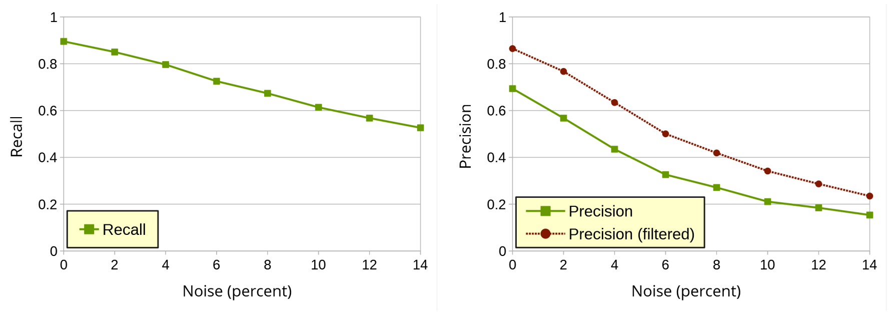


**Fig. 2.** Recall and precision scores as a function of noise. Precision is also plotted after validation.

and two negative affordances. To establish general applicability, we also present results of trials in which two causal laws and two executability conditions are unknown.

**Claim 1: Learned affordances (and other axioms) improve plan quality.** We ran paired tests in which minimal plans were generated using ASP inference, with and without each discovered axiom—a goal was randomly selected for each such paired test. We repeated these trials 1000 times per axiom. We also ran 1000 paired tests with and without all eight target axioms. Learned positive affordances (and causal laws) enabled some new plans that were not otherwise found, and learned negative affordances (and executability conditions) eliminated some plans. We confirmed that plans computed after all the missing knowledge was learned (in any test) were minimal and correct. We therefore used relative measures to evaluate plan quality. For negative affordances, there were 20.7% misses (instances where one run of a paired test found at least one plan and the other did not) on average, and for positive affordances, there were 15.4% misses; there were 10.6% misses (on average) with all axioms. Also, the average length of minimal plans was 2.7; with knowledge of positive and negative affordances, the average difference in plan length was 0.4; it was 0.3 overall.

**Claim 2: Q-RRL accurately discovers unknown axioms.** We ran 2000 repetitions of Q-RRL for each of the eight target axioms. We measured accuracy (*precision* and *recall*) by counting discovered axioms that were logically equivalent to those governing the simulated domain, and thus whose inclusion would improve model accuracy. Recall was 1.0 and 0.61 for negative and positive affordances respectively; it was 0.90 across all eight axioms. Precision was 0.79 for negative affordances, 0.54 for positive affordances, and 0.69 across all eight axioms. After ten validation runs, precision rose to 0.97, 0.72 and 0.86 for negative affordances, positive affordances, and all axioms (respectively).

**Claim 3: Performance degrades gracefully in the presence of noise.** We evaluated Q-RRL in the presence of simulated actuator noise. Figure 2 shows that as noise increases, precision and recall decrease. Validation tests cannot fully compensate for noise because we count candidates that over-specify axioms as false positives, e.g., stating that "a robot with an electromagnetic arm cannot pick up a heavy *red* object" is considered wrong. Such over-specifications are disproportionately likely with increasing noise.

**Claim 4: Q-RRL can learn axioms of varying complexity.** During generalization, the size of a candidate axiom's body is limited to a maximum of four literals, of which half may be negated. In additional tests that increased this cap to six, overall recall increased from 0.90 to 0.94; for positive affordances, recall improved from 0.61 to 0.80. Precision scores were comparable with those for tests without the relaxed axiom length. Positive affordances have a high potential complexity because candidates include literals specific to the affordance and literals of the underlying executability condition it modulates. Increasing the size of the axioms permits additional expressions of each such axiom, and some formulations may be preferred over others. The impact of the unknown positive affordances can thus only be observed in specific combinations of domain attributes. When the number of combinations explored is reduced to a relevant set, only one element of this set may demonstrate the target axiom. Thus, for positive affordances, we also relaxed the requirement that candidate axioms be drawn from multiple branches of the BDT, which improves recall but decreases precision.

**Claim 5: Q-RRL learns from a small number of examples.** Some actions can have undesirable consequences, e.g., handling a brittle object in the RA domain damages it. When the robot is learning about these (previously unknown) outcomes, it is desirable to learn from very few observations. To test whether our approach can support this capability, we altered the reward structure to assign a high penalty to such undesirable transitions. This altered version of Q-RRL applied the corresponding target action 97.8% fewer times (on average) than the unaltered version, e.g., it learns not to handle brittle objects after a few trials. However, this choice decreased recall to 0.55 and precision to 0.62, although both measures increased to 1.0 with validation tests.

