Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics

Tiago Mota[1][0000−0002−2339−0270], Mohan Sridharan[2][0000−0001−9922−8969], and Aleš Leonardis[2][0000−0003−0773−3277]

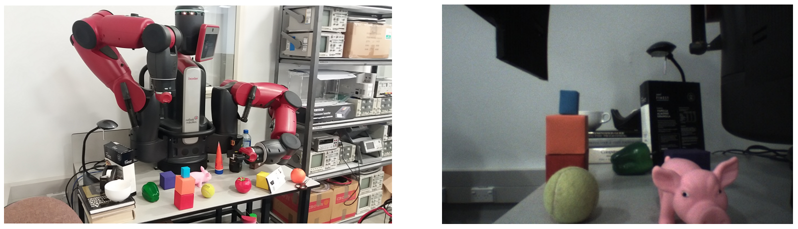
[1] Electrical and Computer Engineering, The University of Auckland, New Zealand
tmot987@aucklanduni.ac.nz
[2] School of Computer Science, University of Birmingham, United Kingdom
m.sridharan@bham.ac.uk, a.leonardis@bham.ac.uk

Abstract. A robot's ability to provide explanatory descriptions of its decisions and beliefs promotes effective collaboration with humans. Providing such transparency in decision making is particularly challenging in integrated robot systems that include knowledge-based reasoning methods and data-driven learning algorithms. Towards addressing this challenge, our architecture couples the complementary strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, deep learning, and inductive learning. During reasoning and learning, the architecture enables a robot to provide on-demand *explanations* of its decisions, beliefs, and the outcomes of hypothetical actions, in the form of relational descriptions of relevant domain objects, attributes, and actions. The architecture's capabilities are illustrated and evaluated in the context of scene understanding tasks and planning tasks performed using simulated images and images from a physical robot manipulating tabletop objects. Experimental results indicate the ability to reliably acquire and merge new information about the domain in the form of constraints, and to provide accurate explanations in the presence of noisy sensing and actuation.

Keywords: Explainable reasoning and learning · Non-monotonic logical reasoning · Deep learning· Scene understanding · Robotics.

1 Introduction

Imagine a robot arranging objects in desired configurations on a table, and estimating the occlusion of objects and stability of object configurations. Figure 1a illustrates a scene in this setting. An object is considered to be occluded if the view of any minimal fraction of its frontal face is hidden by another object, and any given configuration (i.e., a vertical stack of objects) is unstable if any object in the configuration is unstable. To perform these tasks, the robot extracts information from on-board camera images, reasons with this information and incomplete domain knowledge, and executes actions to achieve desired outcomes. The robot also incrementally learns previously unknown constraints, and responds to questions about its plans, actions, associated decisions, and beliefs. For instance, assume that the target configuration in Figure 1b is to have the pig on the orange block, and that the plan is to move the blue block on to the table before



(a) Test scenario. (b) Image from robot's camera.

Fig. 1: (a) Motivating scenario of a Baxter robot arranging objects in desired configurations on a tabletop; (b) Image from the camera on the robot's left gripper.

placing the pig on the orange block. When asked to justify a step of the plan, e.g., "why do you want to pick up the blue block first?", the robot answers "I have to put the pig on the orange block. The blue block is on the orange block"; when asked, after executing the plan, to explain why an action was not executed, e.g., "why didn't you pick up the pig first?", the robot responds "Because the blue block is on the orange block".

Realizing the motivating scenario described above poses challenges in knowledge representation, reasoning, learning, and control. This paper focuses on enabling a robot to provide an on-demand *explanation* of its decisions and beliefs in the form of a description comprising relations between relevant objects, actions, and attributes of the domain. Such "explainability" will help establish accountability in the robot's decision making and help the human designer improve the algorithms, but it remains an open problem. It is particularly challenging with integrated robot systems that include knowledge-based reasoning methods (e.g., for planning and diagnostics) and data-driven (e.g., deep learning) algorithms that are the state of the art for many pattern recognition problems. Research in cognitive systems and architectures indicates that relational representations and reasoning with commonsense knowledge help promote transparency in decision making [12,14,28]. Inspired by this insight, our architecture tightly couples the complementary strengths of knowledge-based and data-driven methods, while providing transparent decision making. It builds on and significantly expands our prior work that combined non-monotonic logical reasoning and deep learning for scene understanding in simulated images [20]. This paper contributes the ability to:

– Automatically extract relevant information and construct explanations as relational descriptions provided in response to questions about the robot's decisions and beliefs, including under hypothetical situations.
– Incrementally merge newly acquired information with existing knowledge, exploiting the interplay between representational choices, reasoning methods, and learning algorithms to generate accurate explanations.

These capabilities are evaluated in the context of planning tasks and scene understanding tasks in simulated scenes and on a physical robot manipulating tabletop objects. Specifically, the robot (i) computes and executes plans to arrange objects in desired configurations; and (ii) estimates occlusion of scene objects and stability of object configurations. Experimental results indicate the ability to (i) incrementally reduce uncertainty about the scene by learning previously unknown state constraints; and (ii) construct explanations reliably and efficiently by automatically identifying and reasoning with the relevant knowledge despite noisy sensing and actuation.

The remainder of the paper is organized as follows. We first discuss related work in Section 2, followed by a description of the architecture in Section 3. Experimental results are discussed in Section 4, and the conclusions are presented in Section 5.

2 RELATED WORK

Early work on explanation generation drew on research in cognition, psychology, and linguistics to characterize explanations in terms of generality, objectivity, connectivity, relevance, and information content [6]. Subsequent studies involving human subjects have also indicated that the important attributes of good explanations include coherence, simplicity, generality, soundness, and completeness [23]. In parallel, fundamental computational methods were developed for explaining unexpected outcomes by reasoning logically about potential causes [9].

