Towards An Architecture for Representation, Reasoning, and Learning in Human-Robot Collaboration

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

Abstract

Robots collaborating with humans need to represent knowledge, reason, and learn, at the sensorimotor level and the cognitive level. This paper summarizes the capabilities of an architecture that combines the complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning, to represent, reason with, and learn from, qualitative and quantitative descriptions of incomplete domain knowledge and uncertainty. Representation and reasoning is based on two tightly-coupled domain representations at different resolutions. For any given task, the coarse-resolution symbolic domain representation is translated to an Answer Set Prolog program, which is solved to provide a tentative plan of abstract actions, and to explain unexpected outcomes. Each abstract action is implemented by translating the relevant subset of the corresponding fine-resolution probabilistic representation to a partially observable Markov decision process (POMDP). Any high probability beliefs, obtained by the execution of actions based on the POMDP policy, update the coarse-resolution representation. When incomplete knowledge of the rules governing the domain dynamics results in plan execution not achieving the desired goal, the coarse-resolution and fine-resolution representations are used to formulate the task of incrementally and interactively discovering these rules as a reinforcement learning problem. These capabilities are illustrated in the context of a mobile robot deployed in an indoor office domain.

1 Introduction

Consider a robot assisting humans by locating and moving specific objects to specific places in an office with multiple rooms. While it is difficult for such a robot to operate without considerable domain knowledge, it is also difficult for humans to provide complete domain knowledge or elaborate feedback. The robot may be equipped with some commonsense knowledge, e.g., "books are usually in the library", and some exceptions to this knowledge that may be known or unknown to the robot, e.g., "cookbooks are in the kitchen", and "manuals are in the laboratory". In addition, the robot's actions are non-deterministic, and any information extracted by processing the data from sensors mounted on the robot, provides a partial and unreliable domain description. To assist 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 to support reliable operation. Furthermore, for efficient operation, the robot has to tailor sensing and actuation to tasks at hand, incrementally and interactively revising the existing knowledge in response to unexpected changes.

As a step towards addressing the challenges described above, 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 learning capability of reinforcement learning (RL). Key features of this architecture are:

- An action language is used for describing a coarse-resolution and a fine-resolution transition diagram for the target domain. The fine-resolution diagram is defined as a refinement of the coarse-resolution diagram. The coarse-resolution domain representation also includes a history with initial-state default knowledge.
- For any given goal, non-monotonic logical reasoning with the coarse-resolution representation provides a tentative plan of abstract actions. Each abstract action is implemented probabilistically based on the fine-resolution representation, with the corresponding action outcomes revising the coarse-resolution representation.
- For any abstract action, tight coupling between the two diagrams enables the robot to probabilistically represent just the relevant subset of the fine-resolution diagram, and use this probabilistic diagram to plan and execute a sequence of concrete actions to implement the abstract action.
- The current beliefs and the domain representations are used to provide a reinforcement learning formulation of the task of incrementally discovering previously unknown rules governing domain dynamics, using these discovered rules for subsequent reasoning.

In our architecture, we translate the coarse-resolution representation to an Answer Set Prolog (ASP) program, and use the probabilistic version of the relevant subset of the fine-resolution representation to construct a partially observable Markov decision process (POMDP). The architecture thus supports reasoning with violation of defaults, noisy observations and unreliable actions in large, complex domains. Subsets of these capabilities have been reported in our previous papers (Colaco and Sridharan 2015; Zhang, Sridharan, and Wyatt 2015; Sridharan and Rainge 2014; Zhang et al. 2014). Here, we summarize the technical contributions, and the result of experimental trials in simulation and on a mobile robot moving objects to specific places in an office domain.

