Artificial Intelligence for Human-Robot Interaction
AAAI Technical Report FS-17-01

Integrating Knowledge Representation, Reasoning, and Learning for Human-Robot Interaction

Mohan Sridharan
Department of Electrical and Computer Engineering
The University of Auckland, NZ
m.sridharan@auckland.ac.nz

Abstract

Robots interacting with humans often have to represent and reason with different descriptions of incomplete domain knowledge and uncertainty, and revise this knowledge over time. Towards achieving these capabilities, the architecture described in this paper combines the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning. For any given goal, non-monotonic logical reasoning with a coarse-resolution representation of the domain is used to compute a tentative plan of abstract actions. Each abstract action is implemented as a sequence of concrete actions by reasoning probabilistically over the relevant part of a fine-resolution representation tightly-coupled to the coarse-resolution representation. The outcomes of executing the concrete actions are used for subsequent reasoning at the coarse-resolution. Furthermore, the task of interactively learning axioms governing action capabilities, preconditions and effects, is posed as a relational reinforcement learning problem, using decision tree regression and sampling to construct and generalize over candidate axioms. These capabilities are illustrated in simulation and on a physical robot moving objects to specific people or locations in an indoor domain.

1 Introduction

Consider a robot assisting humans in an office by finding and delivering particular objects to particular locations or people. In such complex domains, it will be difficult to equip the robot with an accurate and complete model of the domain. The incomplete domain model may include rich commonsense knowledge of the form "books are usually in the library", which holds in all but a few exceptional circumstances, e.g., cookbooks are in the kitchen and manuals are in the laboratory. The robot also obtains a partial and unreliable domain description by processing inputs from sensors (e.g., cameras, microphones). In addition, the robot has some knowledge of the axioms governing domain dynamics and the robot's action capabilities (*affordances*), which may change over time. For any given goal, reasoning with this incomplete knowledge may not identify an existing plan, or the computed plan may be incorrect or sub-optimal. For instance, a robot that does not know it can open doors may compute a longer path to its destination, and a robot moving on a newly carpeted surface may end up in an unexpected location. To truly work with humans, the robot has to represent and reason with different descriptions of knowledge and uncertainty, and incrementally revise the existing knowledge, which are open problems in robotics and AI.

This paper summarizes an architecture that seeks to address the challenges described above. It builds on the understanding that the robot's observations are a good source of information about the domain and the robot's action capabilities, and that reasoning and learning can bootstrap off each other. It is based on the following tenets:

- Knowledge elements include symbolic and numerical content encoding object constants, relations representing attributes and actions at different abstractions, and axioms composed of these relations.
- Action capabilities are 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 observed outcomes of actions.
- Learning is coupled with interaction, revising the perceived utility of actions through domain exploration performed actively or in response to unexpected transitions.

Our architecture incorporates these tenets by combining the principles of declarative programming, probabilistic graphical models, and reinforcement learning. This paper describes the following characteristics of our architecture:

- An action language describes tightly-coupled transition diagrams of the domain at two resolutions. The coarse-resolution description includes commonsense knowledge and incomplete knowledge of action capabilities, preconditions and effects. The fine-resolution description is defined as a refinement of the coarse-resolution one.
- For any given goal, non-monotonic logical reasoning with the coarse-resolution description provides a tentative plan of abstract actions. Each abstract action is implemented as a sequence of concrete actions by reasoning probabilistically over the relevant part of the fine-resolution representation. The outcomes of executing these concrete actions revise the coarse-resolution description.

- Interactive discovery of axioms governing action capabilities and domain dynamics, is posed as a relational reinforcement learning problem based on exploration performed actively or in response to unexpected transitions. Decision tree regression and sampling are used to find and generalize over candidate axioms, adding the generic axioms to the coarse-resolution description for reasoning.

In our architecture, the coarse-resolution representation is translated to an Answer Set Prolog (ASP) program, and a partially observable Markov decision process (POMDP) is constructed to reason probabilistically with the relevant part of the fine-resolution representation. Some of these capabilities are described in our papers (Sridharan 2016; Sridharan and Meadows 2017a; Sridharan, Meadows, and Gomez 2017; Sridharan et al. 2017). Here, we summarize the technical contributions, focusing on more recent work on representing, reasoning about and learning action capabilities, preconditions and effects. We report experimental results in simulation and on a mobile robot moving objects to specific places and people in an indoor domain.

