What Can I Not Do? Towards an Architecture for Reasoning about and Learning Affordances

Mohan Sridharan and **Ben Meadows** and **Rocio Gomez**
Department of Electrical and Computer Engineering
The University of Auckland, Auckland 1023, NZ
{m.sridharan@auckland.ac.nz, bmea011@aucklanduni.ac.nz, mgom004@aucklanduni.ac.nz}

Abstract

This paper describes an architecture for an agent to learn and reason about affordances. In this architecture, Answer Set Prolog, a declarative language, is used to represent and reason with incomplete domain knowledge that includes a representation of affordances as relations defined jointly over objects and actions. Reinforcement learning and decision-tree induction based on this relational representation and observations of action outcomes, are used to interactively and cumulatively (a) acquire knowledge of affordances of specific objects being operated upon by specific agents; and (b) generalize from these specific learned instances. The capabilities of this architecture are illustrated and evaluated in two simulated domains, a variant of the classic Blocks World domain, and a robot assisting humans in an office environment.

1 Introduction

Consider a robot[1] assisting humans by delivering specific objects to specific people or places in an office environment. This robot will be called upon to do a variety of tasks in a complex, dynamic environment under resource constraints. While it will be difficult for the robot to operate in such an environment without substantial domain knowledge and guidance, it will be equally challenging for human participants to continuously observe and guide the operation of the robot. Considerable attention continues to be devoted to the underlying problems such as autonomy, perception, control and high-level cognition. However, to truly assist and collaborate with humans in complex environments, the robot also needs the ability to recognize and reason about human intent and its own action capabilities, which remains an open problem in robotics and artificial intelligence (AI).

This paper focuses on enabling a robot to reason about and learn affordances. Research in psychology and AI has provided many theories and computational models for affordances, as discussed in Section 4. We consider a thing's affordance as a combination of its attributes with reference to an agent and an action under consideration (Gibson 1986), e.g., we describe the affordance of a person climbing a stair in terms of the stair's height with reference to the person's leg length (Warren 1984). Reasoning with such affordances is important to compute and successfully execute a planned sequence of actions to achieve any given goal, but it will be difficult to fully and correctly specify the attributes of all objects, agents and actions. In addition, the robot may have to reason with different descriptions of knowledge and uncertainty, including default knowledge ("textbooks are typically in the library") and probabilistic notations ("I am 90% sure I saw Marc in his office"). These challenges are partially ameliorated by the understanding that the robot's observations are a rich (albeit unreliable) source of information about action capabilities. Furthermore, planning and learning can bootstrap off each other, e.g., the robot can actively direct its sensing and actuation to acquire relevant information, and the learned information can be used to compute better plans to achieve desired goals. The architecture proposed in this paper builds on this understanding to address the challenges in reasoning about and learning affordances—it has the following key features:

- An action language describes the incomplete domain knowledge, potentially at different resolutions. The corresponding tightly-coupled transition diagrams are translated to relational representations that are used for inference, planning and diagnostics.
- Non-monotonic logical reasoning, relational reinforcement learning and decision tree induction are used for cumulatively acquiring knowledge regarding affordances, and for generalizing across specific instances. This knowledge is used for subsequent reasoning.

Although the overall architecture supports reasoning with both logic-based and probabilistic knowledge at different resolutions, we abstract away perceptual uncertainty and probabilistic reasoning in this paper to focus on the interplay between planning and learning. Non-monotonic logical reasoning is achieved by translating the domain knowledge to an Answer Set Prolog (ASP) program that is solved for planning and diagnostics. Knowledge about affordances is acquired and generalized using reinforcement learning and decision-tree induction based on the relational representation. These capabilities are illustrated in two simulated domains: (1) a variant of the classic Blocks World domain in which the goal is to stack blocks in specific configurations; and (2) an assistive robotics domain in which the robot has to deliver specific objects to specific people or locations.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] We use terms "robot", "agent" and "learner" interchangeably.

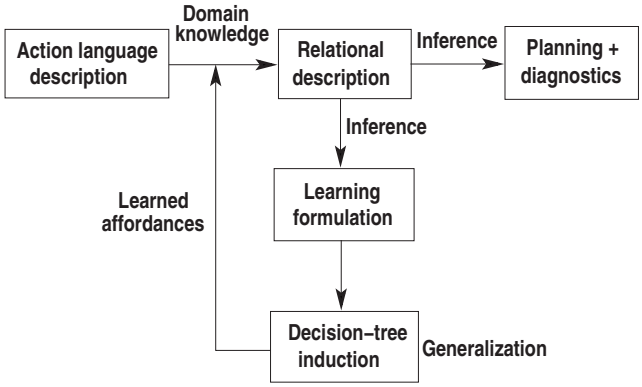

Figure 1: Architecture combines non-monotonic logical reasoning and relational learning for reasoning with and learning affordances.

The remainder of the paper is organized as follows. First, Section 2 describes the example domains and the proposed architecture. Next, Section 3 illustrates, and describes the results of experimentally evaluating, the architecture's capabilities in the example domains. Section 4 discusses a representative set of related work, followed by the conclusions and directions for future research in Section 5.

