In the *International Conference on Development and Learning and Epigenetic Robotics* (ICDL-EpiRob), Paris, France, September 19-22, 2016.

Should I do that? Using Relational Reinforcement Learning and Declarative Programming to Discover Domain Axioms

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

Ben Meadows
Department of Computer Science
The University of Auckland, NZ
bmea011@aucklanduni.ac.nz

Abstract— **Robots assisting humans in complex domains need the ability to represent, reason with, and learn from, different descriptions of incomplete domain knowledge and uncertainty. This paper focuses on the challenge of incrementally and interactively discovering previously unknown axioms governing domain dynamics, and describes an architecture that integrates declarative programming and relational reinforcement learning to address this challenge. Answer Set Prolog (ASP), a declarative programming paradigm, is used to represent and reason with incomplete domain knowledge for planning and diagnostics. For any given goal, unexplained failure of plans created by ASP-based inference is taken to indicate the existence of unknown domain axioms. The task of discovering these axioms is formulated as a reinforcement learning problem, and a relational representation is used to incrementally generalize from specific axioms identified over time. These generic axioms are then added to the ASP-based representation for subsequent inference. The architecture's capabilities are demonstrated and evaluated in two domains, *Blocks World* and *Robot Butler*.**

I. Introduction

Robots[1] assisting humans in complex domains such as health care and disaster rescue find it difficult to operate without considerable domain knowledge. At the same time, humans interacting with these robots may not have the expertise or time to provide elaborate and accurate domain knowledge. Robots are likely to receive some commonsense domain knowledge, including *default* knowledge that holds in all but a few exceptional situations, e.g., "books are typically in the library, but cookbooks may be in the kitchen". Robots also obtain information by processing sensor inputs—the uncertainty associated with this information is typically represented probabilistically, e.g., "I am 95% sure the cookbook is in the kitchen". Furthermore, some of the the axioms governing domain dynamics may not be known or may change over time. For instance, if the floor of the room the robot is moving in has just been polished, and the robot does not have an accurate model of moving on this surface, executing plans that require the robot to move on this surface may not produce the desired outcomes. To truly assist humans in such domains, robots thus need the ability to represent, reason with, and learn from, different descriptions of incomplete domain knowledge and uncertainty.

This paper focuses on the incremental and interactive discovery of previously unknown domain axioms. We build on prior work that combined declarative programming with probabilistic graphical models to address planning and diagnostics challenges in robotics [1], [2], [3], and some initial work in combining declarative programming with relational reinforcement learning for discovering domain axioms [4]. Our approach has the following features:

- An action language is used to describe the known (incomplete) knowledge about the domain's dynamics, translating this description and initial state defaults to an Answer Set Prolog (ASP) program that is solved for planning and diagnostics.
- The uncertainty in perception is abstracted away (for simplicity), and unexplained plan failures for any given goal are considered to be due to previously unknown domain axioms. The discovery of these axioms is formulated as a reinforcement learning problem.
- ASP-based reasoning, a relational representation, and a sampling-based approach are used for efficiently identifying candidate axioms and generalizing across candidate axioms. These generic axioms are included in the ASP program for subsequent reasoning.

We use two simulated domains, *Blocks World* and *Robot Butler*, to demonstrate the incremental and interactive discovery of axioms. Section II discusses some related work to motivate the proposed approach described in Section III. The experimental results are discussed in Section IV, followed by conclusions in Section V.

II. Related Work

This section motivates the proposed approach by reviewing related work in logic programming, probabilistic planning, reinforcement learning, and relational representations, in the context of robotics.

Probabilistic graphical models are used widely for planning sensing, navigation, and interaction, on robots [5], but these formulations, by themselves, make it difficult to reason with commonsense knowledge. Research in planning has provided many algorithms for knowledge representation and reasoning on robots, but these algorithms require considerable prior knowledge about the domain. Algorithms based on first-order logic do not support non-monotonic logical reasoning, default reasoning, and the ability to merge new, unreliable information with the current beliefs. Other logic-based formalisms address some of these limitations, e.g.,

[1]Terms "robot", "agent" and "learner" are used interchangeably.

Answer Set Prolog (ASP), a declarative language designed for representing and reasoning with commonsense knowledge [6], has been used by an international research community for cognitive robotics [7], [8]. However, ASP does not inherently support probabilistic models of uncertainty, or incremental and interactive acquisition of knowledge.

Many domains require robots to reason with logic-based and probabilistic representations—supporting this ability is a fundamental problem in robotics and artificial intelligence. Architectures have been developed for hierarchical representation of knowledge in first-order logic, and probabilistic processing of perceptual information [9], [10]. Existing approaches have combined deterministic and probabilistic algorithms for task and motion planning [11], [12], and used a three-layered organization of knowledge with first-order logic and probabilistic reasoning for open world planning [13]. Other approaches for combining logical and probabilistic reasoning include Markov logic networks [14], Bayesian Logic [15], relational partially observable Markov decision processes (POMDPs) [16], and probabilistic extensions to ASP [17], [18]. However, algorithms based on first-order logic do not provide the desired expressiveness for capabilities such as default reasoning, e.g., they model uncertainty by attaching probabilities to logic statements, which is not always meaningful. Other algorithms based on logic programming do not support one or more of the capabilities such as reasoning with large probabilistic components; reasoning about open worlds; and incremental and interactive learning of previously unknown domain knowledge.