With the use of AI and machine learning methods in different domains, there is much interest in understanding the decisions of these methods[3]. This understanding can be used to improve the underlying algorithms, and to make automated decision-making more acceptable or trustworthy to humans [16]. Recent work in *explainable AI* and *explainable planning* can be broadly categorized into two groups [18]. Methods in one group modify or map learned models or reasoning systems to make their decisions more interpretable, e.g., by tracing decisions back to input data [10] or explaining the predictions of any classifier by learning equivalent interpretable models [24], or biasing a planning system towards making decisions easier for humans to understand [32]. Methods in the other group provide descriptions that make a reasoning system's decisions more transparent, e.g., explaining planning decisions [3], or causal and temporal relations [26]. Much of this research is agnostic to how an explanation is structured or assumes comprehensive domain knowledge. Given the use of deep networks and related algorithms in different applications, methods are being developed to understand the operation of these networks, e.g., by computing the features most relevant to a deep network's outputs [2]. As documented in a recent survey, these methods compute gradients and decompositions in a network's layers to obtain heatmaps of the relevant features [25]. There has also been work on reasoning with learned symbolic structure, or with a learned graph encoding scene structure, in conjunction with deep networks to answer questions about images of scenes [22,31]. However, these approaches do not (i) fully integrate reasoning and learning to inform and guide each other; or (ii) use the rich commonsense knowledge, which is available in almost every domain, for reliable and efficient reasoning, learning, and the generation of descriptions of the decisions and beliefs of the system under consideration.

Our focus is on integrated robot systems that use a combination of knowledge-based and data-driven algorithms to represent, reason with, and learn from incomplete

[3] For a recent debate on whether interpretability is needed in machine learning, please see: `https://www.youtube.com/watch?v=93Xv8vJ2acI`

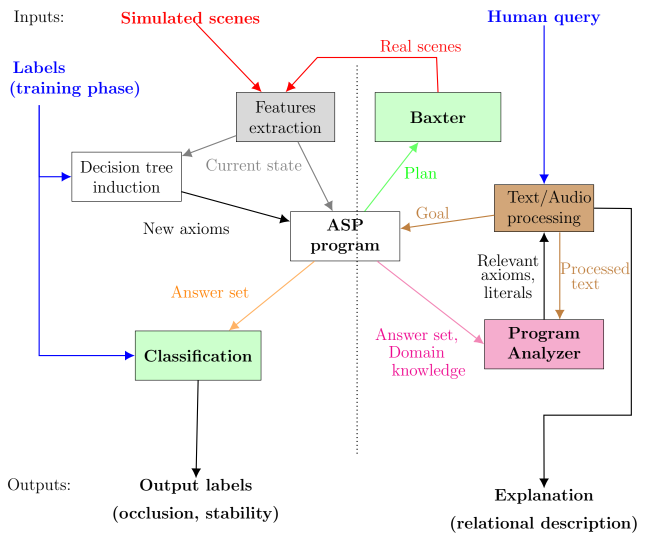


Fig. 2: Architecture combines strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, deep learning, and inductive learning. New components to the right of the dashed line support desired explainability.

domain knowledge and noisy observations. We enable such robots to generate relational descriptions of decisions, beliefs, and hypothetical or counterfactual situations; humans often consider such hypothetical options to infer causal relations [4]. Recent surveys state that these capabilities are not supported by existing systems [1,18]. Our architecture addresses this limitation by extending work in our group on explainable agency [13], a theory of explanations [30], and on combining non-monotonic logical reasoning and deep learning for classification of simulated images [20].

3 ARCHITECTURE

Figure 2 shows the overall architecture. Components to the left of the dashed vertical line were introduced in our prior work that combined non-monotonic logical reasoning and deep learning for classification in simulated images [20]; we summarize these components for completeness. Components to the right of the dashed line are introduced here to expand reasoning capabilities and answer questions about decisions, beliefs, and hypothetical situations. We describe these new components and revisions to existing components in more detail. We do so using the following example domain.

Example Domain 1 *[Robot Assistant (RA) Domain]*
A Baxter (see Figure 1a): (i) estimates occlusion of scene objects and stability of object structures, and arranges objects in desired configurations; and (ii) provides relational descriptions of decisions, beliefs, and hypothetical situations as responses to questions and commands. There is uncertainty in the robot's perception and actuation, and the robot uses probabilistic algorithms to visually recognize and move objects. The robot has

incomplete (and potentially imprecise) domain knowledge, which includes object attributes such as $size$ (small, medium, large), $surface$ (flat, irregular) and $shape$ (cube, apple, duck); spatial relations between objects (above, below, front, behind, right, left, in); some domain attributes; and some axioms governing domain dynamics such as:

– Placing an object on top of an object with an irregular surface results in an unstable object configuration.
– For any given object, removing all objects blocking the view of any minimal fraction of its frontal face causes this object to be not occluded.

This knowledge may need to be revised over time, e.g., some actions, axioms, and the values of some attributes may not be known, or the robot may find that placing certain objects on an object with an irregular surface results in a stable configuration.

3.1 Knowledge Representation, Reasoning, and Learning

We first describe the knowledge representation, reasoning, and learning components.

Feature extraction: In our architecture, the sensor inputs are RGB images of simulated scenes, or noisy top and front views of any given scene from the robot's cameras; our previous work considered RGB-D images (i.e., point clouds) of simple simulated scenes [20]. From each image, a probabilistic algorithm is used to extract objects and their attributes. Also, the spatial relations between objects are computed using our prior work that incrementally learns the *grounding*, i.e., the meaning in the physical world, for position-based and distance-based prepositional words such as "above", "in", and "far", in the form of 2D and 1D histograms [19].