2 Related Work

Many algorithms and architectures have been developed for representing and reasoning with different descriptions of knowledge, and for interactive learning. Classical logic-based representations and probabilistic models have been used widely in robotics and AI. More recently, non-monotonic logical reasoning paradigms such as ASP have also been used for robotics by an international community (Chen et al. 2012; Balduccini, Regli, and Nguyen 2014; Saribatur, Erdem, and Patoglu 2014). Formulations based on probabilistic representations (by themselves) make it difficult to reason with commonsense knowledge. Approaches based on classical first-order logic or non-monotonic logic, on the other hand, tend to require detailed prior knowledge of the domain and the agent's capabilities, and do not (by themselves) support a probabilistic representation of uncertainty. Different formalizations also exist for affordances based on combinations of attributes of the agent and the environment (Sahin et al. 2007) motivated by studies into how people judge the capabilities of others (Ramenzoni et al. 2010). Many computational approaches for reasoning about affordances use probabilistic perceptual descriptions (Shu, Ryoo, and Zhu 2016), but have difficulty using relational and declarative knowledge. Other approaches use logics to encode detailed relational domain knowledge and infer affordances (Gabaldon 2009), but do not support probabilistic models of uncertainty. Researchers have developed approaches that combine deterministic, logical, and/or probabilistic reasoning for robots (Kaelbling and Lozano-Perez 2013; Hanheide et al. 2015), including extensions to ASP (Lee and Wang 2015), and for reasoning about affordances (Sarathy and Scheutz 2016; Skarlatidis et al. 2015). However, existing approaches do not support one or more of the desired capabilities, e.g., expressiveness for commonsense knowledge, reasoning with large probabilistic components, or incremental knowledge revision.

In complex domains, agents often have to start with incomplete domain knowledge, and learn from observations of the environment. Early research used first-order logic and the observed effects of actions to learn causal laws, or to construct the preconditions or effects of actions (Shen and Simon 1989; Gil 1994). In addition to the limitations of first-order logic, these approaches 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 (Otero 2003). In recent times, *interactive task learning* has been proposed as a general approach for learning concepts from observations of the domain or human demonstrations and instructions (Kirk, Mininger, and Laird 2016). Interactive learning is often posed as an RL problem, and relational RL (RRL) supports efficient learning in dynamic domains using relational representations and regression for Q-function generalization (Driessens and Ramon 2003; Tadepalli, Givan, and Driessens 2004). However, existing RRL algorithms limit generalization to a single planning task, or do not support reasoning with commonsense knowledge.

As a step towards addressing the limitations of existing work, we have developed architectures that exploit the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning, as described in our papers (Sridharan 2016; Sridharan and Meadows 2017a; Sridharan et al. 2017). Here, we describe the overall architecture, focusing primarily on recent work on representing, reasoning about, and learning action capabilities, preconditions and effects. These capabilities are illustrated in the context of a robot finding and moving objects in an indoor domain.

3 Architecture Description

Figure 1 is a block diagram of the components of our architecture. We illustrate the components using the following running example.

Office Domain: Consider a robot that is assigned the goal of moving specific objects to specific places in an office domain. The domain under consideration contains:

- Sorts such as *place*, *thing*, *robot*, *object*, *textbook*, and *cup*, which include attributes such as *color* and *weight*, and are arranged hierarchically. Also, the domain has specific instances of sorts, e.g., instance rob_1 of *robot*.
- 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 *location* 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*.
- Some unknown information of interest may include:
  - A brittle object is damaged when it is put down.
  - Delivering an unlabeled object to a sales person causes it to be labeled.
  - A heavy object cannot be picked up by a robot with an electromagnetic arm.