2 Proposed Architecture

Figure 1 presents an overview of the proposed architecture. Incomplete domain knowledge about actions is encoded in an action language and used in conjunction with other domain knowledge (e.g., about initial domain state) to construct a relational representation. This architecture builds on our prior work that supports logic-based and probabilistic reasoning at two tightly coupled resolutions (Sridharan and Gelfond 2016; Sridharan et al. 2016). However, in this paper we focus on the interplay between reasoning and learning and abstract away perceptual uncertainty. We thus do not describe probabilistic reasoning and translate the relational domain representation into an ASP program that is solved for planning and diagnostics. Reasoning with the ASP program is also used to formulate the task of discovering affordances as a relational reinforcement learning problem, using decision tree induction to generalize across specific instances of affordances. The discovered affordance relations are added to the ASP program and used for subsequent reasoning. We illustrate the architecture's capabilities using the following two simulated domains.

- **Blocks Puzzle (BP):** a variant of the classic Blocks World domain, in which the robot's objective is to use an arm/gripper to stack blocks in particular configurations. The arm is characterized by *atype* (magnetic, pneumatic) and *asize* (small, large). Blocks are characterized by *bcolor* (red, green, blue), *bsize* (small, large), and *bmaterial* (metallic or wooden). The action in the domain is *move*, and the robot may not know that:
  - A magnetic arm cannot move a wooden block.
  - A small arm cannot move a large block.
- **Robot Assistant (RA):** a domain that simulates a robot delivering objects to people or places in an office building with four places (office, kitchen, library, workshop). Each room has one or more doors and instances of objects such as *desk*, *book*, *cup* and *computer*. Human(s) in each room may have a different *role* (engineer, manager, programmer). Objects are characterized by *osize* (small or large), *oweight* (heavy, light) and *otype* (fragile, normal). A door is characterized by *dsize* (small, large) and *dknobpos* (high, medium, low). The robot's arm is characterized by *astrength* (weak, strong), *amaterial* (padded, normal), and *alength* (short, medium, long). The robot's actions include *pickup*, *putdown*, *opendoor*, *move* and *serve*. The robot may not know the following about the *pickup* action:
  - A weak arm cannot pick up a heavy object.
  - Only a padded arm can pick up a fragile object.

  and the following about the *opendoor* action:
  - A weak arm cannot open a heavy door.
  - A short arm cannot open a door with a high knob.

2.1 Relational Domain Representation

Action languages are formal models of parts of natural language that are used for describing action effects. We use action language AL_d (Gelfond and Inclezan 2013) to describe the transition diagrams of our domains. AL_d has a sorted signature with three *sorts*: *statics*, *fluents* and *actions*. Statics are domain properties whose truth values cannot be changed by actions, whereas fluents are domain properties whose truth values can be changed by actions. *Basic* fluents obey laws of inertia and are changed directly by actions, and *defined* fluents do not obey inertial laws and are not changed directly by actions. Actions are defined as a set of elementary operators. A domain property p or its negation $\neg p$ is a domain *literal*. AL_d allows three types of statements:

a **causes** l_b **if** $p_0, \ldots, p_m$ (Causal law)

l **if** $p_0, \ldots, p_m$ (State constraint)

impossible $a_0, \ldots, a_k$ **if** $p_0, \ldots, p_m$ (Executability condition)

where a is an action, l is a literal, l_b is a basic literal, and $p_0, \ldots, p_m$ are domain literals.

Domain description The domain representation consists of system description $\mathscr{D}$, a collection of statements of AL_d, and history $\mathscr{H}$. $\mathscr{D}$ has a sorted signature Σ and axioms that describe the desired transition diagram τ. Signature Σ defines the basic sorts, domain properties and actions of the domain. For instance, basic sorts of the BP domain include *block*, *place*, *bcolor*, *bsize*, *bmaterial*, *robot* etc, whereas the sorts of the RA domain include *place*, *robot*, *role*, *book*, *computer*, *osize*, *oweight*, *otype*, *astrength*, *amaterial*, *alength* etc. Sorts that are subsorts of other sorts, e.g., *cup* and *book* are subsorts of *object*, are arranged hierarchically. Σ also includes ground instances of sorts, e.g., in the RA domain, rob_1 of sort

robot, $\{office, workshop, kitchen, library\}$ of sort *place*, and $\{engineer, programmer, manager\}$ of sort *role*. Both domains include sort *step* for temporal reasoning.

Domain properties and actions are described in terms of the sorts of their arguments. The BP domain's basic fluent $loc(block, place)$ describes the place location of each block. Static attributes include $color(block, bcolor)$, $size(block, bsize)$ and $material(block, bmaterial)$. Action $move(robot, block, place)$ has the robot move a block to another block or the *table*. The RA domain's fluents are $loc(entity, place)$, the location of the robot and objects—the location of humans is a defined fluent (e.g., obtained from a sensor network); and $in_hand(robot, object)$. Static attributes include $armlength(robot, alength)$, $knobpos(door, dknobpos)$ etc. Actions include $move(robot, place)$, $pickup(robot, object)$, $putdown(robot, object)$, $serve(robot, object, person)$ etc. In both domains, relation $holds(fluent, step)$ implies that a particular fluent holds true at a particular timestep.