2 Related Work

Knowledge representation, planning and explanation generation are well-researched areas in human-robot collaboration, and in artificial intelligence. Logic-based representations and probabilistic graphical models have been used to control sensing, navigation and interaction for robots and agents (Bai, Hsu, and Lee 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, and make it difficult to merge new, unreliable information with an existing knowledge base. For instance, theories of reasoning about action and change, and the non-monotonic logical reasoning ability of ASP (Gelfond and Kahl 2014) have been used by an international research community for reasoning by simulated robot housekeepers (Erdem, Aker, and Patoglu 2012), natural language human-robot interaction (Chen et al. 2012), control of unmanned aerial vehicles (Balduccini, Regli, and Nguyen 2014), and coordination of robot teams (Saribatur, Erdem, and Patoglu 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), couple declarative programming and continuous-time planners for path planning in robot teams (Saribatur, Erdem, and Patoglu 2014), 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 (instance, default and diagnostic), and a three-layered architecture (competence, belief, and deliberative layers), which combines first-order logic and probabilistic reasoning for open world planning on robots (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, Gelfond, and Rushton 2009; Lee and Wang 2015). However, algorithms based on first-order logic do not provide the desired expressiveness, e.g., it is not always possible to express degrees of belief quantitatively. Algorithms based on logic programming do not support one or more of the desired capabilities such as incremental revision of (probabilistic) information; reasoning as in causal Bayesian networks; and reasoning with large probabilistic components. As a step towards addressing these limitations, we have developed architectures that couple declarative programming, probabilistic graphical models, and reinforcement learning (Sridharan et al. 2015; Sridharan and Rainge 2014; Zhang, Sridharan, and Wyatt 2015). Here, we describe the overall architecture, and illustrate its capabilities in the context of a robot finding and moving objects in an office 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:

- The sorts: *place*, *thing*, *robot*, and $object$, with $object$ and *robot* being subsorts of *thing*. Sorts *textbook*, *printer* and *kitchenware*, are subsorts of the sort $object$. We also have 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.

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 rules about object configurations.

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,

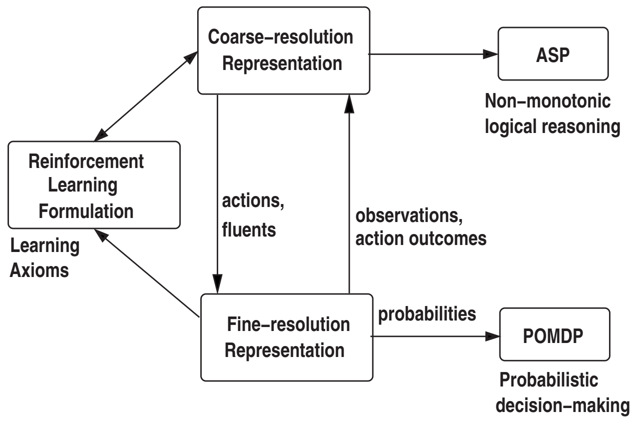
Coarse-resolution Representation
ASP
Non-monotonic logical reasoning
Reinforcement Learning Formulation
Learning Axioms
actions, fluents
observations, action outcomes
Fine-resolution Representation
probabilities
POMDP
Probabilistic decision-making

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.

$loc(thing, place)$ 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:

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

state constraints such as:

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

and executability conditions such as:

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

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:

$$\begin{aligned} &\textbf{initial default} \ loc(X, main\_library) \ \textbf{if} \ textbook(X) \\ &\textbf{initial default} \ loc(X, aux\_library) \ \textbf{if} \ textbook(X), \\ &\qquad\qquad\qquad\qquad \neg loc(X, main\_library) \\ &\textbf{initial default} \ loc(X, office) \ \textbf{if} \ textbook(X), \\ &\qquad\qquad\qquad\qquad \neg loc(X, main\_library), \\ &\qquad\qquad\qquad\qquad \neg loc(X, aux\_library) \end{aligned}$$

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:

$$\begin{aligned} is\_open(D) \ &\leftarrow \ open(R, D), \ \neg ab(D) \\ ab(D) \ &\leftarrow \ locked(D) \end{aligned}$$

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:

$$\begin{aligned} occurs(A, I) \mid \neg \, occurs(A, I) \ &\leftarrow exogenous\_action(A) \\ &\qquad I \ < \ n \\ expl(A, I) \ &\leftarrow action(exogenous, A), \\ &\qquad occurs(A, I), \ not \ hpd(A, I) \end{aligned}$$

where *expl* holds if an exogenous action is hypothesized but there is no matching record in the history. We also include

[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.

awareness axioms and *reality check* axioms:

% awareness axiom
$holds(F,0)\ \ or\ \ \neg\, holds(F,0)\ \leftarrow\ fluent(basic,F)$
$occurs(A,I)\ \leftarrow\ hpd(A,I)$
% 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 is allowed to assume the occurrence of an exogenous action, under exceptional circumstances, to restore consistency. A partial ordering is defined over sets of CR rules, based on the cardinality of sets, and the set 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 inference in the CR Prolog program (of the coarse-resolution representation) includes a sequence of abstract actions. Each such action a^H in state σ of $\tau_H$is executed by reasoning probabilistically at a finer resolution. The fine-resolution reasoning includes three steps:

1. Define the fine-resolution version of the coarse-resolution transition diagram.
2. Identify and randomize (i.e., represent probabilistically) the subset of the fine-resolution transition diagram that is relevant to the execution of a^H.
3. Construct a POMDP from the randomized subset of the diagram, solve POMDP 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., $observed(fluent,value,outcome)$, 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 $\langle\sigma,a,\sigma'\rangle$ in the coarse-resolution diagram, there is a path in the fine-resolution diagram from state s compatible with σ, to some state compatible with σ'.

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 diagram. 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 action a^H in state σ of τ_H, the architecture therefore identifies the relevant sorts, fluents and axioms in the coarse-resolution diagram, and thus identifies the subset of the fine-resolution transition diagram that needs to be represented probabilistically.

The probabilistic version of the subset of the fine-resolution transition diagram relevant to the execution of a^H is 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 resultant 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:

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

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 action a^H, cannot be executed. The corresponding action outcomes are added as observation statements to the history in the coarse-resolution description.

Constructing and solving a POMDP can become computationally intractable as the state space grows, e.g., rooms with many cells connected to many other rooms, even with state of the art approximate 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. Probabilistic reasoning then reduces to simple Bayesian belief updates over the relevant subset of the domain. We observed that the choice of resolution for symbolic and probabilistic reasoning presents an interesting trade-off between representational elegance, accuracy of decision making, and computational efficiency—see (Colaco and Sridharan 2015) for more details.

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 rule: "larger books cannot be stacked on smaller books" is not known to the robot. Generating and executing plans that do not take this rule into account will result in the robot failing to accomplish the desired objective of stacking the books. Our architecture supports incremental discovery of such (previously) unknown rules, and revision of existing rules, governing domain dynamics, by integrating reinforcement learning (RL) with the domain representations—Figure 2 shows the control loop.

When plan execution repeatedly fails to achieve the desired objective, it is hypothesized that this outcome may be due to previously unknown rules governing domain dynamics. The current beliefs of the robot, and the coarse-resolution and fine-resolution domain representations, are used to formulate the task of incrementally learning the previously unknown domain rules as an RL problem. For instance, axioms in the CR-Prolog program corresponding to the coarse-resolution representation eliminates impossible states, actions and state transitions in the RL formulation. The state transition function is based on the corresponding entries in the coarse-resolution and fine-resolution system descriptions. The reward specification is based on global objectives and supports the use of simple high-level feedback,

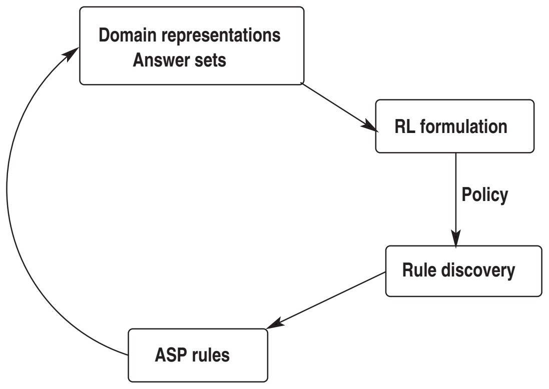

Figure 2: The closed loop of knowledge representation and reinforcement learning enables discovery of new rules and their use in subsequent inference.

e.g., positive/negative or "yes/no" reinforcement, which can be provided even by non-expert human participants. This formulation is used to set up exploration and exploitation trials in which the robot incrementally learns the relative value of specific state-action pairs, i.e., the Q-value function. At each step, state is estimated using the fine-resolution representation and the corresponding (POMDP) belief state. The Q-value functions are input to an algorithm that flags actions that are repeatedly unsuccessful as being actions that should not occur. Based on the assumption that such repeatedly unsuccessful actions should not (and cannot) be executed, individual CR-Prolog rules are created to prevent the execution of specific actions under specific conditions. These rules may (a) conflict with existing rules; or (b) include specific instances of more general rules. Conflicts with existing rules can be identified as inconsistencies in the answer set of the corresponding CR-Prolog program. Specific rule instances may be combined by rule regression to obtain more general rules, e.g., rules about not placing smaller objects of specific colors on larger objects may be combined to only consider the object sizes. The program with the new (or revised) rules is used in subsequent inference for planning or diagnosis—see (Sridharan and Rainge 2014) 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 (Colaco and Sridharan 2015; Sridharan et al. 2015; Sridharan and Rainge 2014; Zhang et al. 2014). 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 (Zhang, Sridharan, and Washington 2013).