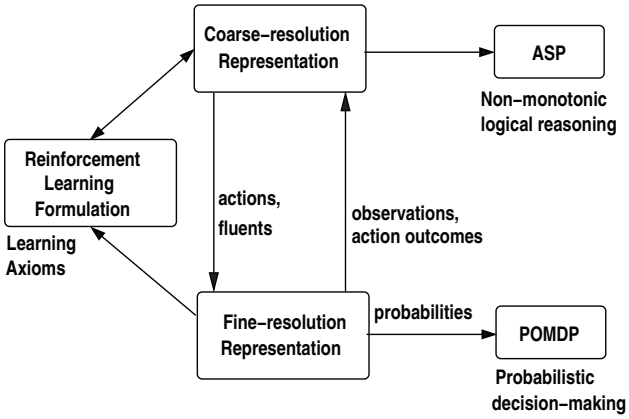

Figure 1: Architecture integrates the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning, for knowledge representation, reasoning, and learning, with qualitative and quantitative descriptions of knowledge and uncertainty.

– A brittle object cannot be labeled, except if the arm is electromagnetic.

In this domain, coarse-resolution reasoning considers the location of objects in places, while fine-resolution reasoning considers the location of objects in specific grid cells in these places, and reinforcement learning can be used to identify previously unknown axioms about the robot's action capabilities, and action preconditions and effects.

Action Language: The transition diagrams of our architecture's coarse-resolution and fine-resolution domain representations are described in an *action language* AL_d (Gelfond and Inclezan 2013), which has a sorted signature containing: *statics* (domain properties whose truth values cannot be changed by actions), $fluents$ (domain properties whose values can be changed by actions) and *actions* (elementary actions that can be executed). AL_d allows three types of statements: causal laws, state constraints and executability conditions, which we extend to include non-deterministic causal laws and definitions (Sridharan and Gelfond 2016).

3.1 Coarse-Resolution Planning and Diagnosis

The coarse-resolution domain representation has a system description $\mathscr{D}_H$ and histories with defaults $\mathscr{H}$. $\mathscr{D}_H$ consists of a sorted signature (Σ_H) that defines the names of objects, functions, and predicates available for use, and axioms to describe the coarse-resolution transition diagram τ_H. Examples of sorts in the office domain are *place*, *thing*, and *robot*. The fluents and actions are defined in terms of their arguments, e.g., in our domain, $loc(thing, place)$ and $in\_hand(robot, object)$ are some fluents, and $move(robot, place)$, $grasp(robot, object)$, $putdown(robot, object)$, and $serve(robot, object, person)$ are some actions. Axioms include causal laws such as:

$$move(R,Pl) \textbf{ causes } loc(R,Pl)$$
$$grasp(R,Ob) \textbf{ causes } in\_hand(R,Ob)$$

state constraints such as:

$$\neg loc(Ob,Pl_1) \textbf{ if } loc(R,Pl_2),\ Pl_1 \neq Pl_2$$
$$loc(Ob,Pl) \textbf{ if } loc(R,Pl),\ in\_hand(R,Ob)$$

and executability conditions such as:

$$\textbf{impossible } move(R,Pl) \textbf{ if } loc(R,Pl)$$
$$\textbf{impossible } grasp(R,Ob) \textbf{ if } loc(R,Pl_1),\ loc(Ob,Pl_2),\ Pl_1 \neq Pl_2$$
$$\textbf{impossible } grasp(R,Ob) \textbf{ if } in\_hand(R,Ob)$$

The recorded history of a dynamic domain is usually a record of (a) fluents observed to be true at a time step $obs(fluent, boolean, step)$, and (b) the occurrence of an action at a time step $hpd(action, step)$. Our architecture *expands on this view by allowing histories to contain (prioritized) defaults describing the values of fluents in their initial states*. For instance, the default "textbooks are typically in the main library. If a textbook is not there, it is in the office" can be represented elegantly as:

$$\textbf{initial default } loc(X,main\_library) \textbf{ if } textbook(X)$$
$$\textbf{initial default } loc(X,office) \textbf{ if } textbook(X),\ \neg loc(X,main\_library)$$

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:

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

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:

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

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: This coarse-resolution domain representation is transformed into a program $\Pi(\mathscr{D}_H, \mathscr{H})$ in CR-Prolog that incorporates consistency restoring (CR) rules in ASP (Balduccini and Gelfond 2003). ASP is based on stable model semantics and non-monotonic logics, and includes *default negation* and *epistemic disjunction*, e.g., unlike $\neg a$ that states *a is believed to be false*, *not a* only implies that *a is not believed to be true*, and unlike "$p \vee \neg p$" in propositional logic, "*p or* $\neg p$" is not a tautology. ASP can represent recursive definitions, defaults, causal relations, and constructs that are difficult to express in classical logic formalisms. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π; statements that hold in all such answer sets are program consequences. Algorithms for computing the entailment of CR-Prolog programs, and for planning and diagnostics, reduce these tasks to computing answer sets of CR-Prolog programs. Π consists of causal laws of $\mathscr{D}_H$, inertia axioms, closed world assumption for defined fluents, reality checks, and records of observations, actions, and defaults, from $\mathscr{H}$. 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—see (Sridharan et al. 2017) for formal definitions of states, entailment, and models for consistent inference.

In addition to planning, the architecture also supports reasoning about exogenous actions to explain the unexpected (observed) outcomes of actions. This is done by introducing *awareness* axioms, which guarantee that an inertial fluent's value is always known and reasoning takes into account actions that actually happened, and *reality check* axioms, which cause a contradiction when observations do not match expectations. Recovery from such contradictions is accomplished using CR rules that allow the robot to assume the occurrence of an exogenous action, under exceptional circumstances, to restore consistency. Explanation for unexpected symptoms is then (again) reduced to finding (and extracting suitable statements from) the answer set of the corresponding program (Gelfond and Kahl 2014). For more details about the explanation generation component, please see (Colaco and Sridharan 2015).

3.2 Fine-Resolution Probabilistic Planning

For any given goal, the answer set obtained by inference in the CR Prolog program (of the coarse-resolution representation) includes a sequence of abstract actions. The transition corresponding to each such action, $\langle \sigma_1, a^H, \sigma_2 \rangle$ of τ_H, is executed by reasoning probabilistically at a finer resolution. The fine-resolution reasoning includes four steps:

1. Define the fine-resolution version of the coarse-resolution system description.
2. Randomize the fine-resolution description, i.e., represent the non-determinism probabilistically.
3. Zoom to the part of this system description relevant to the desired transition T.
4. Construct a POMDP from the zoomed system description, solve POMDP, and use corresponding policy to execute a sequence of concrete actions.

The fine-resolution system description $\mathscr{D}_L$ has a sorted signature Σ_L and axioms that describe transition diagram τ_L. Unlike the coarse-resolution representation, the fine-resolution representation implicitly includes a history of observations and actions—the current state is assumed to be the result of all information obtained in previous time steps. Σ_L inherits the sorts, fluents, actions, and axioms from the coarse resolution signature and introduces new ones (or revised versions) that are viewed as components of their coarse-resolution counterparts. For instance, sorts *room* and *cell* are subsorts of *place*, while new fluent $loc(thing, cell)$ represents the cell location of things in the domain. Since action execution is considered to be non-deterministic in the fine-resolution representation, we introduce new fluents to keep track of observations, e.g., $observed(fluent, value, outcome)$, where $outcomes = \{true, false, undet\}$, keeps track of the observed values of specific fluents. New actions are also introduced, e.g., $test(robot, fluent, value)$ is used to test a fluent for a specific value. In addition, we define new statics to describe relations between the new sorts, and new axioms that describe the relations between the coarse-resolution elements and their fine-resolution counterparts. We specify a sequence of steps that defines the fine-resolution transition diagram as a *refinement* of the coarse-resolution diagram, and show that for every state transition $\langle \sigma_1, a^H, \sigma_2 \rangle$ in the coarse-resolution diagram, there is a path in the fine-resolution diagram from states that are refinements of σ_1 and σ_2 respectively.