For the BP domain, $\mathscr{D}$ includes axioms such as:

$$\begin{aligned}
&move(R,B,L) \ \textbf{causes} \ loc(B,L)\\
&\neg loc(B,L_2) \ \textbf{if} \ loc(B,L_1),\ L_1 \neq L_2\\
&\textbf{impossible} \ move(R,B_1,B_2) \ \textbf{if} \ size(B_1,large),\\
&\qquad\qquad size(B_2,small)
\end{aligned}$$

For the RA domain, $\mathscr{D}$ includes axioms such as:

$$\begin{aligned}
&pickup(rob_1,O) \ \textbf{causes} \ in_hand(rob_1,O)\\
&serve(rob_1,O,P) \ \textbf{causes} \ in_hand(P,O)\\
&\neg loc(E,L_2) \ \textbf{if} \ loc(E,L_1),\ L_1 \neq L_2\\
&\textbf{impossible} \ pickup(rob_1,O) \ \textbf{if} \ loc(rob_1,L_1), loc(O,L_2),\\
&\qquad\qquad L_1 \neq L_2
\end{aligned}$$

The recorded history $\mathscr{H}$ of a dynamic domain is usually a record of fluents observed to be true/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 expands this notion to allow representation of defaults describing the values of fluents in their initial states. For instance, we can encode "books are usually in the library", and exceptions, e.g., "cookbooks are in the kitchen".

ASP-based inference The domain representation is translated into a program $\Pi(\mathscr{D}, \mathscr{H})$ in CR-Prolog[2], a variant of ASP that allows us to represent and reason with defaults, direct and indirect exceptions to defaults, and incorporates consistency restoring (CR) rules (Balduccini and Gelfond 2003). ASP is based on stable model semantics, and supports *default negation* and *epistemic disjunction*, e.g., unlike "$\neg a$" that states *a is believed to be false*, "*not a*" only implies *a is not believed to be true*, and unlike "$p \vee \neg p$" in propositional logic, "*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 formalisms. Π consists of causal laws of $\mathscr{D}$, inertia axioms, closed world assumption for defined fluents, reality checks, and observations, actions, and defaults recorded in $\mathscr{H}$. Every default is turned into an ASP rule and a CR rule that allows the robot to assume, under exceptional circumstances, that the default's conclusion is false, so as to restore program consistency. Although not discussed here, Π also includes relations and axioms for explaining unexpected outcomes and partial descriptions extracted from sensor inputs. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π. Algorithms for computing the entailment, and for inference, planning and diagnostics, reduce these tasks to computing answer sets of CR-Prolog programs.

[2]We use the terms "ASP" and "CR-Prolog" interchangeably.

Affordance representation Positive affordances describe permissible uses of objects in actions by agents, whereas negative (i.e., forbidding) affordances describe unsuitable combinations of objects, agents, and actions. In this paper, we represent forbidding affordances as follows:

$$\begin{aligned}
aff_forbids(ID,A) &\leftarrow not\ fails(ID,A),\\
&\qquad forbidding_aff(ID,A)\\
\neg occurs(A,I) &\leftarrow aff_forbids(ID,A)
\end{aligned}$$

The second statement implies that action A cannot occur if it is not afforded, and the first statement provides the conditions under which the action is not afforded. Any action can have one or more such relations defined with unique *ID*s. For instance, if the robot in the RA domain knows that a weak arm cannot pick up a heavy object, the following statements will be included in the ASP program:

$$\begin{aligned}
&forbidding_aff(id1, pickup(R,O))\\
&fails(id1, pickup(R,O)) \leftarrow\\
&\qquad not\ weight(O, heavy)\\
&fails(id1, pickup(R,O)) \leftarrow\\
&\qquad not\ armstrength(R, weak)
\end{aligned}$$

In complex, dynamic domains, it is difficult to equip a robot with accurate and complete domain knowledge. This is especially challenging for complex action capabilities that depend on the attributes of specific objects, agents and actions. Also, the representation of some affordances may be incomplete or omitted unintentionally, and may change over time (e.g., due to wear and tear). The plans created using this incomplete knowledge may result in unintended outcomes. Consider a scenario in the RA domain in which the goal of a robot (that has just delivered an object to the *kitchen*) is to deliver the AI book bk_1 to the *office*. The plan, based on the default knowledge about the location of books (*library*) and the belief that the kitchen door and library door are open, includes $move(rob_1, library)$, $pickup(rob_1, bk_1)$, $opendoor(rob_1, door_2)$, $move(rob_1, office)$, followed by $putdown(rob_1, bk_1)$. The robot expects this plan to succeed but is unable to open the office's door ($door_2$) because it does not know that its weak arm, in conjunction with the heavy door of the office, does not afford the *opendoor* action. In this paper, we focus on discovering such previously

unknown affordances that will forbid the corresponding actions from being considered during planning to achieve any given goal.

2.2 Relational Learning

Incompleteness or lack of knowledge about affordances is likely to be discovered in one of two ways. The discovery of positive (i.e., enabling) affordances may require the robot to actively explore new object-action combinations, including actions expected to fail. Negative (i.e., forbidding) affordances, which we focus on here, are usually found when the unexpected transition observed after executing a plan step is considered to imply that the action is inappropriate given the object and agent involved. To model the relationships that led to this transition, our architecture explores the space of relevant transitions, produces candidate relations and axioms corresponding to affordances, generalizes across these candidates, validates the most likely candidates, and adds them to the system description, as described below.

Tree induction When an action produces an unexpected transition, the resultant state becomes the goal state in a relational reinforcement learning (RRL) problem to find state-action pairs likely to lead to analogous "error" states. The RL formulation is based on the tuple $\langle S, A, T_f, R_f \rangle$ that defines the underlying Markov decision process (MDP):

- S is the set of states;
- A is the set of actions;
- $T_f : S \times A \times S' \rightarrow [0,1]$ is the state transition function;
- $R_f : S \times A \times S' \rightarrow \Re$ is the reward function.