In complex domains, agents (and humans) often have to start with an incomplete domain model and learn from repeated interactions with the environment. Researchers have used inductive logic with ASP to monotonically learn causal rules [19], and used a maximum satisfiability framework with plan traces for refining incomplete domain models [20]. Interactive learning can also be posed as an Reinforcement Learning (RL) problem with an underlying Markov decision process (MDP) [21]. Approaches for efficient RL in dynamic domains include sample-based planning algorithms [22], and Relational Reinforcement Learning (RRL), which combines relational representations of states and actions with regression for Q-function generalization and reuse of experience [23], [24]. However, existing algorithms focus on planning, limit generalization (e.g., using function approximation [25], [26] or explanation-based RL [27] for RRL) to a single MDP for a given planning task, or do not support the desired commonsense reasoning capabilities for robots.

This paper builds on prior work in (a) combining declarative programming and probabilistic graphical models for planning and diagnostics in robotics [1], [2], [3]; (b) combining declarative programming with RL for heuristic discovery of axioms in simplistic domains [28]; and (c) introducing relational representations for more efficient discovery of axioms [4], [29]. Here, we use ASP-based reasoning, a modified relational representation, and a sampling-based algorithm for efficiently identifying candidate axioms and generalizing across axiom instances.

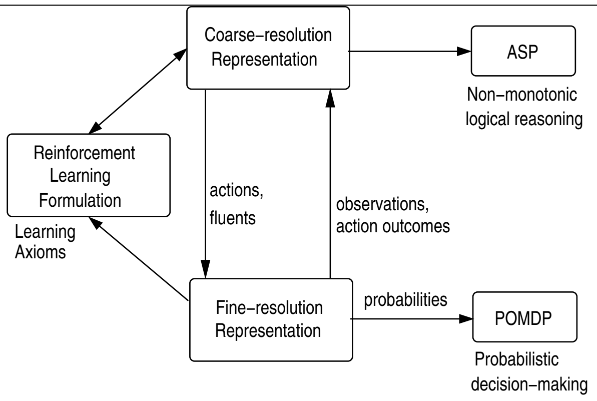

Coarse−resolution Representation
ASP
Non−monotonic logical reasoning
Reinforcement Learning Formulation
Learning Axioms
actions, fluents
observations, action outcomes
Fine−resolution Representation
probabilities
POMDP
Probabilistic decision−making

Fig. 1. Architecture combines complementary strengths of declarative programming, probabilistic graphical models, and reinforcement learning.

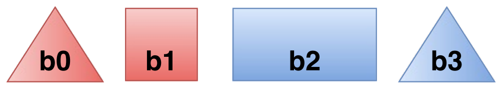


Fig. 2. A scenario in the Blocks World domain with four blocks.

III. Proposed Architecture

This section describes the incremental and interactive discovery of domain axioms. The overall architecture is shown in Figure 1. For any given goal, ASP-based non-monotonic reasoning with a coarse-resolution domain description provides a sequence of abstract actions. Each abstract action is implemented as a sequence of concrete actions, using a POMDP to probabilistically model the relevant part of the fine-resolution description obtained by refining the coarse-resolution description. In this paper, we abstract away the uncertainty in perception for simplicity, and do not discuss probabilistic planning. Instead, we use ASP for reasoning with commonsense knowledge at a single resolution (for planning and diagnostics), and focus on RRL-based interactive discovery of domain axioms. We illustrate the capabilities of this architecture using two simulated domains.

- **Blocks World (BW):** a tabletop domain where the objective is to stack blocks of different colors, shapes, and sizes, in specific configurations. Figure 2 illustrates a scenario with four blocks, which corresponds to ≈ 70 states under a standard RL/MDP formulation [23]. The robot may not know, for instance, that a block should not be placed on a prism-shaped block, and thus the corresponding action should not be attempted.
- **Robot Butler (RB):** a simulation of a "robot butler" that has to navigate between two rooms of various types in an office building to serve a beverage/drink to two people with various roles. Figure 3 is a partial illustration of a scenario in this domain. This domain has only 40 permutations of its physical object properties and four parametrized actions in a standard MDP formulation, but has ≈ 1150 static configurations that may be considered by the domain axioms. The robot butler may not know, for instance, that drinks should not be served to a person operating machinery.

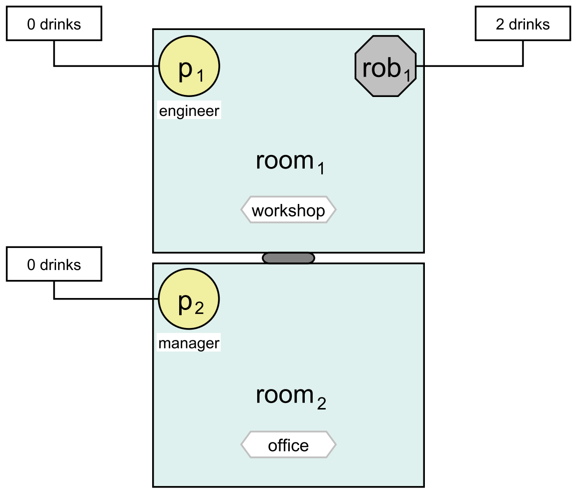


Fig. 3. A scenario in the Robot Butler domain. Physical configuration of objects, and static attributes *room type* and *person role* are depicted—p_1 and p_2 represent people, and rob_1 is the robot.

