KR^3L: An Architecture for Knowledge Representation, Reasoning and Learning in Human-Robot Collaboration

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

Abstract

This paper describes an architecture that combines the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning. Reasoning with different descriptions of incomplete domain knowledge and uncertainty is based on tightly-coupled representations at two different resolutions. For any given goal, non-monotonic logical inference with the coarse-resolution domain representation provides a plan of abstract actions. Each abstract action is implemented as a sequence of concrete actions by reasoning probabilistically over a relevant part of the fine-resolution representation, committing high probability beliefs to the coarse-resolution representation. Unexplained plan step failures trigger relational reinforcement learning for incremental and interactive discovery of domain axioms. These capabilities are illustrated in simulated domains and on a physical robot in an indoor domain.

1 Introduction

Consider a robot assisting humans in locating and moving objects to specific places in an office with multiple rooms. While the robot typically needs considerable domain knowledge to perform these tasks, it is difficult for humans to provide comprehensive domain knowledge. The robot may be equipped with some commonsense knowledge, e.g., "books are usually in the library", and exceptions to this knowledge, e.g., "cookbooks are in the kitchen". In addition, the robot's actions are non-deterministic, and any information extracted from sensor inputs is likely to be incomplete and unreliable. To assist humans in such domains, the robot thus has to represent knowledge, reason, and learn, at both the sensorimotor level and the cognitive level. This objective maps to some fundamental challenges in knowledge representation, reasoning, and learning. For instance, the robot has to encode and reason with commonsense knowledge such that the semantics are readily accessible to humans, while also quantitatively modeling the uncertainty in sensing and actuation. Furthermore, for computational efficiency, the robot has to tailor sensing and actuation to tasks at hand, incrementally and interactively revising its knowledge over time. As a step towards addressing these challenges, the architecture described in this paper combines the knowledge representation and non-monotonic logical reasoning capabilities of declarative programming, with the uncertainty modeling capabilities of probabilistic graphical models, and the incremental and interactive learning capability of reinforcement learning. Key features of this architecture are:

- An action language describes transition diagrams of the domain at two resolutions, with the fine-resolution diagram being a *refinement* of the coarse-resolution diagram.
- For any given goal, non-monotonic logical reasoning with the coarse-resolution commonsense knowledge provides a tentative plan of abstract actions.
- Each abstract action is implemented probabilistically as a sequence of fine-resolution concrete actions, by *zooming* to the relevant part of the fine-resolution diagram and constructing suitable data structures.
- Unexplained plan step failures trigger incremental and interactive discovery of previously unknown domain axioms using relational reinforcement learning.

In our architecture, we translate the coarse-resolution representation to an Answer Set Prolog (ASP) program, and construct a partially observable Markov decision process (POMDP) for probabilistic planning. The architecture has been demonstrated to support reasoning with violation of defaults, noisy observations, and unreliable actions in large, complex domains [Colaco and Sridharan, 2015; Zhang *et al.*, 2015; Sridharan *et al.*, 2016; Zhang *et al.*, 2014]. Here, we summarize the technical contributions, and some results of experimental trials in simulation and on a mobile robot moving objects to specific places in an office domain.

2 Related Work

Knowledge representation, reasoning, and learning are well-researched areas in robotics and AI. Logic-based representations and probabilistic graphical models have been used to control sensing, navigation and interaction for robots and agents [Bai *et al.*, 2014; Galindo *et al.*, 2008]. Formulations based on probabilistic representations (by themselves) make it difficult to perform commonsense reasoning, whereas classical planning algorithms and logic programming tend to require considerable prior knowledge of the domain and the agent's capabilities. For instance, theories of reasoning about action and change, and the non-monotonic logical reasoning

ability of ASP have been used by an international community for controlling the behavior of one or more robots [Balduccini *et al.*, 2014; Saribatur *et al.*, 2014]. However, ASP does not support probabilistic representation of uncertainty, whereas a lot of information extracted from sensors and actuators is represented probabilistically.