Here, T_f and R_f are unknown (to the agent), and each element in S grounds the domain properties. Such an RL formulation is chosen to mimic the cumulative and interactive acquisition of experiences by a robot as it performs its assigned tasks—it also helps distribute the immediate reward suitably over a sequence of past states and actions. In complex domains, instead of operating over the entire state-action space, ASP-based reasoning can automatically compute the system description relevant to a given transition. This reduction will significantly improve the computational efficiency of RL, while elegantly maintaining correspondence with the original description—for details, see our prior work (Sridharan and Gelfond 2016; Sridharan et al. 2016). The description below assumes that such a reduction is constructed before performing RL over the corresponding MDP.

To estimate the Q-values of state-action pairs $Q(s,a)$, we employ a variation on the basic Q-learning algorithm, using a relational representation to generalize to relationally equivalent states. An episode of Q-learning terminates when a time limit is exceeded, or the target action succeeds, i.e., further exploration will not provide useful information—the physical configuration of objects is reset to what it was at the beginning of the episode for another episode. After one or more episodes of Q-learning, all visited state-action pairs and their estimated Q-values are used to update a binary decision tree (BDT). The path from the BDT's root to a leaf node corresponds to a pair (s,a) of partial state description s and action a. Internal nodes correspond to boolean *tests* of specific attributes (of objects, agents or actions, and determine the node's descendant branches. The remainder of the state description is stored at the leaf; some of this information may be transferred to a new node when the BDT is updated, if this reduces variance in Q-values. The revised tree is used to compute a new policy, eliminating the need to completely rebuild the tree after each episode. When the learning process is completed, the BDT relationally represents the robot's experiences.

Construction and revision of the BDT was informed by the algorithm of Driessens and Ramon (2003), but our approach differs substantially. In each Q-learning episode, the system stochastically decides to attempt either a random action or the one preferred by the current policy, ignoring actions currently invalidated by known constraints or affordances. Each action application also updates the information stored at a relevant leaf. As more such examples are processed, a higher value is assigned to outcomes perceived to be similar to the originally encountered instance of unexpected transition. These (similar) unexpected transitions may appear in the context of many different combinations of attributes of agents and objects. To generalize across such related transitions, the underlying combinations of the values of attributes are revised during learning (e.g., after Q-values have converged for a specific MDP). With our BDT representation, we are able to easily include the experiences from the corresponding different, but similar, MDPs. As these different MDPs are explored, many of the explored combinations of values of attributes will modify the current Q-values. The Q-learning episodes are performed until the Q-values stored in the BDT converge. Overall, multiple factors could determine the amount of time dedicated to this learning task that mimics learning through interaction with the environment. For complex domains, we assume that the exploration is terminated when a certain fraction of the possible search space has been sampled.

Candidate generation In the next step, candidate affordances are constructed. Given the focus on forbidding affordances, each candidate is composed of the precluded action, and a conjunction of one or more relevant attributes of objects and agents, as described in Section 2.1.

To construct candidate affordance relations, each leaf from the induced BDT is examined, a partial state-action description is extracted using the path from the leaf to the root, and the stored information about attributes is aggregated. Branches with low Q-values or corresponding to an action that did not result in the observed unexpected transition are eliminated. The resulting structures include information on the mean and variance of the stored Q-value, based on the different samples clustered under each leaf. Each structure's statements about object attributes holding or not holding are partitioned into two subsets, and all possible pairwise combinations of those subsets are examined, producing unique tuples that each form the basis of a candidate and store the (a) amassed Q-value; (b) total variance; and (c) number of

training samples that influenced the candidate.

Next, each candidate's quality is computed based on the Q-values of the samples it has experienced. Random samples are drawn from the BDT[3] without replacement. Each sample is a state description combining the information stored at a leaf and the tests at its non-terminal ancestors. Each state description that matches a candidate affordance adds to its Q-value, variance, and count. For instance, consider the target affordance in the BP domain, which prevents a magnetic arm from trying to move a non-metallic object. Consider a candidate affordance constructed from a leaf node whose path to the root describes the state $color(b_1, red)$, $\neg material(b_1, metallic)$, $armtype(rob_1, magnetic)$, with a predicted Q-value = 9. The resulting candidate may be:

positive: [$armtype(rob_1, magnetic)$]
negative: [$material(b_1, metallic)$]
`Q-sum: 9.0, Count: 1, Mean: 9.0`

Of the random samples drawn during candidate quality estimation, only some will match the candidate's partial state description. For example:

positive: [$color(b_1, blue)$, $color(b_2, red)$, $size(b_1, small)$, $armtype(rob_1, magnetic)$]
negative: [$material(b_1, metallic)$, $size(b_2, small)$]
Q: `10.0`

The system uses this sample to update the candidate:

positive: [$armtype(r_1, magnetic)$]
negative: [$material(b_1, metallic)$]
`Q-sum: 19.0, Count: 2, Mean: 9.5`

Once all candidate are found, we move to the next step.