A. Knowledge Representation

The transition diagrams of our illustrative domains are described in an *action language* AL_d [6]. Action languages are formal models of parts of natural language used for describing transition diagrams. AL_d has a sorted signature containing three *sorts*: *statics*, *fluents* and *actions*. Statics are domain properties whose truth values cannot be changed by actions, whereas fluents are domain properties whose truth values can be changed by actions. Fluents are of two types: *inertial fluents* obey the laws of inertia and are changed directly by actions, whereas *defined fluents* do not obey the laws of inertia and cannot be changed directly by actions—they are changed based on other fluents. Actions are defined as a set of elementary actions that can be executed in parallel. A domain property p or its negation $\neg p$ is a domain literal. The action language allows three types of statements:

a **causes** l_{in} **if** $p_0, \ldots, p_m$ (Causal law)

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

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

where a is an action, l is a literal, l_{in} is an inertial fluent literal, and $p_0, \ldots, p_m$ are domain literals. A collection of statements of AL_d forms a system description.

The domain representation consists of a system description $\mathscr{D}$ and history $\mathscr{H}$. $\mathscr{D}$ has a sorted signature Σ and axioms used to describe the transition diagram τ. The signature Σ is a tuple that defines the names of objects, functions, and predicates available for use in the domain. The sorts of the BW domain include elements such as *block*, *place*, *color*, *shape*, *size*, and *robot*, whereas the sorts of the RB domain include elements such as *location*, *robot*, *people*, *roomtype* and *role*—when some sorts are subsorts of other sorts, e.g., *robot* and *people* may be subsorts of *thing*, they can be arranged hierarchically. Furthermore, the signature includes specific instances of sorts, e.g., rob_1 of sort *robot*, $\{office, conference, kitchen, workshop\}$ of sort *roomtype*, and $\{engineer, salesperson, manager\}$ of sort *role*.

We describe the fluents and actions of the domain in terms of the sorts of their arguments. The BW domain's fluent $on(block, place)$ describes the place location of each block—this is an inertial fluent that obeys the laws of inertia. There are some statics for block attributes $has\_color(block, color)$, $has\_shape(block, shape)$ and $has\_size(block, size)$. The action $move(block, place)$ moves a block to a specific place (*table* or another block). In the RB domain, the fluents are the location of the *robot* and the *people*—we reason about the former ($at(robot, location)$) and assume the latter are defined fluents ($at(person, location)$) known at all times. The robot can move between locations, represented as an action $move(robot, location)$, and serve a person at a location with $serve(robot, person, location)$. We also introduce relations for people's roles, and for specific types of rooms. Additionally, the scenario has attributes describing whether it is early or late in the day.

For the BW domain, the dynamics are defined in terms of causal laws such as:

$$move(B, L) \ \textbf{causes} \ on(B, L)$$

state constraints such as:

$$\neg on(B, L_2) \ \textbf{if} \ on(B, L_1), \ L_1 \neq L_2$$

and executability conditions such as:

$$\textbf{impossible} \ move(B_2, L) \ \textbf{if} \ on(B_1, L), \ B_1 \neq B_2$$

In a similar manner, the RB domain's dynamics are defined using causal laws such as:

$$serve(rob_1, P, L) \ \textbf{causes} \ has(P, drink)$$

state constraints such as:

$$\neg at(Th, L_2) \ \textbf{if} \ at(Th, L_1), \ L_1 \neq L_2$$

and executability constraints such as:

$$\textbf{impossible} \ serve(rob_1, P, L) \ \textbf{if} \ has(P, drink)$$

The recorded history of a dynamic domain is usually a record of fluents observed to be true at a time step, and the occurrence of an action at a time step. Our prior work introduced a new type of record to encode (prioritized) defaults describing the values of fluents in their initial states [2], [30]. For instance, we can encode a default statement such as "blocks are usually on the table or on another block", and encode exceptions to such default statements.

The domain representation is translated into a CR-Prolog[2] program $\Pi(\mathscr{D}, \mathscr{H})$, i.e., a collection of statements describing domain objects and relations between them. CR-Prolog is a variant of Answer Set Prolog (ASP) that supports representation and reasoning with defaults and their direct and indirect exceptions, and incorporates consistency restoring (CR) rules in ASP [31]. ASP is based on stable model semantics and

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

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 *a is not believed to be true*, and unlike "$p \lor \neg p$" in propositional logic, "p *or* $\neg p$" is not a tautology [6]. 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. Algorithms for computing the entailment, and for planning and diagnostics, reduce these tasks to computing answer sets of CR-Prolog programs. Π consists of causal laws of $\mathscr{D}$, 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 [30] for details. Although not discussed here, the program representing the current beliefs of the robot also supports capabilities such as explaining unexpected action outcomes and partial descriptions extracted from sensor inputs—see [1].