The certainty of the robot's observations and the effects of the actions executed are only known with some degree of probability. We model this uncertainty by associating probabilities with the state transitions and observations in the fine-resolution system description, to obtain system description $\mathscr{D}_{LR}$. Since the fine-resolution states are only partially observable, reasoning uses *belief states*, probability distributions over the set of states. Reasoning over this probabilistic fine-resolution transition diagram becomes computationally intractable even for very simple problems. To execute any given abstract transition $\langle \sigma_1, a^H, \sigma_2 \rangle$ τ_H, the architecture therefore automatically *zooms* to $\mathscr{D}_{LR}(T)$, the part of the randomized, fine-resolution diagram relevant to the transition. This system description and the associated probabilities are used to construct a POMDP defined by the tuple $\langle S, A, Z, T, O, R \rangle$ for a specific goal state. The first three elements are the set of states, set of actions, and the set of values of observable fluents. The next two elements are the transition function $T : S \times A \times S' \rightarrow [0,1]$, which defines the probabilistic state transitions, and the observation function $O : S \times A \times Z \rightarrow [0,1]$, which defines the probability of observing the values of observable fluents by executing knowledge producing actions in specific states—the resul-

tant state is not considered because knowledge-producing actions do not change the state. Functions T and O (computed offline or learned online) describe a probabilistic transition diagram over the belief state. The reward specification $R : S \times A \times S' \rightarrow \Re$ is used to encode the relative cost or *utility* of taking specific actions in specific states, based on the goal state that is to be achieved. Planning involves computing a *policy* $\pi : b_t \rightarrow a_{t+1}$ that maximizes the cumulative reward over a planning horizon to map belief states to actions. The POMDP tuple is constructed using appropriate data structures such that existing (approximate) POMDP solvers can be used to obtain the policy. Plan execution uses the policy to repeatedly choose an action in the current belief state, and updates the belief state (through Bayesian update) after executing that action and receiving an observation. Eventually either the probability of one of the states exceeds a threshold (e.g., 0.9), or the robot identifies with high probability that the current task, i.e., current coarse-resolution transition, cannot be executed. The corresponding outcomes are added as statements to the history in the coarse-resolution description. For details, please see (Sridharan et al. 2017).

3.3 Reinforcement Learning

The robot uses the tightly-coupled coarse-resolution and fine-resolution representations (described above) to reason about the information extracted from sensor inputs, and to acquire new information about changes in object positions and configurations. The robot still has to augment existing knowledge and respond to domain changes. Consider the task of stacking books in the *main_library* in our illustrative domain, and assume that the axiom: "larger books cannot be stacked on smaller books" is not known to the robot. Generating and executing plans that do not take this axiom into account will result in the robot failing to accomplish the desired objective of stacking the books. The missing knowledge can be acquired from (say) a human or a repository. However, in complex domains, it is often difficult to obtain labeled training samples, and access to human knowledge may be limited. Also, any observed transition may be influenced by one or more past (or future) states and actions. We therefore focus on interactively acquiring 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. Figure 2 shows the control loop, where (for ease of explanation) we temporarily abstract the uncertainty in perception. 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, with the objective of identifying state-action pairs likely to lead to analogous "error" 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.

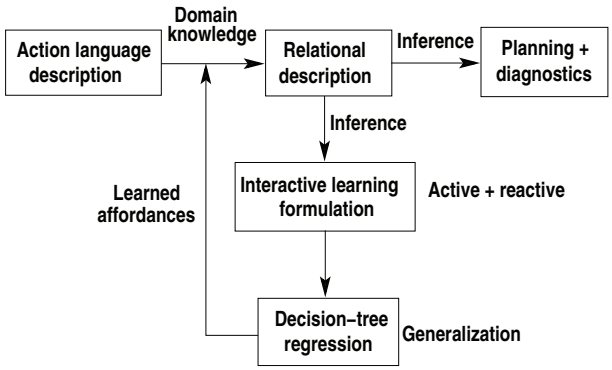

Figure 2: The closed loop of knowledge representation and reinforcement learning enables discovery of new axioms and their use in subsequent inference.

A standard RL formulation has an underlying Markov decision process (MDP) defined by the set of states (S), set of actions (A), the state transition function (T_f), and the reward function (R_f). In the RL formulation, T_f and R_f are unknown (to the robot), and each element in S grounds the domain attributes and whether the expected outcomes of the target action were observed. The values of state-action pairs are estimated in a series of episodes, until convergence, using the Q-learning algorithm (Sutton and Barto 1998). One key benefit of ASP-based reasoning is that we can define and 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 definition in (Sridharan et al. 2017).