Filtering candidates The final set of affordances is identified by eliminating candidates that do not have a sufficiently high likelihood of representing the truth. First, candidates that were not refined by additional training samples after their construction are removed. Then, the highest-valued candidates are ranked by the number of samples used to update them. Any candidates that elaborate other, higher-ranked candidates are also removed. Finally, the adequacy of the remaining candidates is validated in simulation. A candidate negative affordance, if true, should describe conditions when an action is not afforded by the agent and object under consideration. If we can construct a state in which the action corresponding to a forbidding affordance should not be considered, but executing that action results in the expected outcomes, the constructed candidate affordance is incorrect. To conduct such validation tests, our approach takes a random state where the target action is known to succeed, and makes the minimal changes required (to attributes of objects and the agent) to make the state match the partial state description of the candidate. If the action in this adjusted state is executed successfully, the candidate affordance is discarded. These validation tests are guaranteed *not* to retract correct affordances, but may fail to retract incorrect affordances. Candidate affordances that survive all these filters are considered to be generalized affordances—the corresponding axioms are added to the ASP system description after suitably replacing constants with variables.

[3]The number of draws is heuristically set to be proportional to the tree's size and the number of attributes not used as tests.

3 Experimental Setup and Results

In this section, we describe the results of experimentally evaluating the following claims:

1. Proposed architecture supports accurate, cumulative discovery of affordances;
2. Fraction of affordances discovered increases with the fraction of the search space explored;
3. Affordance discovery approach is robust to perceptual noise; and
4. Discovered affordances improve plan quality.

These claims are evaluated in the BP domain and RA domain, with two and four (generic) target affordances (respectively).

3.1 Experimental Setup

Parameters in our RRL approach were initialized with values that were determined (experimentally) to provide reasonable results without unduly increasing the processing time. For instance, the learning rate and the policy's exploration preference were fixed at 0.1. The values of positive and negative reward were 10.0 and 0.0 respectively—the system was agnostic to the values of these parameters as long as they were distinctly dissimilar. The candidate affordances considered were constrained to have no more than two positive literals and two negative literals formed of domain properties. This limit was more than sufficient for the affordances in our domains—they can be increased (as needed) for more complex domains. Furthermore, up to 10 validation tests were conducted to evaluate and filter the candidate affordances.

Methodology In the experimental trials reported below, affordances associated with each action were discovered concurrently. Unless otherwise specified, each value of a performance measure reported below was obtained by averaging the results over 1000 repetitions. We used *precision* and *recall* as the performance measures—they indicate the system's ability to avoid false positives and false negatives respectively. The affordances produced were required to exactly match the ground truth to be counted as true positives, i.e., over-specifications of an affordance, although not exactly incorrect, are considered false positives.

Simplifying assumption: Our interpretation of relevance (in Section 2.2) translates to the simplifying assumption that axioms describing affordances may only refer to the attributes and object classes involved in the action. To determine the effect of this assumption, we examined a "maximal" version not constrained by this simplifying assump-

tion, i.e., it can consider information about any of the domain objects in its search for affordances. We considered four different parameterizations, with 200 repeated trials for each parameterization. Since its search space was rather large, the maximal version discovered fewer affordances than the version with the simplifying assumption, for the same number of trials, e.g., it discovered 34.1% fewer affordances in the BP domain and the recall dropped from 0.86 to 0.61 in comparison with a system that included the simplifying assumption. When the maximal version was allowed to explore the same proportion of the search space as the version with the simplifying assumption, performance improved, e.g., in some trials in the BP domain, the recall and precision were 1.0 and 0.98 respectively. However, in the RA domain, the number of possible combinations to explore was so large ($\approx$ 1.7 billion) that it was not feasible to evaluate the maximal version—this is also likely to be true of other such complex domains. All the experimental results discussed below thus include the simplifying assumption, i.e., only relevant objects and agent attributes were considered while adding nodes to the BDT, varying the object configurations, and constructing candidate affordances.

3.2 Execution Traces

The following execution traces illustrate the working of the proposed architecture and learning approach.

Execution Example 1. *[BP domain execution example]* In the BP domain, assume that the learner does not know that it is impossible to pick up a wooden object with a magnetic arm. Assume there are three blocks: b_1 (small red wooden), b_2 (small blue wooden), and b_3 (small blue metallic), and the initial state is:

$$loc(b_1, table),\ loc(b_2, table),\ loc(b_3, table)$$

and it is known that $armtype(rob_1, magnetic)$. Now, let the goal state description be:

$$loc(b_1, table),\ loc(B, b_1)$$

which implies that the robot has to stack some block on block b_1. Based on the existing knowledge, the robot constructs a plan with the action $move(rob_1, b_2, b_1)$, which is expected to achieve the goal. However, there is an unexpected transition, and the robot observes that b_2 is still on the *table*. This unexpected transition triggers RRL for discovering any unknown affordances.

During the RRL trials, the robot acquires experiences from training examples involving a magnetic arm and a wooden object, which may differ in terms of the other attributes of objects and the robot. Eventually, the robot is able to add the following axioms related to a specific affordance to its system description:

$$\begin{aligned} &forbidding_aff(id1, move(R,B,L)) \\ &fails(id1, move(R,B,L)) \leftarrow \\ &\qquad not\ material(B, wooden) \\ &fails(id1, pickup(R,O)) \leftarrow \\ &\qquad not\ armtype(R, magnetic) \end{aligned}$$

which will, for a wooden object and magnetic arm, ensure that action $move(rob_1, b_2, b_1)$ is not available for inclusion in the computed plan. Upon replanning, rob_1 will now compute a different plan:

$$move(rob_1, b_3, b_1)$$

which when executed will successfully place metallic block b_3 on top of b_1, thus achieving the desired goal.