Researchers have designed architectures that combine deterministic and probabilistic algorithms for task and motion planning [Kaelbling and Lozano-Perez, 2013], or combine a probabilistic extension of ASP with POMDPs for human-robot dialog [Zhang and Stone, 2015]. Recent work used a three-layered organization of knowledge and reasoning, and combined first-order logic with probabilistic reasoning for open world planning [Hanheide *et al.*, 2015]. Some popular formulations that combine logical and probabilistic reasoning include Markov logic network [Richardson and Domingos, 2006], Bayesian logic [Milch *et al.*, 2006], and probabilistic extensions to ASP [Baral *et al.*, 2009; Lee and Wang, 2015]. However, algorithms based on first-order logic do not provide the desired expressiveness—they do not support non-monotonic logical reasoning, and it is not always possible to express degrees of belief quantitatively, e.g., by adding probabilities to logic statements. Algorithms based on logic programming do not support all the desired capabilities such as incremental revision of (probabilistic) information, and reasoning with a large probabilistic component.

Many tasks that require an agent to learn from repeated interactions with the environment have been posed as Reinforcement Learning (RL) problems [Sutton and Barto, 1998]. As a step towards addressing fundamental challenges such as scaling and transfer of learned knowledge, relational RL (RRL) combines relational representations with regression for generalization [Dzeroski *et al.*, 2001]. Existing approaches, however, use RRL for planning, and generalization is limited to a single MDP for a specific planning task [Driessens and Ramon, 2003; Gartner *et al.*, 2003; Tadepalli *et al.*, 2004].

As a step towards addressing the challenges described above, we have developed architectures that couple declarative programming, probabilistic graphical models, and reinforcement learning [Sridharan *et al.*, 2016; 2015; Zhang *et al.*, 2015]. Here, we describe the architecture, and illustrate its capabilities in simulation and in the context of a physical robot assisting in an office domain.

3 Architecture Description

Figure 1 shows the components of our architecture. We illustrate the components using the following examples.

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

- Sorts such as *place*, *thing*, *robot*, and *object*, with *object* and *robot* being subsorts of *thing*. Sorts *textbook*, *printer* and *kitchenware*, are subsorts of *object*. Also, sorts for object attributes *color*, *shape*, and *size*.
- Four specific places: *office*, *main_library*, *aux_library*, and *kitchen*. We assume that these places are accessible from each other without the need to navigate any corridors, and that doors between these places are open.
- An instance of sort *robot*, called rob_1. Also, a number of instances of subsorts of the sort *object* in specific places.

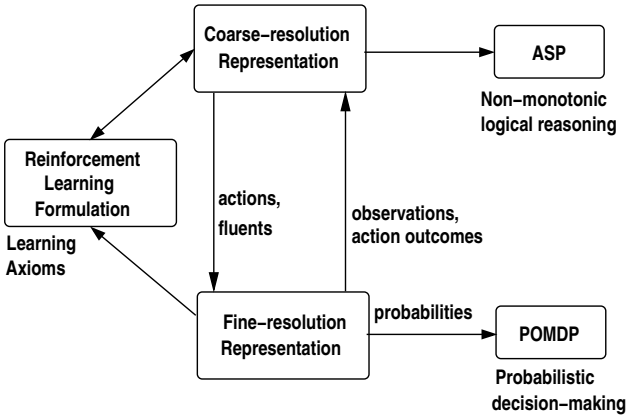

Figure 1: Architecture integrates the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning, for knowledge representation, reasoning, and learning, with qualitative and quantitative descriptions of knowledge and uncertainty.

In this domain, coarse-resolution reasoning considers the location of objects in places, while fine-resolution reasoning considers the location of objects in grid cells in these places. As an additional example used to illustrate the relational reinforcement learning, consider:

Blocks World (BW): a tabletop domain where the robot's objective is to stack blocks characterized by different colors, shapes, and sizes, in specific configurations on a table. The sorts of the BW domain include elements such as *block*, *place*, *color*, *shape*, *size*, and *robot*. A scenario with four blocks of the same size corresponds to ≈ 70 states under a standard RL/MDP formulation [Dzeroski *et al.*, 2001]. In this domain, the robot may not know, for instance, that no block can be placed on a prism-shaped block.