Non-monotonic logical reasoning: To represent and reason with domain knowledge, we use CR-Prolog, an extension to Answer Set Prolog (ASP) that introduces *consistency restoring* (CR) rules; we use the terms "CR-Prolog" and "ASP" interchangeably in this paper. ASP is a declarative language that represents recursive definitions, defaults, causal relations, and constructs that are difficult to express in classical logic formalisms. ASP is based on the stable model semantics, and encodes *default negation* and *epistemic disjunction*, e.g., unlike "$\neg$a", which implies that "*a is believed to be false*", "`not a`" only implies "*a is not believed to be true*" [8]. Each literal can hence be true, false, or unknown, and the *robot only believes statements that it is forced to believe*. ASP supports non-monotonic logical reasoning, i.e., adding a statement can reduce the set of inferences, which helps recover from errors due to reasoning with incomplete knowledge. Knowledge-based reasoning paradigms such as ASP are often criticized for requiring considerable prior knowledge, and for being unwieldy in large, complex domains. However, modern ASP solvers are used by an international community to reason efficiently with a large knowledge base or with incomplete knowledge [5].

A domain's description in ASP comprises a *system description* $\mathcal{D}$ and a *history* $\mathcal{H}$. $\mathcal{D}$ comprises a *sorted signature* Σ and axioms encoding the domain's dynamics. Our prior work explored spatial relations for classification tasks; Σ included *basic sorts*, e.g., $object$, $robot$, $size$, $relation$, and $surface$; *statics*, i.e., domain attributes that do not change over time, e.g., *obj_size(object, size)* and *obj_surface(obj, surface)*; and *fluents*, i.e., attributes whose values can be changed, e.g., *obj_relation(above, A, B)* implies object A is *above* object B. *The robot in this paper also plans and executes physical actions that cause changes in the domain.* Such a dynamic domain is modeled in our architecture by first describing the expanded Σ and transition diagram in action language $\mathcal{AL}_d$ [7]; this description is then translated to ASP statements. For the RA domain, Σ now includes the sort $step$ for temporal reasoning, additional fluents such as *in_hand(robot, object)*, actions such as *pickup(robot, object)* and *putdown(robot, object, location)*, and the relation *holds(fluent, step)* implying that a particular fluent holds true at a particular timestep. Axioms of the RA domain include ASP statements such as:

$$holds(in\_hand(robot, object), I+1) \;\leftarrow\; occurs(pickup(robot, object), I) \tag{1a}$$
$$holds(obj\_relation(above, A, B), I) \leftarrow holds(obj\_relation(below, B, A), I) \tag{1b}$$
$$\neg occurs(pickup(robot, object), I) \;\leftarrow holds(in\_hand(robot, object), I) \tag{1c}$$

which encode a causal law, a state constraint, and an executability condition respectively, e.g., Statement 1(a) implies that executing the "pickup" action causes the target object to be in the robot's grasp in the next time step; *our prior work only included state constraints* [20]. The axioms also encode some commonsense knowledge in the form of default statements that hold unless there is evidence to the contrary, e.g., "larger objects placed on smaller objects are unstable" is encoded in ASP as:

$$\begin{aligned}\neg holds(stable(A), I) \leftarrow\; & holds(obj\_relation(above, A, B), I),\\ & size(A, large),\; size(B, small),\; not\; holds(stable(A), I)\end{aligned} \tag{2}$$

where "not" denotes default negation. In addition to axioms, information extracted from the input images (e.g., spatial relations, object attributes) with sufficiently high probability is converted to ASP statements at that time step. Also, the domain's history $\mathcal{H}$ comprises records of fluents observed to be true or false at a particular time step, i.e., $obs(fluent, boolean, step)$, and of the execution of an action at a particular time step, i.e., $hpd(action, step)$. In [28] this notion was expanded to represent defaults describing the values of fluents in the initial state, e.g., "it is initially believed that a book is in the library", and exceptions, e.g., "a cookbook is in the kitchen".

To reason with the domain knowledge, our architecture constructs the CR-Prolog program $\Pi(\mathcal{D}, \mathcal{H})$, which includes Σ and axioms of $\mathcal{D}$, inertia axioms, reality checks, closed world assumptions for actions, and observations, actions, and defaults from $\mathcal{H}$. Every default also has a CR rule to let the robot assume the default's conclusion is false to restore consistency under exceptional circumstances. For instance, the statement in the ASP program: $\neg loc(X, library) \overset{+}{\leftarrow} book(X)$ is a CR rule that is triggered under exceptional circumstances to assume a book is not in the library as a potential explanation of an unexpected observation. The program for our RA domain is available online [21]. Once Π is constructed, planning, diagnostics, and inference can be reduced to computing *answer sets* of Π [8]. Any answer set represents the beliefs of the robot associated with Π; it is a description of a possible world and the set of literals of domain fluents and statics at any particular time step represents the *state* at that time step. Note that incorrect inferences can be drawn due to incomplete knowledge, noisy sensor input, or the use of a low threshold for elevating probabilistic information to statements in the ASP program. Non-monotonic logical reasoning enables the robot to recover from such errors, and not be very sensitive to the choice of the probability threshold. Also, although we do not describe it here, it is possible to model non-determinism (e.g., in action outcomes) in our architecture. In addition, work by others in our group has combined such logical reasoning at a coarse resolution with probabilistic reasoning over the relevant part of a finer resolution representation of the domain [28]. For ease of understanding and to focus on the interplay between non-monotonic logical reasoning and learning, we limit ourselves to logical reasoning at one resolution in this paper.

Classification: Similar to the approach in our prior work, for any given image, the robot tries to estimate the occlusion of objects and the stability of object configurations using ASP-based reasoning. If an answer is not found, or an incorrect answer is found (on labeled training examples), the robot automatically extracts relevant regions of interest (ROIs) from the corresponding image. Parameters of existing Convolutional Neural Network (CNN) architectures (e.g., Lenet [15], AlexNet [11]) are tuned to map information from each such ROI to the corresponding classification labels. An innovation of our prior work was to reason with knowledge of the task (e.g., estimating occlusion) to identify and ground only the relevant axioms and relations in the image under consideration to determine the ROIs [20]. In this paper, we reason about relevance over a sequence of steps to provide explanations, as described in Section 3.2.