In complex domains, it is difficult for human participants to equip robots with complete and accurate domain knowledge. This is especially true of domain axioms—some of these axioms may not be known or may change over time. The plans created using this incomplete knowledge may result in unintended consequences. Consider a scenario in the BW domain in which the goal is to stack three of four blocks placed on the table. Figure 4(a) shows a possible goal configuration that could be generated based on the available domain knowledge. The corresponding plan, with all four block being on the table in the initial state, has two steps: $move(b_1,b_0)$ followed by $move(b_2,b_1)$. The robot expects this plan to succeed in achieving the desired goal state. However, unknown to the robot, no block can be stacked on top of a prism-shaped block in this domain. As a result, execution of this plan results in failure that cannot be explained—specifically, action $move(b_1,b_0)$ does not result in the expected configuration shown in Figure 4(b). In this paper, we focus on discovering previously unknown executability conditions, which can prevent such actions from being included in a plan for any given goal.

B. Relational RL for Discovering Axioms

Our approach for incremental and interactive discovery of previously unknown domain axioms differs from previous work by us and other researchers. The proposed approach:

- Explores the existence of previously unknown axioms only when unexpected action outcomes cannot be explained by reasoning about exogenous actions.
- Uses ASP-based reasoning, RRL, and inductive and incremental decision tree regression, for efficiently identifying candidate axioms and generalizing across specific candidate axioms.
- Uses an efficient sampling-based approach to construct and consider multiple MDPs for discovering domain axioms, instead of using RRL for planning, which limits generalization to a specific MDP.

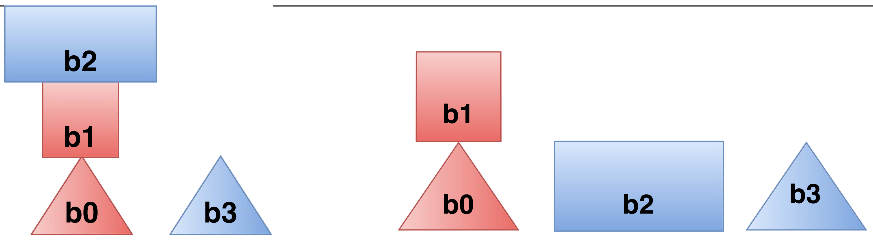
Generalization and computational efficiency are core considerations for incremental and interactive learning. For instance, in the RB domain, discovery of the axiom "person p_1 operating machinery should not be served drinks" does not help with person p_2 operating machinery in the office, unless the robot realizes, over time, that this axiom is a specific instance of the generic axiom "drinks should not be served to any person operating machinery".

A sequence of steps is used to identify candidate axioms and to generalize across these candidates to obtain the generic axioms. First, when a specific plan step fails, the corresponding state is considered the goal state in an RL problem, with the objective of finding state-action pairs that are most likely to lead to this error state. The RL problem uses an underlying MDP formulation defined by the tuple $\langle S,A,T,R\rangle$, where:

- S: set of states.
- A: set of actions.
- $T: S \times A \times S' \rightarrow [0,1]$ is the state transition function.
- $R: S \times A \times S' \rightarrow \Re$ is the reward function

The transition function and reward function (T and R) are not known in an RL problem. Each state, i.e., each element of S, is the assignment of specific (ground) values to the domain fluents and statics, and a boolean fluent describing whether the most recent action resulted in failure. For instance, if we were to consider two blocks with known color and shape in the BW domain, each state would consider a possible physical configuration of the blocks. Similarly, each element of A is a valid action for the domain, e.g., serving drinks to person p_1 who is a *manager* located in $room_2$ in the RB domain. Next, the known axioms in the ASP-based domain description are used to eliminate invalid combinations of states and actions in the domain. For instance, no two people can be in the same location in the RB domain. Also, it is not possible to move a block that is under another block in the BW domain. Inconsistent state transitions are identified by constructing and computing answer sets of ASP programs with the specific state-action combinations and the axioms in $\mathscr{D}$. This "filtering" significantly reduces the size of the problem because only valid state-action combinations are included as elements of T and R. Furthermore, the MDP is constructed automatically from the ASP system description, for any given goal state.

The next step is to estimate the Q-values of the state-action pairs $Q(s,a)$ for particular scenarios (pairs of goal and initial states). Basic RL algorithms for estimating Q-values, such as Q-learning and SARSA, do not scale well with large increases in the state space size, becoming computationally intractable. They also do not generalize to relationally equivalent states. To address these issues, we use a relational representation: after an episode of Q-learning (one iteration from start to end of the scenario), all state-action pairs that have been visited, along with their estimated Q-values, are used to update a binary (i.e., logical) decision tree (BDT). The path from the root node to a leaf node corresponds



(a) Planned goal state (b) Failure in plan execution

Fig. 4. Illustrative example of (a) a planned goal state; and (b) failure during a specific step in plan execution.