Action Language: The transition diagrams of our architecture's coarse-resolution and fine-resolution domain representations are described in an *action language* AL [Gelfond and Kahl, 2014]. AL has a sorted signature containing three *sorts*: *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 in parallel). AL allows three types of statements: causal laws, state constraints and executability conditions.

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 example 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 inertial fluents[1], and $move(robot, place)$, $grasp(robot, object)$, $putdown(robot, object)$, and $put(object, object)$ are some actions. Examples of 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 auxiliary library. If the textbook is not there either, it is in the office" can be represented elegantly as:

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

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 [Gelfond and Kahl, 2014]. 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.*, 2015; Zhang *et al.*, 2014] for formal definitions of states, entailment, and models for consistent inference.

In addition to planning, the architecture supports reasoning about exogenous actions to explain the unexpected (observed) outcomes of actions [Balduccini and Gelfond, 2003]. For instance, to reason about a door between two rooms being locked unexpectedly (e.g., by a human), we introduce exogenous action $locked(door)$ and add the axioms:

$$is\_open(D) \leftarrow open(R, D),\ \neg ab(D)$$
$$ab(D) \leftarrow locked(D)$$

where a door is considered *abnormal*, i.e., $ab(D)$, if it has been locked, say by a human. Actions and suitable axioms are included for other situations in a similar manner. We also introduce an *explanation generation* rule and a new relation *expl* as follows:

$$occurs(A, I) \mid \neg\, occurs(A, I) \leftarrow exogenous\_action(A),\ I < n$$
$$expl(A, I) \leftarrow action(exogenous, A),\ occurs(A, I),\ not\ hpd(A, I)$$

where *expl* holds if an exogenous action is hypothesized but there is no matching record in the history. We also include *awareness* axioms and *reality check* axioms:

$$\%\ \text{awareness axiom}$$
$$holds(F, 0)\ or\ \neg\, holds(F, 0) \leftarrow fluent(basic, F)$$
$$occurs(A, I) \leftarrow hpd(A, I)$$
$$\%\ \text{reality checks}$$
$$\leftarrow obs(fluent, true, I),\ \neg\, holds(fluent, I)$$
$$\leftarrow obs(fluent, false, I),\ holds(fluent, I)$$

The awareness axioms guarantee that an inertial fluent's value is always known, and that reasoning takes into account actions that actually happened. The reality check axioms cause a contradiction when observations do not match expectations, and the explanation for such unexpected symptoms can be reduced to finding (and extracting suitable statements from) the answer set of the corresponding program [Gelfond and Kahl, 2014]. The new knowledge is included in the ASP program and used for subsequent inference. This approach provides *all* explanations of an unexpected symptom. The other option is to use a CR rule instead of the explanation generation rule:

$$occurs(A, I) \stackrel{+}{\leftarrow} exogenous\_action(A),\ I < n$$

where the robot assumes the occurrence of an exogenous action, under exceptional circumstances, to restore consistency.

[1] Inertial fluents obey the laws of inertia and can be changed directly by actions, while defined fluents are not subject to inertia axioms and cannot be changed directly by an action.

the set of CR rules with the smallest cardinality is considered to be the *minimal* explanation. The architecture also includes a similar approach (with CR rules) to reason about partial scene descriptions, e.g., properties of objects and events, extracted from sensor inputs such as camera images. Given ideal descriptions of domain objects, and partial descriptions extracted from sensor input, candidate explanations are sets of CR rules that can be triggered to explain the descriptions, the set with lowest cardinality is the minimal explanation—see [Colaco and Sridharan, 2015] for more details.

3.2 Fine-Resolution Probabilistic Planning

For any given goal, the answer set obtained by ASP-based coarse-resolution inference includes a sequence of abstract actions. Each such action a^H in state σ of τ_H is executed by probabilistic reasoning at a fine-resolution. This reasoning includes three steps:

1. Define the fine-resolution version of the coarse-resolution transition diagram and randomize it.
2. Zoom to the part of the randomized fine-resolution transition diagram that is relevant to the execution of a^H.
3. Construct a POMDP from the zoomed part, solve it to obtain a policy, and use 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., $observed(fluent, value, outcome)$, with $outcomes = \{true, false, undet\}$, 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 such that, for every state transition $T = \langle \sigma, a^H, \sigma' \rangle$ in τ_H, there is a path in τ_L from state s compatible with σ, to some state compatible with σ'—see [Sridharan and Gelfond, 2016] for details.