Decision tree induction: Images used to train the CNNs are considered to contain information about missing or incorrect constraints related to occlusion and stability. Image features and spatial relations extracted from ROIs in each such image, along with the known labels for occlusion and stability (during training), are used to incrementally learn a decision tree summarizing the corresponding state transitions. The learning process repeatedly splits a node based on an unused attribute likely to provide the highest reduction in entropy. Next, branches of the tree that satisfy minimal thresholds on purity at the leaf ($\geq 95\%$ samples in one class) and on the level of support from labeled examples ($\geq 5\%$) are used to construct candidate axioms. Candidates are validated and those without a minimal level of support ($\geq 5\%$) on unseen examples are removed. These thresholds are set to identify a small number of highly likely axioms, and small changes to thresholds do not affect performance. Also, the thresholds can be revised to achieve other outcomes, e.g., they can be lowered significantly to identify default constraints.

Unlike our prior work, we introduce new strategies to process noisy images of more complex scenes. First, we use an ensemble learning approach, retaining only axioms that are identified over a number of cycles of learning and validation. Second, different versions of the same axiom are merged to remove over-specifications, e.g.:

$$\neg stable(A) \leftarrow obj\_relation(above, A, B),\ obj\_surface(B, irregular) \tag{3a}$$

$$\begin{aligned}\neg stable(A) \leftarrow\ & obj\_relation(above, A, B),\ obj\_surface(B, irregular), \\ & obj\_size(B, large)\end{aligned} \tag{3b}$$

where Statement 3(b) can be removed because the size of the object at the bottom of a stack does not provide any additional information about instability given that it has an irregular surface. If the robot later observes that a large object, even with an irregular surface, can support a small object, the axiom will be revised suitably. Specifically, axioms with the same head and some overlap in the body are grouped. Each combination of one axiom from each group is encoded in an ASP program along with axioms that are not in any group. This program is used to classify ten labeled scenes, only retaining axioms in the program that provides the highest accuracy on these scenes. Third, to filter axioms that cease to be useful, the robot associates each axiom with a *strength* that decays exponentially over time if it is not reinforced, i.e., not used or learned again. Any axiom whose strength falls below a threshold is removed. Other work in our group has explored the learning of actions, causal laws, and executability conditions in simulated domains [29]. Here, we only consider the learning of constraints and explore the effect of the learned axioms on the ability to provide explanations.

3.2 Relational Descriptions as Explanations

Our architecture's new components exploit the interplay between representation, reasoning, and learning to provide the desired relational descriptions of decisions, beliefs, and the outcomes of hypothetical events.

Interaction interface and control loop: Human interaction with our architecture is through speech or text. Existing algorithms, software, and a controlled (domain-specific) vocabulary are used to parse human verbal input and to provide a verbal response when appropriate. Specifically, verbal input from a human is transcribed into text drawn from the controlled vocabulary. This (or the input) text is labeled using a part-of-speech (POS) tagger, and normalized with the lemma list [27] and related synonyms and antonyms from WordNet [17]. The processed text helps identify the type of request, which may correspond to a desired goal or a question about decisions, beliefs, or the outcomes of hypothetical events. In the former case, the goal is sent to the ASP program for planning. The robot executes the plan, replanning when unexpected action outcomes cannot be explained, until the goal is achieved. In the latter case, the "Program Analyzer" considers the domain knowledge (including inferred beliefs that are computed as needed) and processed human input to automatically identify relevant axioms and literals. These literals are inserted into generic response templates based on the controlled vocabulary, resulting in human-understandable (textual) descriptions that are converted to synthetic speech if needed.

Program analyzer: Algorithm 1 describes the approach for automatically identifying and reasoning with the relevant information to construct relational descriptions in response to questions or requests. We illustrate the working of the algorithm in the context of four types of such *explanatory* questions or requests at a particular time step I.

1. **Plan description** When asked to describe a particular plan, the robot parses the related answer set(s) to extract a sequence of actions of the form *occurs(action1, step1), ..., occurs(actionN, stepN)* (line 3 in Algorithm 1). These actions are used to construct the response.
2. **Action justification: Why action X at step I?** To justify the execution of any particular action at a particular time step:
   (a) For each action that occurred after time step I, the robot examines relevant executability condition(s) and identifies literal(s) that would prevent the action's execution at step I (lines 5-7). For the goal of placing the $orange\_block$ on the

Algorithm 1: (**Program Analyzer**) Construct answer to input question

```
Input  : Literal of input question; $\Pi(\mathcal{D}, \mathcal{H})$; answer templates.
Output: Answer and answer Literals.
// Compute answer set
1  AS = AnswerSet($\Pi$)
2  if $question = plan\ description$ then
       // Retrieve all actions from answer set
3      answer_literals = Retrieve(AS, actions)
4  else if $question =$ "$why\ action\ X\ at\ step\ I?$" then
       // Extract actions after step I
5      next_actions = Retrieve(AS, actions for step $>$ I)
       // Extract axioms influencing these actions
6      relevant_axioms = Retrieve($\Pi$, head = $\neg$ next_actions)
       // Extract relevant literals from Answer Set
7      relevant_literals = Retrieve(AS, Body(relevant_axioms) $\in I \wedge \notin I+1$)
       // Output literals
8      answer_literals = pair(relevant_literals, next_actions)
9  else if $question =$ "$why\ not\ action\ X\ at\ step\ I?$" then
       // Extract axioms relevant to action
10     relevant_axioms = Retrieve($\Pi$, head = $\neg$ occurs(X))
       // Extract relevant literals from Answer Set
11     answer_literals = Retrieve(AS, Body(relevant_axioms) $\in I \wedge \notin I+1$)
12 else if $question =$ "$why\ belief\ Y\ at\ step\ I?$" then
       // Extract axioms influencing this belief
13     relevant_axioms = Retrieve($\Pi$, head = Y)
       // Extract body of axioms
14     answer_literals = Recursive_Examine(AS, Body(relevant_axioms))
15 Construct_Answer(answer_literals, answer_templates)
```

table in Figure 1b, assume that the action executed are *occurs(pickup(robot, blue_block), 0)*, *occurs(putdown(robot, blue_block), 1)*, and *occurs(pickup(robot, orange_block), 2)*. If the focus is on the first $pickup$ action, an executability condition related to the second $pickup$ action:

$$\neg occurs(pickup(robot, A), I) \ \leftarrow \ holds(obj\_relation(below, A, B), I)$$

is ground in the scene to obtain *obj_relation(below, orange_block, blue_block)* as a literal of interest.