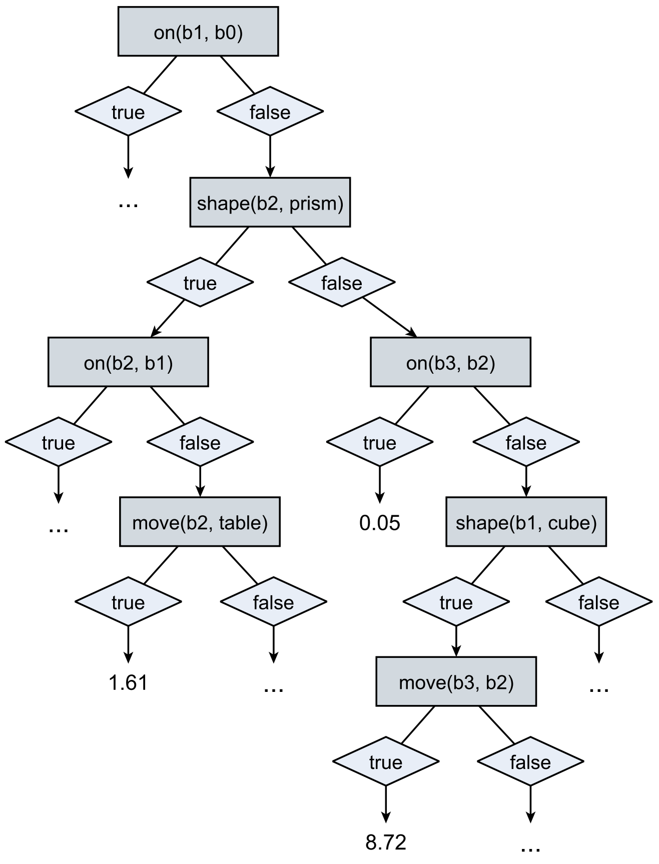


Fig. 5. Subset of the BDT for a specific scenario in the BW domain. Numbers at the leaves reflect Q-values.

to one pair (s,a), where a is an action and s is a partial state description. Individual nodes in the BDT correspond to true/false tests (e.g., specific fluent terms, statics or actions) with which the node's two sub-branches are associated. Figure 5 illustrates a subset of a BDT constructed for the BW domain. This tree is used to compute the policy for the next episode, after which the BDT will be revised further.

Instead of generating this tree from scratch after each episode, we use RRL-TG, an incremental inductive learning algorithm operating over a dynamic decision tree. This algorithm significantly (a) reduces overhead and storage requirements; and (b) improves performance gain by eliminating the need for building the tree anew after each episode [32]. The BDT is altered after every iteration, and used to provide the policy for the next iteration. When RL is terminated, the BDT relationally represents the robot's experiences.

The method described above only considers generalization within a specific MDP. To identify generic domain axioms, the third step of our approach simulates similar errors (to the one actually encountered due to plan step execution failure) and considers the corresponding MDPs as well. Specifically, it varies the underlying *attribute configurations*, i.e., the set of ground statics representing object properties that are fixed within a specific episode in a specific scenario. These configurations are most relevant to the learning task because the robot is assumed to have accurate knowledge about physical configurations, but is missing one or more executability constraints governed by static object properties, e.g., it is a block's prism shape that makes it impossible to stack another block on it. In other words, we construct MDPs for a sample of 1% of the number of valid attribute configurations of the domain. For each such MDP, we perform RRL as described above, but change the static configuration a number of times for each MDP. Each learning instance will thus start with a randomized non-repeated attribute configuration; run learning episodes until convergence; revise the attribute configuration one literal at a time until a non-repeated attribute configuration is found; run learning episodes; and repeat this process until a certain number of attribute configurations have been examined for each MDP. In this manner, the robot explores permutations that implicitly vary the goal state. Also, by making only minor changes to the attribute configuration within each MDP, we maximize the applicability of the current learned policy to the new attribute configuration.

The fourth step identifies candidate domain axioms, or more specifically, executability conditions. The head of an axiom contains a ground action, and the body contains one or more attributes that influence (or are influenced by) the action. Generalization is performed by taking each leaf from each BDT, extracting a partial state-action description using its path to the root, and collecting the fluents from that description. The goal state of the current scenario helps identify candidate actions that had unexpected outcomes. The resulting examples may be modeled as a tree whose root node corresponds to the non-occurrence of the action, intermediate nodes correspond to attributes of objects involved in the action (i.e., relevant statics and fluents), and leaf nodes