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 *randomizing* $\mathscr{D}_L$, i.e., by replacing the deterministic causal laws in $\mathscr{D}_L$ by non-deterministic ones and modifying the signature to declare each affected fluent as a random fluent. The randomized system description $\mathscr{D}_{LR}$ is used in semi-supervised experimental trials to collect statistics and compute the probabilities of action outcomes and reliability of observations. Reasoning probabilistically over $\mathscr{D}_{LR}$ can result in incorrect behavior and be computationally intractable for complex domains. To execute any given abstract action a^H in state σ of τ_H, the architecture thus *zooms* to $\mathscr{D}_{LR}(T)$, the part of $\mathscr{D}_{LR}$ that is relevant to the execution of a^H—see [Sridharan and Gelfond, 2016] for details.

Next, $\mathscr{D}_{LR}(T)$ is used to construct a POMDP defined by the tuple $\langle S^L, A^L, Z^L, T^L, O^L, R^L \rangle$ for a specific goal state. The first three elements are the set of states, actions, and the values of observable fluents. The next two elements are the transition function $T^L : S^L \times A^L \times S'^L \rightarrow [0,1]$, which defines the probabilistic state transitions, and the observation function $O^L : S^L \times A^L \times Z^L \rightarrow [0,1]$, which defines the probability of observing the values of observable fluents by executing knowledge producing actions in specific states—these actions do not change the physical state. Functions T^L and O^L describe a probabilistic transition diagram over the belief state using the statistics collected experimentally. The reward specification $R^L : S^L \times A^L \times S'^L \rightarrow \Re$ 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. Since the true state is partially observable, planning computes a *policy* $\pi : b_t \rightarrow a_{t+1}$ that maximizes the cumulative reward over a planning horizon to map belief states, i.e., probability distributions over the states, to actions. The POMDP tuple is constructed using data structures that allow the use of existing (approximate) POMDP solvers. Plan execution uses the policy to repeatedly choose an action in the current belief state, and updates the belief state after executing that action and receiving an observation:

$$b_{t+1}(s_{t+1}) \propto O(s_{t+1}, a_{t+1}, o_{t+1}) \sum_s T(s, a_{t+1}, s_{t+1}) \cdot b_t(s)$$

Policy execution is terminated by a transition to a terminal state. In our case, this transition occurs because the probability of one of the states is very high, or because none of the states are very likely and there is no value in executing the policy further—the latter case is interpreted as the inability to execute a^H. The corresponding action outcomes are added as statements to the history in the coarse-resolution description—see [Sridharan *et al.*, 2015] for details about constructing and solving the POMDP.

Constructing and solving a POMDP can become computationally inefficient for complex domains, e.g., rooms with many cells connected to many other rooms, even with sophisticated POMDP solvers. To address this problem, we have explored reasoning in ASP at a finer resolution (e.g., areas in places instead of places), with selective grounding of the variables [Colaco and Sridharan, 2015]. Our prior work has also explored hierarchical decompositions of POMDPs for reliable and efficient operation [Sridharan *et al.*, 2010].

3.3 Reinforcement Learning

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

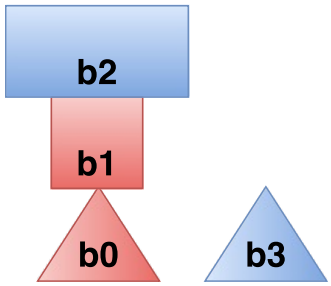

Figure 2: Illustrative example of a planned goal state that the robot cannot achieve.