(b) If any identified literal is in the answer set at the time step of interest (0 in the current example), and is absent (or its negation is present) in the next step, it is taken to be a reason for executing the action under consideration (line 7).

(c) The condition modified by the execution of the action of interest is paired with the subsequent action to construct the answer to the question (line 8). For instance, the question "Why did you pick up the blue block at time step 0?", receives the answer "I had to pick up the orange block, and the orange block was located below the blue block".

A similar approach is used to justify the selection of any particular action in any particular plan that has been computed but not yet executed.

3. **Hypothetical actions: Why not action X at step I?** For questions about actions not selected for execution:
   (a) The robot identifies executability conditions that have the hypothetical action in the head, i.e., conditions that prevent the action from being selected during planning (line 10 in Algorithm 1).
   (b) For each identified executability condition, the robot examines whether literals in the body are satisfied in the corresponding answer set (line 11). If so, these literals are used to construct the answer.

   Suppose action *putdown(robot, blue_block, table)* occurred at step 1 in Figure 1b. For the question "Why did you not put the blue cube on the tennis ball at time step 1?", the following related executability condition is identified:

$$\neg occurs(putdown(robot,\ A,\ B), I)\ \leftarrow\ has\_surface(B,\ irregular)$$

   which implies that an object cannot be placed on another object with an irregular surface. The answer set indicates that the tennis ball has an irregular surface. The robot provides the answer "Because the tennis ball has an irregular surface".

4. **Belief query: Why belief Y at step I?** To explain any particular belief, the robot searches for support axioms in which the belief is the head and the corresponding body is satisfied in the current state. The search is repeated recursively for literals in the body until no more axioms are found (lines 13-14). These relevant literals are used to construct the answer. For instance, to explain the belief that object ob_1 is unstable in step I, the robot finds the support axiom:

$$\neg holds(stable(ob_1), I)\ \leftarrow\ holds(small\_base(ob_1), I)$$

   Assume that the current beliefs include that ob_1 has a small base. Searching for why ob_1 is believed to have a small base identifies the axiom:

$$\begin{aligned} holds(small\_base(ob_1), I)\ \leftarrow\ & holds(relation(below,\ ob_2,\ ob_1),\ I), \\ & has\_size(ob_2,\ small),\ has\_size(ob_1,\ big) \end{aligned}$$

   Asking "why do you believe object ob_1 is unstable at step I?" would provide the answer "Because object ob_2 is below object ob_1, ob_2 is small, and ob_1 is big".

Robot platform: Our prior work explored scene understanding tasks with simulated images, but this paper considers a robot that also plans and executes actions to achieve the desired goals. As stated earlier, we use a Baxter robot that manipulates objects on a tabletop. The Baxter uses probabilistic algorithms to process inputs from its cameras, e.g., to extract information about the presence of objects, their attributes, and the spatial relations between objects, from images such as Figure 1b. The Baxter also uses built-in probabilistic motion planning algorithms to execute primitive manipulation actions, e.g., to grasp and pick up objects. Observations obtained with a high probability are elevated to literals associated with complete certainty in the ASP program. Recall that our architecture's non-monotonic logical reasoning capability enables the robot to identify and recover from errors caused by adding incorrect information to the ASP program.

4 EXPERIMENTAL SETUP AND RESULTS

Section 4.1 describes the setup for evaluating the ability to construct relational descriptions of decisions, beliefs, and hypothetical events. Section 4.2 then describes some execution traces and Section 4.3 discusses quantitative results.

4.1 Experimental Setup

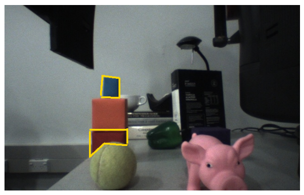
Experiments were designed to evaluate the following hypotheses:

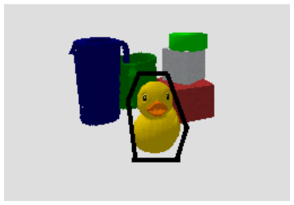
H1 : reasoning with incrementally learned and merged state constraints improves the quality of plans generated; and

H2 : exploiting the links between reasoning and learning improves the accuracy of the descriptions provided to explain the decisions and beliefs.

These hypotheses and the underlying capabilities of our architecture were evaluated considered four kinds of explanatory requests and questions: (i) describing the plan; (ii) justifying the execution of an action at a given time step; (iii) justifying not choosing a hypothetical action; and (iv) justifying particular beliefs. As stated in Algorithm 1 in Section 3.2, the same methodology can also be adapted to address other requests nd questions. The quality of a plan was measured in terms of the ability to compute minimal plans, i.e., plans with the least number of actions to achieve the desired goals. The quality of an explanation was measured in terms of precision and recall of the literals in the answer provided by our architecture in comparison with the expected response. The expected ("ground truth") response was provided in a semi-supervised manner based on manual input and automatic selection of relevant literals.

Experimental trials considered images from the robot's camera and simulated images. Real world images contained $5-7$ objects of different colors, textures, shapes, and sizes in the RA domain of Example 1. The objects included cubes, a pig, a capsicum, a tennis ball, an apple, an orange, and a pot. These objects were either stacked on each other or spread on the table—see Figure 1b. A total of 40 configurations were created, each with five different goals for planning and four different questions for each plan, resulting in a total of 200 plans and 800 questions. Since evaluating applicability to a wide range of objects and scenes is difficult on robots, we also used a real-time physics engine (Bullet) to create 40 simulated images, each with $7-9$ objects ($3-5$ stacked and the remaining on a flat surface). Objects included cylinders, spheres, cubes, a duck, and five household objects from the Yale-CMU-Berkeley dataset (apple, pitcher, mustard bottle, mug, and box of crackers). We once again considered five different goals for planning and four different questions for each plan, resulting in the same number of plans (200) and questions (800) as with the real world data.