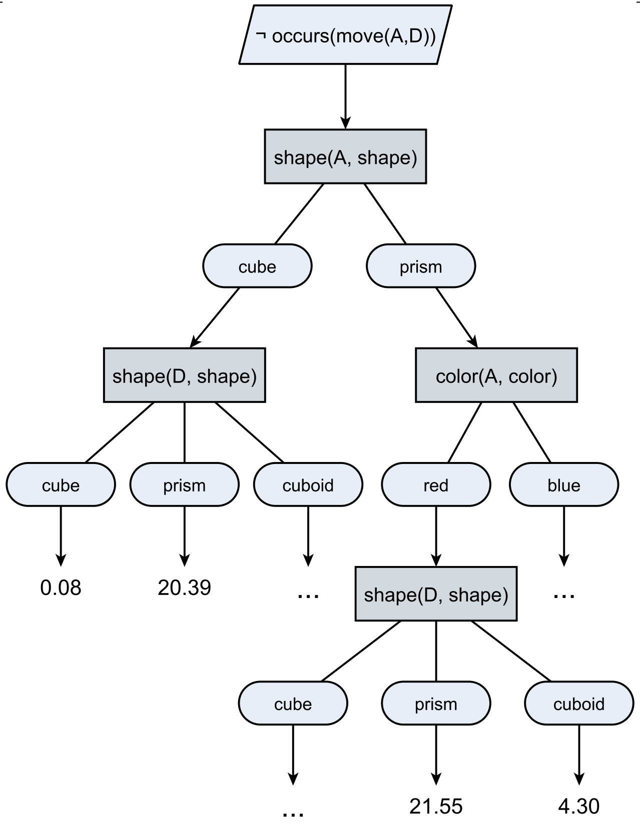


Fig. 6. Decision tree representing candidate axioms related to a specific action in the blocks world domain. Numbers attached to the leaves reflect amassed mean values.

average the values of the samples grouped under that node. Invalid branches, e.g., corresponding to an action without attributes, are removed. After the set of partial examples of candidate axioms is created, examples with identical contents are identified, and ground values in the examples are replaced with variables to create a *generalized* example. Figure 6 depicts a subset of a decision tree visualizing candidate axioms related to a particular action (i.e., $move(A,D)$) in the Blocks World domain.

For some complex domains (e.g., the Robot Butler domain), we may still end up with multiple different axioms. If so, the final step uses K-means algorithm to cluster axioms based on their value. The axioms in the cluster with the largest mean value are added to the ASP program $\Pi(\mathscr{D},\mathscr{H})$ to be used for subsequent reasoning.

IV. Experimental Setup and Results

The proposed approach, henceforth referred to as "Q-RRL", was grounded and evaluated in the Blocks World domain and Robot Butler domains—see Section III for a description of these domains. We describe the performance in illustrative execution scenarios drawn from these domains. We also compare the rate of convergence and the memory requirements of Q-RRL with those of traditional Q-learning.

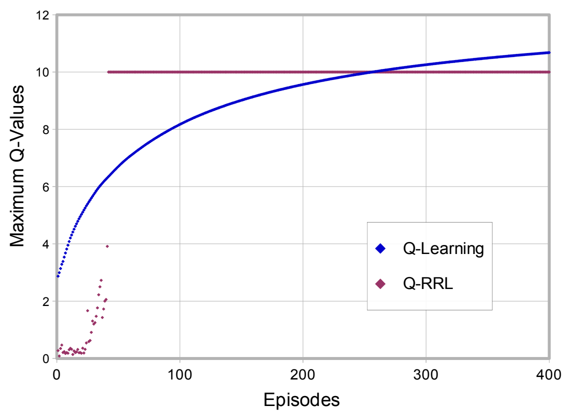


Fig. 7. Comparing the rate of convergence of Q-RRL with Q-learning in a specific scenario in the BW domain—Q-RRL converges much faster.

A. Blocks World

The objective in the BW domain was to stack the blocks in a particular configuration. Consider trials in which the robot did not know it was impossible to place any block on a prism-shaped block. Among possible scenarios, consider the scenario with four blocks: $b0$ (Red Prism); $b1$ (Red Cube); $b2$ (Blue Cuboid); and $b3$ (Blue Prism). The initial state was:

$$on(b0,table),\ on(b1,table),\ on(b2,table),\ on(b3,table)$$

and the goal state description was:

$$on(b0,table),\quad on(b1,b0),\quad on(b2,b1),\quad on(b3,table)$$

The plan had actions $move(b1,b0)$ and $move(b2,b1)$—action $move(b1,b0)$ fails. During Q-RRL and Q-learning, the agent received a reward of $+100$ for reaching the goal state and a reward of -1.5 for other actions.

As stated earlier, RRL is triggered when executing the computed plan (to stack blocks) results in an unexpected outcome than cannot be explained. During RRL, different related scenarios are simulated (e.g., combinations of attributes of blocks) to generate the training samples for generalization. Figure 7 shows the rate of convergence of the maximum Q-value obtained using Q-RRL and Q-learning. The Q-RRL algorithm has a much better rate of convergence, i.e., the optimal policy is computed in significantly fewer number of episodes. In these trials, we are concerned about the rate of convergence—the actual Q-values do not matter. The following are some axioms identified during the iterations:

$$\begin{aligned}\neg occurs(move(A,D),I) \ \leftarrow\ & has\_shape(D,prism),\\ & has\_shape(A,cuboid),\\ & has\_color(D,blue)\\ \neg occurs(move(A,D),I) \ \leftarrow\ & has\_shape(D,prism),\\ & has\_shape(A,cube),\\ & has\_color(D,red)\end{aligned}$$

$$\begin{aligned}\neg occurs(move(A,D),I) \;\leftarrow\; & has\_shape(D,prism),\\ & has\_shape(A,prism),\\ & has\_color(A,red),\\ & has\_color(D,blue)\end{aligned}$$

As the robot explores different scenarios, there are fewer errors because impossible actions are no longer included in the plans. Furthermore, the robot is able to incrementally generalize from the different specific axioms to add the following generic axiom to the CR-Prolog program:

$$\neg occurs(move(A,D),I) \;\leftarrow\; has\_shape(D,prism)$$

The proposed approach resulted in a similar discovery of unknown axioms in other experimental trials conducted in the BW domain.