## 5 Conclusions

This paper described an architecture to reason about and learn affordances as relations defined over attributes of objects and the robot in the context of specific actions. Answer Set Prolog was used to reason with incomplete domain knowledge for planning and diagnostics. Reinforcement learning with the relational representation was used to learn affordances through both active exploration and exploration in response to unexpected transitions. Decision tree regression and sampling help identify and generalize over candidate affordances. Experimental results indicate the reliable discovery of affordances, robustness to noise, and improvement in plan quality. Future work will focus on reasoning about and learning complex affordances, and re-introduce probabilistic reasoning about perceptual inputs to run experiments on physical robots.

## Acknowledgments

This work was supported in part by the US Office of Naval Research award N00014-13-1-0766 and Asian Office of Aerospace Research and Development award FA2386-16-1-4071. All conclusions are those of the authors alone.

## References

1. Kurt Driessens and Jan Ramon. Relational Instance-Based Regression for Relational Reinforcement Learning. In *International Conference on Machine Learning*, pages 123–130, 2003.

2. Alfredo Gabaldon. Activity Recognition with Intended Actions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, USA, July 11-17 2009.
3. Michael Gelfond and Daniela Inclezan. Some Properties of System Descriptions of $AL_d$. *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming*, 23(1-2):105–120, 2013.
4. Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press, 2014.
5. James J. Gibson. *The Ecological Approach to Visual Perception*. Psychology Press, 1986.
6. Yolanda Gil. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *International Conference on Machine Learning*, pages 87–95, New Brunswick, USA, July 10-13, 1994.
7. James Kirk, Aaron Mininger, and John Laird. Learning Task Goals Interactively with Visual Demonstrations. *Biologicall Inspired Cognitive Architectures*, 18:1–8, 2016.
8. Tim Oates and Paul R. Cohen. Searching for Planning Operators with Context-Dependent and Probabilistic Effects. In *AAAI Conference on Artificial Intelligence*, Portland, USA, August 4-8, 1996.
9. Ramón P. Otero. Induction of the Effects of Actions by Monotonic Methods. In *International Conference on Inductive Logic Programming*, pages 299–310, 2003.
10. Veronica C. Ramenzoni, Tehran J. Davis, Michael A. Riley, and Kevin Shockley. Perceiving Action Boundaries: Learning Effects in Perceiving Maximum Jumping-Reach Affordances. *Attention, Perception and Psychophysics*, 72(4):1110–1119, May 2010.
11. Erol Sahin, Maya Cakmak, Mehmet R. Dogar, Emre Ugur, and Gokturk Ucoluk. To Afford or Not to Afford: A New Formalization of Affordances Towards Affordance-based Robot Control. *Adaptive Behavior*, 15(4):447–472, 2007.
12. Vasanth Sarathy and Matthias Scheutz. A Logic-based Computational Framework for Inferring Cognitive Affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3), October 2016.
13. Wei-Min Shen and Herbert Simon. Rule Creation and Rule Learning through Environmental Exploration. In *International Joint Conference on Artificial Intelligent*, pages 675–680, Detroit, USA, August 20-25, 1989.
14. T. Shu, M. S. Ryoo, and S.-C. Zhu. Learning Social Affordance for Human-Robot Interaction. In *International Joint Conference on Articial Intelligence (IJCAI)*, New York, USA, July 9-15, 2016.
15. Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, `http://arxiv.org/abs/1508.03891`, April 2017.
16. Mohan Sridharan and Ben Meadows. A Combined Architecture for Discovering Affordances, Causal Laws, and Executability Conditions. In *International Conference on Advances in Cognitive Systems (ACS)*, Troy, USA, May 12-14, 2017.
17. Mohan Sridharan, Ben Meadows, and Rocio Gomez. What can I not do? Towards an Architecture for Reasoning about and Learning Affordances. In *International Conference on Automated Planning and Scheduling (ICAPS)*, Pittsburgh, USA, June 18-23, 2017.
18. Thomas A. Stoffregen, Chih-Mei Yang, Russell Giveans, Moira Flanagan, and Benoit G. Bardy. Movement in the Perception of an Affordance for Wheelchair Locomotion. *Ecological Psychology*, 21(1):1–36, February 2009.
19. R. L. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.
20. Prasad Tadepalli, Robert Givan, and Kurt Driessens. Relational Reinforcement Learning: An Overview. In *Relational Reinforcement Learning Workshop at International Conference on Machine Learning*, 2004.