(a) Execution Example 1.


(b) Execution Example 3.

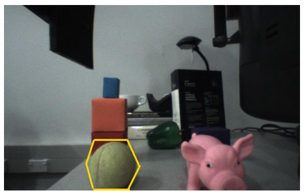

(c) Additional example.

Fig. 3: (a) Relation between blue cube and red cube is important for the explanation in Execution Example 1; (b) The rubber duck is the focus of attention in Execution Example 3; and (c) Example of another trial (not described in this paper) in which a tennis ball plays an important role in the explanation constructed by our architecture.

To explore the interplay between reasoning and learning, we focused on the effect of learned knowledge on planning and constructing explanations. Specifically, we ran experiments with and without some learned constraints in the knowledge base. Learned constraints were revised over time in our architecture, as described in Section 3.1, whereas the learned constraints were not used by the baselines for planning and explanation generation. During planning, we measured the number of optimal, sub-optimal, and incorrect plans, and the planning time. An *optimal* plan is a minimal plan that achieves the goal; a *sub-optimal* plan requires more than the minimum number of steps and/or has to assume an unnecessary exception to default knowledge; and an *incorrect* plan leads to undesirable outcomes and fails to achieve the desired goal. We discuss the quantitative results of these experimental trials after describing some execution traces of our architecture's operation.

4.2 Execution Trace

The following execution traces illustrate our architecture's ability to construct relational descriptions explaining the decisions, beliefs, and the outcomes of hypothetical actions.

Execution Example 1 *[Plans, actions, and beliefs]*
Consider a robot that starts with objects as shown in Figure 1b. The robot is assigned the goal of achieving a state in which the red block is on top of the orange block, i.e., *holds(relation(on, red_block, orange_block), I)*. The robot answers the following questions *after executing* a plan and successfully achieving the assigned goal:

– **Human:** "Please describe the plan."
  Baxter: "I picked up the blue block. I put the blue block on the table. I picked up the orange block. I put the orange block on the table. I picked up the red block. I put the red block on the orange block."
– The human may ask the robot to justify a particular action in the executed plan.
  Human: "Why did you pick up the blue block at step 0?"
  Baxter: "Because I had to pick up the red block, and it was below the blue block."
  The constructed answer is also used to automatically highlight the relevant image regions that influenced this answer, as shown in Figure 3a.

– The human now may ask about particular actions that were not considered.
  Human: ”Why did you not put down the orange block on the blue block?”
  Baxter: ”Because the blue block is small.” In the absence of a reference to a particular time step, the robot answers the question based on the single instance (in the executed plan) of putting the orange block on another block or surface. In addition, the answer is based on learned default knowledge, i.e., that objects with a small base are typically unstable.
– The human may also ask about particular beliefs.
  Human: ”Why did you believe that the red block was below the blue block in the initial state?”
  Baxter: ”Because I observed the red block below the blue block in step zero.”

Execution Example 2 *[Reasoning and explanation]*
Continuing with the previous example, the subsequent interactions are as follows:

– **Human:** ”Put the tennis ball on the blue cube.”
  The goal *holds(relation(on, ball, blue_block), I)* is encoded in the ASP program for planning. The default rule about configurations with small bases being unstable prevents the robot from placing objects on the blue block. At the same time, the robot has to compute plans to achieve the assigned goal. This causes an inconsistency that is resolved by invoking a CR rule and planning to place the ball on the top of the blue block. The following interaction takes place after plan execution:
– **Human:** ”Please describe the plan you executed.”
  Robot: ”I picked up the ball. I put the ball on the blue block.”
– The human may now explore the belief of the agent that requires it to consider exceptions to the default knowledge:
  Human: ”Why do you believe the ball is on the blue block?”
  Robot: ”Because I observed the ball on the blue block in step one.”

Combining reasoning with constructing explanations thus allows the robot to adapt to unforeseen exceptions.

Execution Example 3 *[Learning and explanation]*
In some situations, the robot may be unable to respond to the human request or question because it is not possible to achieve the desired object configuration or belief. Even in such cases, our architecture enables the robot to answer explanatory questions. For instance, consider the simulated scene in Figure 3b, with the following interaction:

– **Human:** ”Please put the pitcher on the duck.”
  This action is not executed because of a constraint learned during a previous trial that any object configuration that has an object on another object with an irregular surface will be unstable.
– If asked, the robot can justify its decision of not executing the action.
  Human: ”Why did you not put the pitcher on the duck?”.
  Robot: ”Because the duck has an irregular surface.”
  The image region relevant to the construction of the robot’s answer to the question posed by the human is automatically highlighted in the corresponding image, as indicated in Figure 3b above.

This example illustrates how integrating reasoning and learning helps justify the decision to not execute a requested action that will have an unfavorable outcome.

Overall, these and other such examples demonstrate how our architecture uses a relational representation, automatically reasons with just the relevant knowledge, incrementally revises axioms, and identifies image regions, attributes, and actions contributing to particular decisions and beliefs. Since the same set of samples are used to learn axioms and train the deep networks, our approach also provides a partial understanding of the behavior of learned deep networks.

4.3 Experimental Results

In this section, we discuss quantitative results of evaluating the desired hypotheses. The first set of experiments was designed as follows to evaluate hypothesis **H1**:

1. Forty initial object configurations were arranged (similar to that in Figure 1a). The Baxter automatically extracted information (e.g., attributes, spatial relations) from images corresponding to top and frontal views (using the cameras on the left and right grippers), and encoded it in the ASP program as the initial state.
2. For each initial state, five goals were randomly chosen and encoded in the ASP program. The robot reasoned with the existing knowledge to create plans for these 200 combinations (40 initial states, five goals).
3. The plans were evaluated in terms of the number of optimal, sub-optimal and incorrect plans, and planning time.
4. Experiments were repeated with and without the learned axioms.
5. Steps 1-4 (above) were also repeated with the simulated images.