axiom into account will result in the robot failing to accomplish the desired objective of stacking the books. Similarly, in the BW domain, a robot that does not know the axiom "no block can be placed on a prism-shaped block", and asked to stack any three of the four blocks placed on a table, may attempt to reach the goal configuration shown in Figure 2. The first action in the corresponding plan: $move(b_1, b_0)$ followed by $move(b_2, b_1)$, will result in a failure that cannot be explained. Our architecture supports incremental discovery of unknown axioms governing domain dynamics, by integrating relational reinforcement learning (RRL). The current beliefs of the robot, and the system descriptions at the two resolutions, are used to formulate the task of incrementally discovering domain axioms as an RL problem. Ideally, the state should be estimated using the fine-resolution representation and the corresponding (POMDP) belief states. However, to explore the feasibility of RRL for discovering axioms, our current RRL implementation abstracts away the uncertainty in perception and consider the corresponding MDP. Furthermore, we currently focus on discovering executability conditions for actions in the coarse-resolution description, i.e., axioms that encode the conditions under which each specific action cannot be executed.

Axioms in the coarse-resolution ASP program eliminate impossible state transitions in the RL formulation. In the RL formulation, the goal (achieving which provides high rewards) is set to be the state that resulted in the unexplained plan step failure. Over repeated episodes of Q-learning, the relative values of different state-action pairs (i.e., the Q-values) are computed. Once the Q-values converge, this approach can identify specific ground actions that should not be attempted. However, these axioms may conflict with existing axioms, or include specific instances of more general axioms. Conflicts can be identified as inconsistencies in the answer set of the corresponding ASP program. To discover the general axioms, we first support generalization within the MDP, using state-action pairs visited in a set of episodes to construct a binary decision tree—each path from the root to a leaf corresponds to a state-action pair, and individual nodes are specific fluents. This tree is used to provide a policy for the subsequent episode(s). When Q-learning is terminated, this tree relationally represents the robot's experiences. The second step simulates similar errors (to the one that triggered RRL) and considers the corresponding MDPs. The Q-value of a state-action pair is now the the weighted average of the values across different MDPs—the weight is inversely proportional to the distance to the goal state based on the optimal policy for the MDP. These similar MDPs may be chosen using the information encoded in the ASP program. The third step identifies candidate axioms by constructing training samples based on specific actions and the corresponding related fluents encoded in the binary decision tree. These training samples are used to construct a decision tree in which each path from the root node to a leaf is a candidate executability condition. The final step considers all candidate axioms for different actions, and uses K-means algorithm to cluster these candidates based on their value. The axioms that fall within the cluster with the largest mean are considered to represent generalized axioms, and are added to the ASP program to be used in the subsequent steps—see [Sridharan *et al.*, 2016] for details about our RRL approach.

4 Summary of 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 [Colaco and Sridharan, 2015; Sridharan *et al.*, 2015; Sridharan and Gelfond, 2016; Sridharan *et al.*, 2016]. The simulator uses models that represent objects using probabilistic functions of features extracted from images, and models that reflect the robot's motion and perception capabilities.

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), 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 *et al.*, 2013]. 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). The robot's ability to successfully complete the task is shown in Figure 3 as a function of the number of cells in the domain; each data point is the average of 1000 trials, and each room has four cells. As the number of cells increases, it becomes computationally difficult to generate good POMDP action policies that, in conjunction with incorrect observations, significantly impacts the ability to complete the trials. PA focuses the

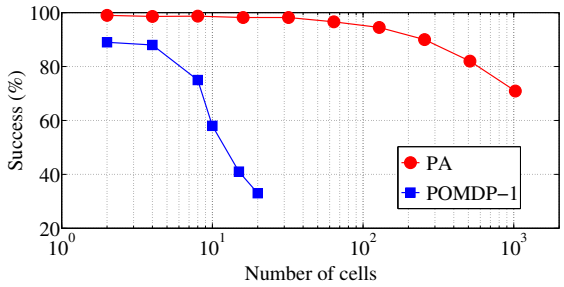

Figure 3: With limited policy computation time, PA is much more accurate than POMDPs as the number of cells increases.