B. Robot Butler

In the RB domain, the robot has to travel to the location of a person who has no beverage and serve her/him a drink. As described in Section III, *move* actions move the robot between rooms, while *serve* actions result in a person in the same room as the robot being served a beverage. Successfully serving the two people in any given scenario, or failing to do so over a long period of time, terminates the episode. Individual scenarios in the RB domain can differ in terms of the locations and attributes of the objects. For instance, each person has one of three different *roles* and each location corresponds to one of four different *roomtypes*. Any unexpected termination (e.g., the robot fails to deliver a beverage to specific person) that cannot be explained using existing knowledge triggers RRL for discovering axioms. During the process, scenarios similar to the one causing the failure are simulated, e.g., with different roles for people or types of room—see Section III. These simulated scenarios provide the training examples for generalizing from the specific axioms discovered. Figure 8 compares the rate of convergence of Q-RRL and Q-learning as a function of the number of episodes. Q-RRL converges faster than Q-learning, although the difference is less significant than in the BW domain, possibly because the RB domain had multiple axioms missing concurrently.

In the experimental trials in the RB domain, we specifically omitted some executability conditions, e.g., managers only accept beverages early in the day; drinks cannot be served in the workshop; or drinks can *only* be served in the kitchen. We noticed that our approach was less accurate in the RB domain than it was in the BW domain, possibly because there were multiple concurrently missing axioms. The sole highest-valued candidate axiom was not always the correct executability condition, but many of the higher-valued candidates were correct (or almost correct). The errors mostly corresponded to the axiom being too generic or including additional (unnecessary) attributes. We list below

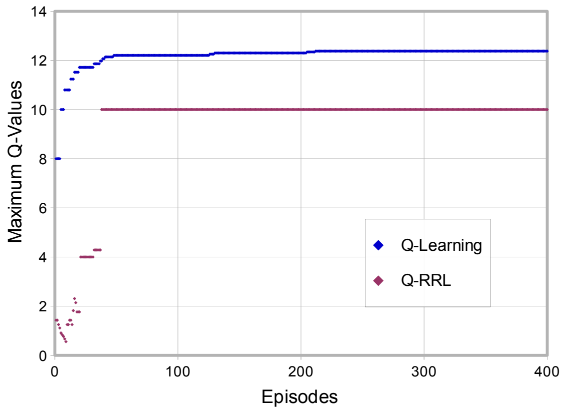


Fig. 8. Comparing the rate of convergence of Q-RRL with Q-learning in a specific scenario in the RB domain—Q-RRL converges faster.

some examples of axioms generated.

$$\begin{aligned}&(1) & \neg occurs(serve(R,P,L)) \;\leftarrow\; & roomtype(L,workshop)\\ &(2) & \neg occurs(serve(R,P,L)) \;\leftarrow\; & earlyinday(false),\\ &&& \neg role(P,engineer)\\ &&& \neg role(P,salesperson)\\ &(3) & \neg occurs(serve(R,P,L)) \;\leftarrow\; & roomtype(L,office),\\ &&& \neg roomtype(L,workshop)\end{aligned}$$

The first axiom states that a beverage cannot be served in the workshop. The second axiom is also correct, although specifying that person P is not an engineer or salesperson is a convoluted way of saying P is a manager. The third axiom makes sense when the unknown axiom is that people can only be served in the kitchen. We also observe that when the actual generic axiom is not found, the values assigned to the top candidate is much lower than when the generic axiom is found.

Memory costs: For Q-learning, the data for an training example (i.e., record), including its Q-value, is stored in a record, whereas Q-RRL stores this data in a leaf. One of the motivations for the RRL-TG algorithm is that it is expensive to keep track of a monotonically increasing number of examples, and to build trees anew from examples after each episode [32]. Memory costs can be estimated by counting the number of records and leaves generated over all MDPs produced in the course of learning. We find that Q-RRL with incremental tree learning reduces the number of records drastically: by between 89% (for RB) and 98% (for BW). Even if we count all the nodes in the BDTs towards the complexity of the framework, using Q-RRL results in a reduction in complexity of at least 75%, which is also supported by an (experimentally) observed reduction in processing time when we use Q-RRL.

V. Conclusion

Robots assisting humans in complex domains frequently need to represent, reason with, and learn from, different descriptions of incomplete domain knowledge and uncertainty. The architecture described in this paper combines the complementary strengths of declarative programming and relational reinforcement learning to discover previously unknown axioms governing domain dynamics. We illustrated the architecture's capabilities in two simulated domains (Blocks World and Robot Butler), with promising results. Future work will explore the ability to discover other kinds of axioms (e.g., state constraints and causal laws), and explore other application domains. We will also conduct experimental trials on a mobile robot after introducing probabilistic models of the uncertainty in perception, which will transform the decision-making problem from an MDP to a POMDP—this will also enable us to fully utilize the probabilistic planning component of the overall architecture depicted in Figure 1. Furthermore, it may be possible to use the proposed approach for incremental and interactive (relational) learning, and the overall architecture, to analyze and explain the behavior of robots and humans in dynamic domains.