In domains with complex relationships between objects, the state space may still be very large. Also, Q-learning and other standard RL algorithms (by themselves) do not generalize to relationally equivalent states, making it intractable to conduct RL trials in complex domains. To address these challenges, after one or more episodes of Q-learning, all visited state-action pairs and their estimated Q-values are used 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, and internal nodes correspond to boolean *tests* of specific domain attributes or actions, and determine the node's descendants. The revised tree is used to compute a new policy, eliminating the need to completely rebuild the tree after each episode. In each Q-learning episode, the system stochastically chooses to execute a random action or the one preferred by the current policy. Each action application also updates the information stored at a relevant leaf. Over time, the system assigns a higher value 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 BDT summarizes experiences from multiple MDPs.

The known structure of axioms is used to construct candidate axioms from the BDT. The system examines each leaf from the BDT, extracts a partial state-action description using its path to the root, and aggregates the attribute information from this description. Branches with low Q-values or corresponding to an action that did not result in the observed transition are eliminated. The system estimates the quality of each candidate from the Q-values of samples it has experienced. It makes a number of random sample draws from the BDT (without replacement); each sample is a full state description of information at the leaf and along the path to the root. The final step generalizes over the candidate axioms by eliminating candidates not refined by additional training samples after their construction, and candidates that elaborate other candidates. Finally, the remaining candidates undergo validation tests that 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 relational learning algorithm as "Q-RRL"—see (Sridharan and Meadows 2017a) for details.

4 Experimental Results

This section summarizes some experimental results in simulation and on physical robots to demonstrate the capabilities of the architecture—for more information, please see (Sridharan and Meadows 2017a; Sridharan et al. 2017). The simulator uses models that represent objects using probabilistic functions of features extracted from images, and models that reflect the robot's motion. The robot also collects data (e.g., computational time of different algorithms) in an initial training phase to define the probabilistic components of the fine-resolution domain representation.

First, consider an execution scenario in which the robot is in the $office$, and it is assigned the goal of moving a specific textbook tbk to the $office$. Based on default knowledge (about the location of textbooks) in the coarse-resolution representation, the robot creates a plan of abstract actions:

$$move(rob_1, main\_library)$$
$$grasp(rob_1, tbk)$$
$$move(rob_1, office)$$
$$putdown(rob_1, tbk)$$

where the robot rob_1 will have to search for tbk in the $main\_library$ before grasping it. Each action is executed probabilistically by constructing and solving the corresponding POMDP, as described above.

Next, consider the comparison of the proposed architecture (henceforth "PA") with just using POMDPs ("POMDP-1") in simulation trials. In these trials, the objective of the robot was to move specific objects (with unknown locations) to specific places in the domain. Note that POMDP-1 includes a hierarchical decomposition to make the task of solving the POMDPs computationally tractable (Zhang, Sridharan, and Wyatt 2015). The POMDP solver is given a fixed amount of time to compute action policies. An object's location in a cell is assumed to be known with certainty if the probabilistic belief (of the object's existence in the cell) exceeds a threshold (0.85). Experiments results indicate that as the domain size increases, it becomes computationally difficult to generate good POMDP action policies which, in conjunction with incorrect observations significantly impacts the ability to complete the trials. PA focuses the robot's attention on relevant rooms and cells to improve computational efficiency while still maintaining high accuracy—for larger domains, there is a drop in accuracy, but the impact is much less pronounced. The simulation trials also indicate that PA efficiently generates appropriate plans for domains with a large number of rooms and objects using only the knowledge relevant to the goal under consideration. As stated earlier, this relevant subset of the domain knowledge can be automatically selected using the relations in the coarse-resolution system description.