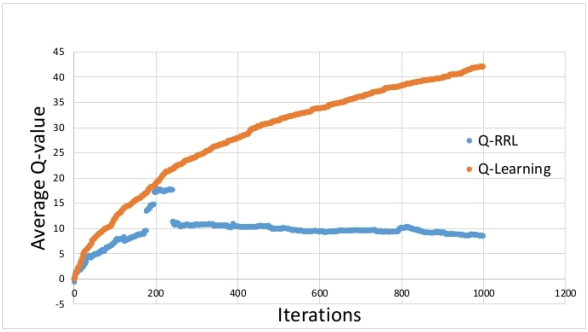

Figure 4: Comparing the rate of convergence of Q-RRL with that of Q-learning—Q-RRL converges much faster.

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 time taken by PA to generate a plan was also computed as a function of the domain size (characterized as the number of rooms and objects). PA generates appropriate plans for domains with a large number of rooms and objects. Using only the knowledge relevant to the goal significantly reduces the planning time in comparison with using all the domain knowledge available. This relevant part of the domain knowledge can be identified using the relations encoded in the coarse-resolution 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`

Next, to evaluate the robot's ability to discover previously unknown executability conditions, we designed multiple simulated trials in which the robot had to arrange objects in specific configurations. Some axioms were hidden from the robot, resulting in failure when certain intermediate configurations were reached. Rewards were provided by the simulator based on the success or failure of the plan. The robot successfully identified actions that could not be executed, and added suitable axioms to the coarse-resolution system description. For instance, in the BW domain, Figure 4 shows the rate of convergence of the average Q-values obtained using Q-RRL (i.e., our approach for relational reinforcement learning) is much better than that of Q-learning. It does not matter whether the actual average Q-values of Q-Learning are higher or lower than those of Q-RRL. In the BW domain, the robot also successfully discovered that no object should be stacked on a prism-shaped object:

$$\textbf{impossible}\ \ move(A,D)\ \ \textbf{if}\ \ has\_shape(D,prism)$$

In a similar fashion, in the context of stacking books in the office domain, the robot discovered that bigger books should not be stacked on smaller ones:

$$\textbf{impossible}\ put(B_1,B_2)\ \ \textbf{if}\ \ bigger(B_1,B_2),\ textbook(B_1),\ textbook(B_2).$$

Including such axioms in the ASP program enables the robot to generate plans that can be successfully executed to achieve the desired goal state, e.g., stacking of blocks or books in a desired configuration. For additional experimental results of evaluating our RRL approach, see [Sridharan *et al.*, 2016].

5 Conclusions

This paper described an architecture that combines the complementary strengths of declarative programming, probabilistic graphical models, and relational reinforcement learning (RRL). Tentative plans created by reasoning with commonsense knowledge in the coarse-resolution are implemented in the fine-resolution using probabilistic algorithms, adding statements to the coarse-resolution history. The incremental and interactive discovery of previously unknown domain axioms is formulated as an RRL problem. Experimental results indicate that the architecture supports reasoning and learning at the sensorimotor level and the cognitive level, and scales well to complex domains. These capabilities are important for robots collaborating with humans. Future work on the architecture will explore tighter coupling of the logical and probabilistic reasoning, and extensive trials on robots collaborating with humans in different domains.

Acknowledgments

The architecture summarized in this paper is based on collaboration with Michael Gelfond, Jeremy Wyatt, Shiqi Zhang, Zenon Colaco, Rashmica Gupta, and Prashanth Devarakonda. This work was supported in part by 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 author.

References

[Bai *et al.*, 2014] Haoyu Bai, David Hsu, and Wee Sun Lee. Integrated Perception and Planning in the Continuous Space: A

POMDP Approach. *International Journal of Robotics Research*, 33(8), 2014.

[Balduccini and Gelfond, 2003] Marcello Balduccini and Michael Gelfond. Diagnostic Reasoning with A-Prolog. *Theory and Practice of Logic Programming*, 3(4-5):425–461, 2003.

[Balduccini *et al.*, 2014] Marcello Balduccini, William C. Regli, and Duc N. Nguyen. An ASP-Based Architecture for Autonomous UAVs in Dynamic Environments: Progress Report. In *International Workshop on Non-Monotonic Reasoning (NMR)*, Vienna, Austria, July 17-19, 2014.

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, 9(1):57–144, January 2009.