Execution Example 2. *[RA domain execution example]* In the RA domain, suppose the robot does not know that it cannot open a door with a high knob if it has a short arm. Assume that the library has two doors—$door_1$ is light and has a high knob while $door_2$ is light and has a knob at medium height. Assume that the initial state has $loc(rob_1, kitchen)$, i.e., the robot is in the kitchen. Let the desired goal state be:

$$in_hand(rob_1, bk_1)$$

which implies that the robot has to have the book bk_1 in its arm. It is also known that:

$$size(bk_1, small),\ weight(bk_1, light),$$
$$armlength(rob_1, short),\ armstrength(rob_1, strong)$$

Based on default knowledge, the robot believes that bk_1 is in the *library*. Hence, the robot constructs the following plan to achieve this goal:

$$opendoor(rob_1, door_1),\ move(rob_1, library),$$
$$pickup(rob_1, bk_1).$$

which could potentially be because $door_1$ is closer, although we do not model that here. Its first action is to open $door_1$ in order to enter the *library*. This results in an unexpected transition because the robot observes that $door_1$ is not open—this is because the robot has a short arm. This unexpected transition triggers RRL, and the robot gathers experiences corresponding to different attributes of the robots and doors. Eventually, the robot is able to learn a new forbidding affordance for the *opendoor* action that is encoded as:

$$\begin{aligned} &forbidding_aff(id1, opendoor(R,D)) \\ &fails(id1, opendoor(R,D)) \leftarrow \\ &\qquad not\ armlength(R, short) \\ &fails(id1, pickup(R,O)) \leftarrow \\ &\qquad not\ knobpos(D, high) \end{aligned}$$

which, in our current example of a short arm and high doorknob, is sufficient to prevent the robot from creating plans that require it to open $door_1$, i.e., it does not consider action $opendoor(rob_1, door_1)$ while computing a plan. Instead, rob_1 finds an alternative plan:

$$opendoor(rob_1, door_2),\ move(rob_1, library),$$
$$pickup(rob_1, bk_1).$$

which it is able to execute to successfully achieve the goal—open $door2$, enter the *library*, and pick up $bk1$.

An appealing qualitative outcome of the proposed approach is that the discovered affordances are relations that may be used in different ways with different actions. For instance, having a weak arm may forbid the robot from lifting an heavy object. The same relation may, however, *enable* the robot to move faster. Modeling such enabling affordances is a direction for future research.

Table 1: Rate of ground truth affordances found (recall) for different extents of learning.

Exploration	Recall	Precision
5%	0.09	1.0
10%	0.27	1.0
20%	0.56	0.99

3.3 Discussion

We now experimentally examine and evaluate the four claims made in the beginning of Section 3.

Accuracy We conducted 1000 trials in each domain to evaluate the ability to discover affordances. Each trial explored 10% of the search space. In the BP domain, the 1000 trials corresponded to 2000 possible affordances—two possible affordances in each trial—of which 1570 were found, leading to a recall of 0.79. Three of the discovered affordances were over-specifications that were considered to be incorrect, i.e., precision ≈ 1.0. In the RA domain, the 1000 trials corresponded to 500 for each of the two actions that each have two unknown affordances. Of the total 2000 possible affordances, 533 were found, providing recall and precision scores of 0.27 and 1.0 respectively. These results support the claim that the proposed architecture is able to cumulatively and accurately discover the affordances.

Discovery and effort To determine whether the fraction of affordances discovered increases with the fraction of the search space explored, we conducted trials in the RA domain by systematically varying the fraction of the space of possible combinations of domain properties that the system is allowed to explore—this implicitly varies the number of Q-learning episodes the robot uses to discover the affordances. The corresponding performance, described in terms of the recall and precision scores, is summarized in Table 1. The results indicate that as the robot explores a larger fraction of the search space, the recall scores increase. The recall scores are low when only a small fraction of the relevant search space is explored. However, the almost perfect precision scores (there are a few over-specifications when 20% of the relevant space is explored) indicate that the robot is able to learn cumulatively from experience.

Perceptual noise Next, we evaluated whether the proposed approach provided robustness to perceptual noise, which we interpreted as the noise having a negative but non-catastrophic impact on performance. We introduced noise in the form of a fixed chance for an action to have an unexpected outcome in the form of the removal or addition of a single literal formed of a random fluent of the desired resultant state. We performed 1000 trials in the RA domain with a 10% exploration of the search space, and for 3% and 6% added noise. The performance with added noise was compared with the performance with no noise, which corresponds to the second line of Table 1.

With 3% noise, the robot was still able to discover some affordances, with a recall of 0.29—although this is better than the recall with no noise, the difference is not statistically significant. The corresponding precision score was 0.87, but all the false positives were over-specifications of the target axioms, i.e., they included conditions that would correctly prevent an action from being considered during planning but were not in the most general form possible. For example, the following affordance was reported:

$$
\begin{aligned}
&forbidding_aff(id2, opendoor(R,D))\\
&fails(id2, opendoor(R,D)) \leftarrow\\
&\qquad not\ armlength(R, short)\\
&fails(id2, pickup(R,O)) \leftarrow\\
&\qquad not\ knobpos(D, high)\\
&fails(id2, pickup(R,O)) \leftarrow\\
&\qquad not\ armmaterial(R, padded)
\end{aligned}
$$

where the fact that the robot's arm was padded was an over-specification of the desired affordance, that a short arm cannot be used to open doors with high knobs.