We also compared PA with POMDP-1 on a wheeled robot deployed on multiple floors of an office building. POMDP-1 takes 1.64 as much time as PA to move specific objects to specific places; this 39% reduction in execution time is statistically significant. Furthermore, we instantiated and evaluated our architecture in a different domain, e.g., of a robot waiter assisting in seating people and delivering orders in a restaurant. Results indicated that a purely probabilistic approach takes twice as much time as PA to locate and move objects to specific places. Videos of experimental trials can be viewed online: http://youtu.be/8zL4R8te6wg, https://vimeo.com/136990534

Finally, to evaluate the robot's ability to discover previously unknown rules, we designed multiple simulated trials in which the robot had to find and deliver objects to specific locations or people. Some axioms were intentionally hidden from the robot, resulting in failure to find plans, or unexpected outcomes under certain conditions. Specifically, we conducted tests in which the robot had to learn one or more of two positive and two negative affordances, two causal laws and two executability conditions. As stated earlier, we also conducted tests corresponding to active exploration and exploration in response to unexpected transitions. Rewards were provided by the simulator based on the success or failure of the plan to achieve the desired goal. The robot successfully identified all the missing axioms and adds them to the coarse-resolution system description. For instance, in the example where the robot had to find and move a cup to a specific location, the robot discovered the axiom about not being able to move a heavy cup with an electromagnetic arm. This axiom was encoded as:

$$\textbf{impossible}\ \ grasp(rob_1, C)\ \ \textbf{if}$$
$$aff\_forbids(id_1, grasp(rob_1, C))$$
$$aff\_forbids(id_1, grasp(rob_1, C))\ \ \textbf{if}\ \ obj\_weight(C, heavy),$$
$$arm\_type(rob_1, electromagnetic)$$

Including such newly discovered rules in the CR-Prolog program enables the robot to generate and successfully execute plans to achieve the desired goals.

We also explored the effect of the discovered axioms on the quality of plans generated. We conducted 1000 paired ASP-based planning trials for each axiom, and for all the

Condition	Causal laws		Executability conditions		(Dis)Affordances	
	Axiom 1	**Axiom 2**	**Axiom 3**	**Axiom 4**	**Axiom 5**	**Axiom 6**
Without axioms	101.4	0	37.9	164.4	29.7	121.3
With axiom	110.7	11.2	24.8	75.7	23.0	86.5

Table 1: Number of plans found without and with each of the target axioms under consideration. On average, discovering causal laws increases the number of feasible plans to achieve any given goal, whereas discovering executability conditions and disaffordances decreases the number of plans that can be used to achieve any given goal.

axioms, with and without the corresponding target axiom(s) in the system description. Table 1 summarizes the results for each of the six axiom (except the two for positive affordances), and displays some interesting trends. For instance, the set of plans found after including axioms that represent knowledge corresponding to a causal law (axiom 1 or 2 in this study) was a *superset* of the plans found without including these axioms in the system description. In other words, discovering previously unknown knowledge corresponding to a causal law (on average) increases the number of possible plans that can be constructed to achieve an assigned goal—similar results are obtained with positive affordances. On the other hand, the set of plans found after including axioms that represent knowledge of an executability condition or a forbidding affordance (axioms $3-6$ in this study) was a *subset* of the plans found without including these axioms. In other words, discovering knowledge corresponding to a previously unknown executability condition or forbidding affordance (on average) reduces the number of plans that can be executed to achieve an assigned goal. However, when axioms corresponding to different types of knowledge are considered together, the set of plans found after including these axioms is no longer a subset or superset of the plans found without including the axioms. For instance, when all six target axioms are considered together, all we can say is that 29.2 is the average magnitude of the difference in the number of plans found with and without including these axioms. Furthermore, we verified that all the plans that were computed after including all the target axioms were correct. Overall, experimental results indicated the reliable discovery of different axioms, robustness to noise, and improvement in plan quality—see (Sridharan and Meadows 2017b) for more details.

5 Conclusions

This paper described an architecture for knowledge representation, reasoning, and learning, in robotics, which combines the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning (RL). Tentative plans created by reasoning with commonsense knowledge in the coarse-resolution representation are implemented in the fine-resolution using probabilistic algorithms, adding relevant statements to the coarse-resolution history. Current beliefs and domain representations are used to formulate incremental and interactive learning of previously unknown domain axiom as an RL problem, using decision tree regression and sampling based on the underlying relational representations for efficiently identifying and generalizing over the learned axioms. Experimental results indicate that the architecture supports reasoning and learning at the sensorimotor level and the cognitive level, and scales well to large, complex domains. These capabilities are very important for robots collaborating with humans in complex application domains. Future work on the architecture will investigate: (a) tighter coupling of the logical and probabilistic reasoning components; and (b) extensive experimental evaluation on robots collaborating with humans in complex domains.