Since the number of plans and planning time vary depending on the initial conditions and the goal, we conducted paired trials with and without the learned constraints included in the ASP program used for reasoning. The initial conditions and goal were identical in each paired trial, and differed between different paired trials. Then, we expressed the number of plans and the planning time with the learned constraints as a fraction of the corresponding values obtained by reasoning without the learned constraints. The average of these fractions over all the trials is reported in Table 1. In addition, we computed the number of optimal, sub-optimal, and incorrect plans in each trial as a fraction of the total number of plans in the trial; we did this separately with and without using the learned axioms for reasoning, and the average over all trials is summarized in Table 2. These results indicate that using the learned axioms for reasoning significantly reduced the search space, resulting in a much smaller number of plans and a substantial reduction in the planning time. In addition, when the robot used the learned axioms for reasoning, it resulted in a much smaller number of sub-optimal plans and eliminated all incorrect plans. Also, each such sub-optimal plan was created only when the corresponding goal could not be achieved without creating an exception to a default, e.g., stacking an object on a small base. Without the learned axioms, a larger fraction of the plans are sub-optimal or incorrect. These results support hypothesis **H1**.

The second set of experiments was designed as follows to evaluate hypothesis **H2**:

Table 1: Number of plans and planning time with the learned axioms expressed as a fraction of the values without the learned axioms. Reasoning with the learned axioms improves performance.

	Ratio (with/without)	
Measures	**Real scenes**	**Simulated scenes**
Number of plans	0.36	0.33
Planning time	0.66	0.89

Table 2: Number of optimal, sub-optimal, and incorrect plans expressed as a fraction of the total number of plans. Reasoning with the learned axioms improves performance.

	Real Scenes		**Simulated Scenes**	
Plans	**Without**	**With**	**Without**	**With**
Optimal	0.31	0.82	0.15	0.49
Sub-optimal	0.12	0.18	0.31	0.51
Incorrect	0.57	0	0.54	0

1. For each of the 200 combinations (40 configurations, five goals) from the first set of experiments with real-world data, we considered knowledge bases with and without the learned axioms and had the robot compute plans to achieve the goals.
2. The robot had to describe the plan and justify the choice of a particular action (chosen randomly) in the plan. Then, one parameter of the chosen action was changed randomly to pose a question about why this new action could not be applied. Finally, a belief related to the previous two questions had to be justified.
3. The literals present in the answers were compared against the expected literals in the ideal response, with the average precision and recall scores reported in Table 3.
4. We also performed these experiments separately for simulated images, with the average results summarized in Table 4.

Tables 3, 4 show that when the learned axioms were used for reasoning, the precision and recall of relevant literals (for constructing the explanation) were higher than when the learned axioms were not included. The improvement in performance is particularly pronounced when the robot has to answer questions about actions that it has not actually executed. The precision and recall rates were reasonable even when the learned axioms were not included; this is because not all the learned axioms are needed to accurately answer each explanatory question. When the learned axioms were used for reasoning, errors were very rare and corresponded to some additional literals being included in the answer (i.e., over-specified explanations). In addition, when we specifically removed axioms related to the goal under consideration, precision and recall values were much lower. Furthermore, there was noise in both sensing and actuation, especially in the robot experiments. For instance, recognition of spatial relations, learning of constraints, and manipulation have approximate error rates of 15%, $5-10\%$, and 15% respectively. Experimental results thus indicate that coupling reasoning and learning to inform and guide each other enables the robot to provide accurate relational descriptions in re-

Table 3: (**Real scenes**) Precision and recall of retrieving relevant literals for constructing answers to questions with and without using the learned axioms for reasoning. Using the learned axioms significantly improves the ability to provide accurate explanations.

	Precision		**Recall**	
Query Type	**Without**	**With**	**Without**	**With**
Plan description	91.77%	100%	91.77%	100%
Why X?	91.75%	94.75%	91.98%	94.75%
Why not X?	93.57%	95.16%	87.91%	98.88%
Belief	93.04%	99.35%	93.63%	100%

Table 4: (**Simulated scenes**) Precision and recall of retrieving relevant literals for constructing answers to questions with and without reasoning with learned axioms. Using the learned axioms significantly improves the ability to provide accurate explanations.

	Precision		**Recall**	
Query Type	**Without**	**With**	**Without**	**With**
Plan description	90.04%	100%	90.04%	100%
Why X?	93.0%	93.0%	93.0%	93.0%
Why not X?	93.22%	100%	89.43%	98.04%
Belief	97.22%	99.19%	97.9%	100%

sponse to questions about decisions, beliefs, and the outcomes of hypothetical actions. This supports hypothesis **H2**. Additional examples of images, questions, and answers, are in our open source repository [21].

5 Conclusions

This paper described a cognitive systems-inspired approach that enables an integrated robot system to explain its decisions, beliefs, and the outcomes of hypothetical actions. These explanations are constructed on-demand in the form of descriptions of relations between relevant objects, actions, and attributes of the domain. We have implemented this approach in an architecture that combines the complementary strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, deep learning, and inductive learning. In the context of some scene understanding and planning tasks performed in simulation and a physical robot, we have demonstrated that our architecture exploits the interplay between knowledge-based reasoning and data-driven learning. It automatically identifies and reasons with the relevant information to efficiently construct the desired explanations, with both the planning and explanation generation performance improving significantly when previously unknown state constraints are learned incrementally and used for subsequent reasoning.