Next, with 6% noise, the performance dropped further, resulting in a recall of 0.21 and a precision of 0.79. In this case, all but two of the false positives were over-specifications of the desired target axioms, i.e., the affordances discovered were still (for the most part) just insufficiently general rather then incorrect. Discounting these over-specific affordances would result in near perfect precision. We also observed that, even with the added noise, the recall and precision scores improved as the fraction of the search space explored increased. We hypothesize that is may be possible to further improve performance by using ASP-based reasoning to identify and eliminate inconsistencies during generalization.

Planning performance with learned affordances: Finally, to evaluate the effect of the discovered affordances on the quality of plans generated, we conducted 1000 paired trials of ASP-based planning with and without the corresponding axioms—the initial state and goal were set randomly in each paired trial. For the BP domain, 1.67 plans were found with the axioms, and 2.84 plans were found without the axioms. In another set of 1000 paired trials in the BP domain with an observation (in history $\mathscr{H}$) to limit the number of plans computed, e.g., $obs(on(b1,b0),2)$, 0.87 plans were found with the axioms and 1.42 plans were found without the axioms. In a set of 1000 trials conducted in the RA domain, 1.87 plans were found with the axioms and 3.90 plans were found without the axioms. We verified that including the discovered axioms always provided correct plans that were a subset of the plans found without the axioms.

4 Related Work

In psychology, action capabilities have been determined by modeling how people judge the capabilities of another person (Mark 2007). Researchers have shown that such judgments are based on the observed kinematics that do not have

to be associated with the desired action (Ramenzoni et al. 2010; Stoffregen et al. 2009). Observers can judge if an actor can jump to reach an object by watching the actor walk or lift a weight—also, different kinematics are not equally informative. Research has also shown that humans make accurate judgments of others' capabilities based on very simple representations, e.g., the movement of lights on a few joints of a person can be used to judge gender and physical abilities (Ramenzoni et al. 2010). These studies and other work in ecological psychology have inspired approaches in AI for modeling and reasoning about affordances, e.g., in the context of tool and object use (Griffith et al. 2012; Guerin, Kruger, and Kraft 2013), but open questions remain regarding the definition and representation of affordances (Horton, Chakraborty, and Amant 2012).

There are two broad classes of approaches for inferring affordance, intent, and activity. Systems in the first class apply probabilistic methods such as hidden Markov models and partially observable Markov decision processes on perceptual descriptions (Brand, Oliver, and Pentland 1997; Kelley et al. 2012). These systems assume that the model structure is given, and estimate probabilities of transitions and observations. Systems in this class have difficulty encoding and using relational structures and declarative knowledge, and do not scale to complex domains. Approaches in the second class infer affordance and intent as logic statements over relational, hierarchical knowledge structures. These approaches have used first-order logic and non-monotonic logic for recognizing activities and intentions (Gabaldon 2009; Hobbs, Stickel, and Martin 1993). Since logic-based reasoning methods typically require detailed domain knowledge and do not support probabilistic models of uncertainty, methods such as Bayesian logic programs (Milch et al. 2006) and probabilistic event calculus (Skarlatidis et al. 2015) reason with logic-based and probabilistic representations. However, these methods do not support all the desired capabilities such as non-monotonic logical reasoning, and qualitative reasoning with default knowledge and beliefs. Probabilistic extensions to ASP address some of these limitations (Baral, Gelfond, and Rushton 2009; Lee and Wang 2015), but they do not support incremental addition of knowledge and are computationally expensive.

Algorithms have been developed for agents to start with incomplete domain knowledge and learn from interactions. Early research used first-order logic statements and the observed effects of actions to learn causal laws when predictions fail, but only an action's encoded conditions or effects were monitored (Shen and Simon 1989). Another approach incrementally refined first-order logic operators by making the unexpected observations preconditions or effects—this work augmented existing knowledge but did not revise incorrect axioms, and did not consider that the same action can lead to different outcomes in different contexts (Gil 1994). In addition to the limitations of first-order logic, these approaches did 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 (Otero 2003), and a maximum satisfiability framework has been used with plan traces for refining incomplete domain models (Zhuo, Nguyen, and Kambhampati 2013). Interactive learning has also been posed as an RL problem with an underlying MDP (Sutton and Barto 1998). Approaches for efficient RL in dynamic domains include sample-based planning algorithms (Walsh, Goschin, and Littman 2010), and Relational RL, which uses relational representations and regression for Q-function generalization (Dzeroski, Raedt, and Driessens 2001; Tadepalli, Givan, and Driessens 2004). However, most RRL algorithms focus on planning, limit generalization to a single planning task, or do not support the desired commonsense reasoning capabilities. One exception was our prior work that combined ASP with RRL to discover conditions under which specific actions cannot be executed (Sridharan and Meadows 2016). The proposed architecture builds on the complementary notions of affordance in ecological psychology and AI, while supporting non-monotonic logical reasoning and probabilistic reasoning with incomplete domain knowledge, in the context of cumulative and interactive learning of affordances.

5 Conclusions

The paper described an architecture that combined non-monotonic logical reasoning and relational learning to reason about and learn affordances that are defined as relations between the attributes of objects, agents and actions. Answer Set Prolog was used to reason with incomplete domain knowledge for planning and diagnostics, while relational reinforcement learning and decision tree induction were used to identify specific candidate affordances and generalize across specific instances. Experimental results indicate the reliable, cumulative discovery of affordances, robustness to noise, and improvement in plan quality.