Acknowledgments

The architecture summarized in this paper was developed in collaboration with Michael Gelfond, Shiqi Zhang, Jeremy Wyatt, and Ben Meadows. This work was supported in part by the ONR Science of Autonomy award N00014-13-1-0766 and Asian Office of Aerospace Research and Development award FA2386-16-1-4071.

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.

Balduccini, M.; Regli, W. C.; and Nguyen, D. N. 2014. An ASP-Based Architecture for Autonomous UAVs in Dynamic Environments: Progress Report. In *International Workshop on Non-Monotonic Reasoning (NMR)*.

Chen, X.; Xie, J.; Ji, J.; and Sui, Z. 2012. Toward Open Knowledge Enabling for Human-Robot Interaction. *Journal of Human-Robot Interaction* 1(2):100–117.

Colaco, Z., and Sridharan, M. 2015. What Happened and Why? A Mixed Architecture for Planning and Explanation Generation in Robotics. In *Australasian Conference on Robotics and Automation (ACRA)*.

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

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.

Gelfond, M., and Kahl, Y. 2014. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press.

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

Hanheide, M.; Gobelbecker, M.; Horn, G.; Pronobis, A.; Sjoo, K.; Jensfelt, P.; Gretton, C.; Dearden, R.; Janicek, M.; Zender, H.; Kruijff, G.-J.; Hawes, N.; and Wyatt, J. 2015. Robot Task Planning and Explanation in Open and Uncertain Worlds. *Artificial Intelligence*.

Kaelbling, L., and Lozano-Perez, T. 2013. Integrated Task and Motion Planning in Belief Space. *International Journal of Robotics Research* 32(9-10):1194–1227.

Kirk, J.; Mininger, A.; and Laird, J. 2016. Learning Task Goals Interactively with Visual Demonstrations. *Biologicall Inspired Cognitive Architectures* 18:1–8.

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

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.

Sahin, E.; Cakmak, M.; Dogar, M. R.; Ugur, E.; and Ucoluk, G. 2007. To Afford or Not to Afford: A New Formalization of Affordances Towards Affordance-based Robot Control. *Adaptive Behavior* 15(4):447–472.

Sarathy, V., and Scheutz, M. 2016. A Logic-based Computational Framework for Inferring Cognitive Affordances. *IEEE Transactions on Cognitive and Developmental Systems* 8(3).

Saribatur, Z.; Erdem, E.; and Patoglu, V. 2014. Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans. In *International Conference on Intelligent Robots and Systems*, 2923–2930.

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

Shu, T.; Ryoo, M. S.; and Zhu, S.-C. 2016. Learning Social Affordance for Human-Robot Interaction. In *International Joint Conference on Articial Intelligence (IJCAI)*.

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. 2017a. A Combined Architecture for Discovering Affordances, Causal Laws, and Executability Conditions. In *International Conference on Advances in Cognitive Systems (ACS)*.

Sridharan, M., and Meadows, B. 2017b. Towards an Architecture for Discovering Domain Dynamics: Affordances, Causal Laws, and Executability Conditions. In *ICAPS Workshop on Planning and Robotics (PlanRob)*.

Sridharan, M.; Gelfond, M.; Zhang, S.; and Wyatt, J. 2017. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, http://arxiv.org/abs/1508.03891.

Sridharan, M.; Meadows, B.; and Gomez, R. 2017. What can I not do? Towards an Architecture for Reasoning about and Learning Affordances. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Sridharan, M. 2016. Towards An Architecture for Knowledge Representation, Reasoning and Learning in Human-Robot Collaboration. In *AAAI Spring Symposium on Enabling Computing Research in Socially Intelligent Human-Robot Interaction*.

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*.

Zhang, S.; Sridharan, M.; and Wyatt, J. 2015. Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds. *IEEE Transactions on Robotics* 31(3):699–713.