First, consider an execution scenario in which the robot is in the $office$, and it is assigned the goal of moving a spe-

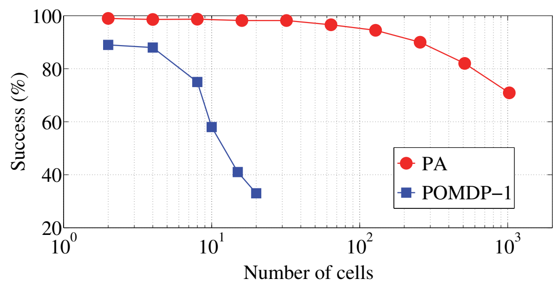

Figure 3: With a limit on the policy computation time, PA significantly increases accuracy in comparison with just POMDPs as the number of cells increases (Zhang et al. 2014).

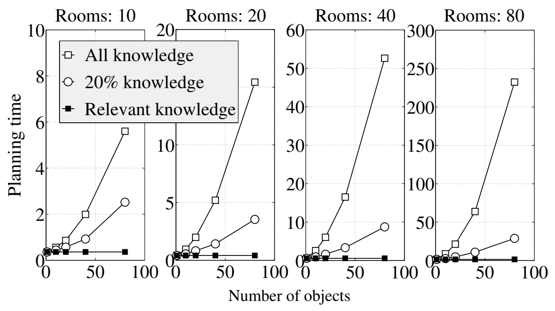

Figure 4: Planning time with PA scales well to larger number of rooms and objects (Zhang et al. 2014).

cific 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 Washington 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 is set to have four cells. As the number of cells (i.e., 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 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)—these results are summarized in Figure 4. 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 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 arrange objects in specific configurations. Some rules were intentionally 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 rules to the coarse-resolution system description. For instance, in the example where the robot had to stack books, it discovered the rule about not stacking bigger books on smaller ones, which was encoded as:

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

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

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 rules as an RL problem. 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 ; (b) relational representation for rule learning and generalization of discovered rules; and (c) extensive experimental on robots collaborating with humans in different application domains.

Acknowledgments

The architecture summarized in this paper is the result of collaboration with Michael Gelfond, Jeremy Wyatt, Shiqi Zhang, Zenon Colaco, and Sarah Rainge. This work was supported in part by the ONR Science of Autonomy award N00014-13-1-0766.

References

Bai, H.; Hsu, D.; and Lee, W. S. 2014. Integrated Perception and Planning in the Continuous Space: A POMDP Approach. *International Journal of Robotics Research* 33(8).

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

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)*.

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

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)*.

Erdem, E.; Aker, E.; and Patoglu, V. 2012. Answer Set Programming for Collaborative Housekeeping Robotics: Representation, Reasoning, and Execution. *Intelligent Service Robotics* 5(4):275–291.

Galindo, C.; Fernandez-Madrigal, J.-A.; Gonzalez, J.; and Saffioti, A. 2008. Robot Task Planning using Semantic Maps. *Robotics and Autonomous Systems* 56(11):955–966.

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

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.

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

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.

Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62(1-2):107–136.

Saribatur, Z.; Erdem, E.; and Patoglu, V. 2014. 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)*.

Sridharan, M., and Rainge, S. 2014. Integrating Reinforcement Learning and Declarative Programming to Learn Causal Laws in Dynamic Domains. In *International Conference on Social Robotics (ICSR)*.

Sridharan, M.; Gelfond, M.; Zhang, S.; and Wyatt, J. 2015. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, Unrefereed CoRR abstract: http://arxiv.org/abs/1508.03891.

Zhang, S., and Stone, P. 2015. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *AAAI Conference on Artificial Intelligence*, 1394–1400.

Zhang, S.; Sridharan, M.; Gelfond, M.; and Wyatt, J. 2014. Towards An Architecture for Knowledge Representation and Reasoning in Robotics. In *International Conference on Social Robotics (ICSR)*, 400–410.

Zhang, S.; Sridharan, M.; and Washington, C. 2013. Active Visual Planning for Mobile Robot Teams using Hierarchical POMDPs. *IEEE Transactions on Robotics* 29(4):975–985.

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.