The proposed architecture opens up multiple directions for further research. We have only focused on learning affordance relations that forbid the execution of specific actions—future work will also support the learning of affordances that enable the execution of specific actions. In addition, we have currently abstracted away the ability to reason probabilistically, especially with perceptual inputs—this ability will be important when the architecture is used on physical robots that sense and interact with the environment. The long-term objective is to enable reliable and efficient inference of intent and affordance on robots that assist humans in complex domains.

Acknowledgements

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

References

Balduccini, M., and Gelfond, M. 2003. Logic Programs with Consistency-Restoring Rules. In *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, 9–18.

Baral, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming* 9(1):57–144.

Brand, M.; Oliver, N.; and Pentland, A. 1997. Coupled Hidden Markov Models for Complex Action Recognition. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 994–999.

Driessens, K., and Ramon, J. 2003. Relational Instance-Based Regression for Relational Reinforcement Learning. In *International Conference on Machine Learning*, 123–130. AAAI Press.

Dzeroski, S.; Raedt, L. D.; and Driessens, K. 2001. Relational Reinforcement Learning. *Machine Learning* 43:7–52.

Gabaldon, A. 2009. Activity Recognition with Intended Actions. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gelfond, M., and Inclezan, D. 2013. 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.

Gibson, J. J. 1986. *The Ecological Approach to Visual Perception*. Psychology Press.

Gil, Y. 1994. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *International Conference on Machine Learning*, 87–95.

Griffith, S.; Sinapov, J.; Sukhoy, V.; and Stoytchev, A. 2012. A Behavior-Grounded Approach to Forming Object Categories: Separating Containers From Noncontainers. *IEEE Transactions on Autonomous Mental Development* 4:54–69.

Guerin, F.; Kruger, N.; and Kraft, D. 2013. A Survey of the Ontogeny of Tool Use: from Sensorimotor Experience to Planning. *IEEE Transactions on Autonomous Mental Development* 5:18–45.

Hobbs, J.; Stickel, M.; and Martin, P. 1993. Interpretation as Abduction. *Artificial Intelligence* 63:69–142.

Horton, T. E.; Chakraborty, A.; and Amant, R. S. 2012. Affordances for Robots: A Brief Survey. *Avant: Journal of Philosophical-Interdisciplinary Vanguard* III(2):70–84.

Kelley, R.; Tavakkoli, A.; King, C.; Ambardekar, A.; Nicolescu, M.; and Nicolescu, M. 2012. Context-Based Bayesian Intent Recognition. *IEEE Transactions on Autonomous Mental Development* 4(3):215–225.

Lee, J., and Wang, Y. 2015. A Probabilistic Extension of the Stable Model Semantics. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.

Mark, L. S. 2007. Perceiving the Actions of Other People. *Ecological Psychology* 19(2):107–136.

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2006. BLOG: Probabilistic Models with Unknown Objects. In *Statistical Relational Learning*. MIT Press.

Otero, R. P. 2003. Induction of the Effects of Actions by Monotonic Methods. In *International Conference on Inductive Logic Programming*, 299–310.

Ramenzoni, V. C.; Davis, T. J.; Riley, M. A.; and Shockley, K. 2010. Perceiving Action Boundaries: Learning Effects in Perceiving Maximum Jumping-Reach Affordances. *Attention, Perception and Psychophysics* 72(4):1110–1119.

Shen, W.-M., and Simon, H. 1989. Rule Creation and Rule Learning through Environmental Exploration. In *International Joint Conference on Artificial Intelligent*, 675–680.

Skarlatidis, A.; Artikis, A.; Filippou, J.; and Paliouras, G. 2015. A Probabilistic Logic Programming Event Calculus. *Theory and Practice of Logic Programming* 15(2):213–245.

Sridharan, M., and Gelfond, M. 2016. Using Knowledge Representation and Reasoning Tools in the Design of Robots. In *IJCAI Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS)*.

Sridharan, M., and Meadows, B. 2016. Should I do that? Using Relational Reinforcement Learning and Declarative Programming to Discover Domain Axioms. In *International Conference on Developmental Learning and Epigenetic Robotics (ICDL-EpiRob)*.

Sridharan, M.; Gelfond, M.; Zhang, S.; and Wyatt, J. 2016. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, http://arxiv.org/abs/1508.03891.

Stoffregen, T. A.; Yang, C.-M.; Giveans, R.; Flanagan, M.; and Bardy, B. G. 2009. Movement in the Perception of an Affordance for Wheelchair Locomotion. *Ecological Psychology* 21(1):1–36.

Sutton, R. L., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

Tadepalli, P.; Givan, R.; and Driessens, K. 2004. Relational Reinforcement Learning: An Overview. In *Relational Reinforcement Learning Workshop at International Conference on Machine Learning*.

Walsh, T. J.; Goschin, S.; and Littman, M. L. 2010. Integrating Sample-Based Planning and Model-Based Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*.

Warren, W. H. 1984. Perceiving Affordances: Visual Guidance of Stair Climbing. *Journal of Experimental Psychology: Human Perception and Performance* 10(5):683–703.

Zhuo, H. H.; Nguyen, T.; and Kambhampati, S. 2013. Refining Incomplete Planning Domain Models Through Plan Traces. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2451–2457.