[Colaco and Sridharan, 2015] Zenon Colaco and Mohan Sridharan. What Happened and Why? A Mixed Architecture for Planning and Explanation Generation in Robotics. In *Australasian Conference on Robotics and Automation (ACRA)*, Canberra, Australia, December 2-4, 2015.

[Driessens and Ramon, 2003] Kurt Driessens and Jan Ramon. Relational Instance-Based Regression for Relational Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, pages 123–130. AAAI Press, 2003.

[Dzeroski *et al.*, 2001] Saso Dzeroski, Luc De Raedt, and Kurt Driessens. Relational Reinforcement Learning. *Machine Learning*, 43:7–52, 2001.

[Galindo *et al.*, 2008] Cipriano Galindo, Juan-Antonio Fernandez-Madrigal, Javier Gonzalez, and Alessandro Saffioti. Robot Task Planning using Semantic Maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[Gartner *et al.*, 2003] Thomas Gartner, Kurt Driessens, and Jan Ramon. Graph Kernels and Gaussian Processes for Relational Reinforcement Learning. In *International Conference on Inductive Logic Programming (ILP)*, pages 140–163. Springer, 2003.

[Gelfond and Kahl, 2014] Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press, 2014.

[Hanheide *et al.*, 2015] Marc Hanheide, Moritz Gobelbecker, Graham Horn, Andrzej Pronobis, Kristoffer Sjoo, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, Hendrik Zender, Geert-Jan Kruijff, Nick Hawes, and Jeremy Wyatt. Robot Task Planning and Explanation in Open and Uncertain Worlds. *Artificial Intelligence*, 2015.

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

[Lee and Wang, 2015] Joohyung Lee and Yi Wang. A Probabilistic Extension of the Stable Model Semantics. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, March 2015.

[Milch *et al.*, 2006] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *Statistical Relational Learning*. MIT Press, 2006.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, February 2006.

[Saribatur *et al.*, 2014] Zeynep Saribatur, Esra Erdem, and Volkan Patoglu. Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[Sridharan and Gelfond, 2016] Mohan Sridharan and Michael Gelfond. Using Knowledge Representation and Reasoning Tools in the Design of Robots. In *IJCAI Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS)*, New York, USA, July 10, 2016.

[Sridharan *et al.*, 2010] Mohan Sridharan, Jeremy Wyatt, and Richard Dearden. Planning to See: A Hierarchical Approach to Planning Visual Actions on a Robot using POMDPs. *Artificial Intelligence*, 174:704–725, 2010.

[Sridharan *et al.*, 2015] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, Unrefereed CoRR abstract: `http://arxiv.org/abs/1508.03891`, August 2015.

[Sridharan *et al.*, 2016] Mohan Sridharan, Prashanth Devarakonda, and Rashmica Gupta. Discovering Domain Axioms Using Relational Reinforcement Learning and Declarative Programming. In *ICAPS Workshop on Planning and Robotics (PlanRob)*, London, UK, June 13-14, 2016.

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

[Tadepalli *et al.*, 2004] Prasad Tadepalli, Robert Givan, and Kurt Driessens. Relational Reinforcement Learning: An Overview. In *Relational Reinforcement Learning Workshop at the International Conference on Machine Learning*, 2004.

[Zhang and Stone, 2015] Shiqi Zhang and Peter Stone. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *AAAI Conference on Artificial Intelligence*, pages 1394–1400, Austin, USA, 2015.

[Zhang *et al.*, 2013] Shiqi Zhang, Mohan Sridharan, and Christian Washington. Active Visual Planning for Mobile Robot Teams using Hierarchical POMDPs. *IEEE Transactions on Robotics*, 29(4):975–985, 2013.

[Zhang *et al.*, 2014] Shiqi Zhang, Mohan Sridharan, Michael Gelfond, and Jeremy Wyatt. Towards An Architecture for Knowledge Representation and Reasoning in Robotics. In *International Conference on Social Robotics (ICSR)*, pages 400–410, Sydney, Australia, October 27-29, 2014.

[Zhang *et al.*, 2015] Shiqi Zhang, Mohan Sridharan, and Jeremy Wyatt. Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.