Our architecture opens up multiple avenues for further research. First, we will integrate the ability to learn other kinds of axioms and the corresponding knowledge represented as an ASP program. We will do so by building on the approach developed in our group by combining non-monotonic logical reasoning, active learning, and relational reinforcement learning [29]. Second, we will explore more complex domains, tasks, and explanations, reasoning with relevant knowledge at different tightly-coupled resolutions for scalability [28]. We are specifically interested in exploring scenarios in which there is ambiguity in the questions (e.g., it is unclear which of two occurrences of the $pickup$ action the human is referring to), or the explanation is needed at a different level of abstraction, specificity, or verbosity. We will do so by building on a related theory of explanations [30]. Third, we will use our architecture to better understand the behavior of deep networks. The key advantage of using our architecture is that it uses reasoning to guide learning. Unlike "end to end" data-driven learning methods based on deep networks, our architecture uses reasoning to trigger learning only when existing knowledge is insufficient to perform the desired task(s). The long-term objective is to develop an architecture that exploits the complementary strengths of knowledge-based reasoning and data-driven learning for the reliable and efficient operation of robots in complex, dynamic domains.

Acknowledgments

This work was supported in part by the Asian Office of Aerospace Research and Development award FA2386-16-1-4071 and the U.S. Office of Naval Research Science of Autonomy Award N00014-17-1-2434. Opinions and conclusions reported in this paper are those of the authors.

References

1. Anjomshoae, S., Najjar, A., Calvaresi, D., Framling, K.: Explainable agents and robots: Results from a systematic literature review. In: International Conference on Autonomous Agents and Multiagent Systems. Montreal, Canada (2019)
2. Assaf, R., Schumann, A.: Explainable Deep Neural Networks for Multivariate Time Series Predictions. In: International Joint Conference on Artificial Intelligence. pp. 6488–6490. Macao, China (July 2019)
3. Borgo, R., Cashmore, M., Magazzeni, D.: Towards Providing Explanations for AI Planner Decisions. In: IJCAI Workshop on Explainable Artificial Intelligence. pp. 11–17 (2018)
4. David, H., Tom, B.: An Enquiry concerning Human Understanding: A Critical Edition. Oxford University Press (2000)
5. Erdem, E., Patoglu, V.: Applications of ASP in Robotics. Kunstliche Intelligenz **32**(2-3), 143–149 (2018)
6. Friedman, M.: Explanation and scientific understanding. Philosophy **71**(1), 5–19 (1974)
7. Gelfond, M., Inclezan, D.: Some Properties of System Descriptions of AL_d. Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming **23**(1-2), 105–120 (2013)
8. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning and the Design of Intelligent Agents. Cambridge University Press (2014)
9. de Kleer, J., Williams, B.C.: Diagnosing Multiple Faults. Artificial Intelligence **32**, 97–130 (1987)
10. Koh, P.W., Liang, P.: Understanding Black-box Predictions via Influence Functions. In: International Conference on Machine Learning. pp. 1885–1894 (2017)

11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet Classification with Deep Convolutional Neural Networks. In: Neural Information Processing Systems. pp. 1097–1105 (2012)
12. Laird, J.E.: The Soar Cognitive Architecture. The MIT Press (2012)
13. Langley, P., Meadows, B., Sridharan, M., Choi, D.: Explainable Agency for Intelligent Autonomous Systems. In: Innovative Applications of Artificial Intelligence (2017)
14. Langley, P.: Progress and Challenges in Research on Cognitive Architectures. In: AAAI Conference on Artificial Intelligence. San Francisco, USA (February 4-9, 2017)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
16. Lewandowsky, S., Mundy, M., Tan, G.: The Dynamics of Trust: Comparing Humans to Automation. Journal of Experimental Psychology: Applied **6**(2), 104 (2000)
17. Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM **38**(11), 39–41 (1995)
18. Miller, T.: Explanations in Artificial Intelligence: Insights from the Social Sciences. Artificial Intelligence **267**, 1–38 (2019)
19. Mota, T., Sridharan, M.: Incrementally Grounding Expressions for Spatial Relations between Objects. In: International Joint Conference on Artificial Intelligence. pp. 1928–1934 (2018)
20. Mota, T., Sridharan, M.: Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots. In: Robotics Science and Systems (2019)
21. Mota, T., Sridharan, M.: Scene Understanding, Reasoning, and Explanation Generation (2020), `https://github.com/tmot987/Scenes-Understanding`
22. Norcliffe-Brown, W., Vafeais, E., Parisot, S.: Learning Conditioned Graph Structures for Interpretable Visual Question Answering. In: Neural Information Processing Systems. Montreal, Canada (December 3-8, 2018)
23. Read, S.J., Marcus-Newhall, A.: Explanatory coherence in social explanations: A parallel distributed processing account. Personality and Social Psychology **65**(3), 429 (1993)
24. Ribeiro, M., Singh, S., Guestrin, C.: Why Should I Trust You? Explaining the Predictions of Any Classifier. In: International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
25. Samek, W., Wiegand, T., Mller, K.R.: Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. ITU Journal: ICT Discoveries: The Impact of Artificial Intelligence on Communication Networks and Services **1**, 1–10 (2017)
26. Seegebarth, B., Müller, F., Schattenberg, B., Biundo, S.: Making Hybrid Plans More Clear to Human Users: A Formal Approach for Generating Sound Explanations. In: International Conference on Automated Planning and Scheduling (2012)
27. Someya, Y.: Lemma List for English Language (1998)
28. Sridharan, M., Gelfond, M., Zhang, S., Wyatt, J.: REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Journal of Artificial Intelligence Research **65**, 87–180 (May 2019)
29. Sridharan, M., Meadows, B.: Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. Advances in Cognitive Systems **7** (2018)
30. Sridharan, M., Meadows, B.: Towards a Theory of Explanations for Human-Robot Collaboration. Kunstliche Intelligenz **33**(4), 331–342 (2019)
31. Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., Tenenbaum, J.B.: Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In: Neural Information Processing Systems. Montreal, Canada (December 3-8, 2018)
32. Zhang, Y., Sreedharan, S., Kulkarni, A., Chakraborti, T., Zhuo, H.H., Kambhampati, S.: Plan explicability and predictability for robot task planning. In: International Conference on Robotics and Automation. pp. 1313–1320 (2017)