Acknowledgments

The authors thank Prashanth Devarakonda and Rashmica Gupta for their help in implementing an initial version of the approach described in this paper, as documented in [4]. 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 expressed in this paper are those of the authors.

References

[1] Z. Colaco and M. 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.

[2] S. Zhang, M. Sridharan, M. Gelfond, and J. Wyatt, "Towards An Architecture for Knowledge Representation and Reasoning in Robotics," in *International Conference on Social Robotics (ICSR)*, Sydney, Australia, October 27-29, 2014, pp. 400–410.

[3] S. Zhang, M. Sridharan, and J. Wyatt, "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 699–713, 2015.

[4] M. Sridharan, P. Devarakonda, and R. Gupta, "Can I Do That? Discovering Domain Axioms Using Declarative Programming and Relational Reinforcement Learning," in *AAMAS Workshop on Autonomous Robots and Multiagent Systems (ARMS)*, Singapore, May 9, 2016.

[5] H. Bai, D. Hsu, and W. S. Lee, "Integrated Perception and Planning in the Continuous Space: A POMDP Approach," *International Journal of Robotics Research*, vol. 33, no. 8, 2014.

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

[7] M. Balduccini, W. C. Regli, and D. 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.

[8] E. Erdem and V. Patoglu, "Applications of Action Languages to Cognitive Robotics," in *Correct Reasoning*. Springer-Verlag, 2012.

[9] J. E. Laird, "Extending the Soar Cognitive Architecture," in *International Conference on Artificial General Intelligence*, Memphis, USA, March 1-3, 2008.

[10] K. Talamadupula, J. Benton, S. Kambhampati, P. Schermerhorn, and M. Scheutz, "Planning for Human-Robot Teaming in Open Worlds," *ACM Transactions on Intelligent Systems and Technology*, vol. 1, no. 2, pp. 14:1–14:24, 2010.

[11] L. Kaelbling and T. Lozano-Perez, "Integrated Task and Motion Planning in Belief Space," *International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[12] Z. Saribatur, E. Erdem, and V. Patoglu, "Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans," in *International Conference on Intelligent Robots and Systems*, Chicago, USA, 2014, pp. 2923–2930.

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

[14] M. Richardson and P. Domingos, "Markov Logic Networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, February 2006.

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

[16] S. Sanner and K. Kersting, "Symbolic Dynamic Programming for First-order POMDPs," in *AAAI Conference on Artificial Intelligence*, Atlanta, USA, July 11-15, 2010, pp. 1140–1146.

[17] C. Baral, M. Gelfond, and N. Rushton, "Probabilistic Reasoning with Answer Sets," *Theory and Practice of Logic Programming*, vol. 9, no. 1, pp. 57–144, January 2009.

[18] J. Lee and Y. Wang, "A Probabilistic Extension of the Stable Model Semantics," in *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, March 2015.

[19] R. P. Otero, "Induction of the Effects of Actions by Monotonic Methods," in *International Conference on Inductive Logic Programming*, 2003, pp. 299–310.

[20] H. H. Zhuo, T. Nguyen, and S. Kambhampati, "Refining Incomplete Planning Domain Models Through Plan Traces," in *International Joint Conference on Artificial Intelligence (IJCAI)*, Beijing, China, August 2013, pp. 2451–2457.

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

[22] T. J. Walsh, S. Goschin, and M. L. Littman, "Integrating Sample-Based Planning and Model-Based Reinforcement Learning," in *AAAI Conference on Artificial Intelligence*, Atlanta, USA, July 11-15, 2010.

[23] S. Dzeroski, L. D. Raedt, and K. Driessens, "Relational Reinforcement Learning," *Machine Learning*, vol. 43, pp. 7–52, 2001.

[24] P. Tadepalli, R. Givan, and K. Driessens, "Relational Reinforcement Learning: An Overview," in *Relational Reinforcement Learning Workshop at the International Conference on Machine Learning*, 2004.

[25] K. Driessens and J. Ramon, "Relational Instance-Based Regression for Relational Reinforcement Learning," in *International Conference on Machine Learning (ICML)*. AAAI Press, 2003, pp. 123–130.

[26] T. Gartner, K. Driessens, and J. Ramon, "Graph Kernels and Gaussian Processes for Relational Reinforcement Learning," in *International Conference on Inductive Logic Programming (ILP)*. Springer, 2003, pp. 140–163.

[27] C. Boutilier, R. Reiter, and B. Price, "Symbolic Dynamic Programming for First-Order MDPs," in *International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, USA, August 4-10, 2001, pp. 690–700.

[28] M. Sridharan and S. Rainge, "Integrating Reinforcement Learning and Declarative Programming to Learn Causal Laws in Dynamic Domains," in *International Conference on Social Robotics (ICSR)*, Sydney, Australia, October 27-29, 2014.

[29] M. Sridharan, P. Devarakonda, and R. 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.

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

[31] M. Balduccini and M. Gelfond, "Logic Programs with Consistency-Restoring Rules," in *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, 2003, pp. 9–18.

[32] Kurt Driessens and Jan Ramon and Hendrik Blockeel, "Speeding up Relational Reinforcement Learning Through the Use of an Incremental First Order Decision Tree Learner," in *European Conference on Machine Learning (ECML)*. Springer-Verlag, 2001, pp. 97–108.