Answer me this: Constructing Disambiguation Queries for Explanation Generation in Robotics

Tiago Mota,[1] **Mohan Sridharan** [2]

[1]The University of Auckland, Auckland, New Zealand
[2] University of Birmingham, Birmingham, United Kingdom
tmot987@aucklanduni.ac.nz, m.sridharan@bham.ac.uk

Abstract

Robot assisting humans in complex domains can operate more effectively if they can describe their decisions and beliefs. Such transparency is difficult to achieve in integrated robot systems that include methods for learning from data and reasoning with incomplete commonsense domain knowledge. The architecture described in this paper seeks to address the associated challenges by building on a baseline system that supports non-monotonic logical reasoning with incomplete commonsense domain knowledge, data-driven learning from a limited set of examples, and inductive learning of previously unknown axioms governing domain dynamics. In the context of enabling a simulated robot to provide on-demand, relational descriptions of its decisions and beliefs, we introduce an interactive system that automatically traces beliefs, and constructs and poses queries to solicit human input and reduce ambiguity in the robot's beliefs. We present results of evaluation in scene understanding and planning tasks to demonstrate the abilities of our architecture.

Introduction

Consider a robot[1] estimating the occlusion of objects and stability of object structures while arranging objects in desired configurations on a flat table; Figure 1 shows such a scene. 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 learns previously unknown axioms governing domain dynamics, and provides on-demand *explanatory* descriptions of its decisions and beliefs in the form of relations between domain attributes, robot attributes, and robot actions. For instance, assume that the goal in Figure 1 is to have the yellow cylinder on the partially occluded green block, and that the plan is to move the objects on the yellow cylinder, and the yellow block (on the green block), to the table before the yellow cylinder on the green block. When the robot is asked to justify a step in the plan, e.g., "why do you want to put the yellow duck on the table first?", it correctly answers "the yellow duck is on the yellow cylinder that I need to put on the green block". However, the question "why did you pick up the yellow object?" posed after plan execution is ambiguous both in terms of the object and time step being referred to. The robot recognizes this ambiguity, reasons that the human is probably not referring to the rubber duck, asks "are you referring to the yellow cylinder or yellow block?", and uses the answer to provide the response to the original query from the human.

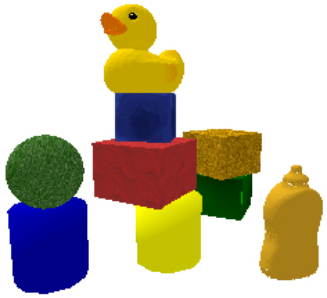

Figure 1: An illustrative simulated scene with objects in different configurations. Any reference to a yellow object in this image is ambiguous.

Our work seeks to enable such on-demand *explanations* of a robot's decisions and beliefs, and hypothetical situations, in the form of descriptions of relations between relevant objects, actions, and domain attributes. This "explainability" can help improve the underlying algorithms and establish accountability. This is challenging to achieve with integrated robot systems that include knowledge-based reasoning methods (e.g., for planning) and data-driven (deep) learning algorithms (e.g., for pattern recognition), especially when the human query is ambiguous. Inspired by research in cognitive systems that indicates the benefits of coupling different representations and reasoning schemes (Laird 2012; **?**), our architecture combines the complementary strengths of knowledge representation tools and data-driven methods to provide transparent decision making. It builds on our prior work that combined non-monotonic logical reasoning and deep learning for scene understanding in simulated images, and our proof of concept work on learning previously unknown state constraints while constructing descriptions of related decisions and beliefs (Mota and Sridharan 2019; Mota, Sridharan, and Leonardis 2020). Here, we summarize these capabilities and describe extensions to:

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

- Automatically trace the evolution of any given belief, using knowledge representation tools to infer the application of a suitable sequence of known or incrementally learned axioms governing domain dynamics.
- Interactively address ambiguity in human queries by introducing and using heuristic measures of ambiguity, human confusion, and the relative utility of domain attributes, to construct disambiguation queries.

In our implementation, non-monotonic logical reasoning is achieved using Answer Set Prolog (Gelfond and Kahl 2014), and existing network models are adapted for deep learning. We illustrate our architecture's capabilities in the context of a robot (i) computing and executing plans to arrange objects in desired configurations; and (ii) estimating occlusion of objects and stability of object configurations.

Related Work

Early work on explanation generation drew on research in psychology and linguistics to characterize explanations in terms of generality, objectivity, connectivity, relevance, and content (Friedman 1974). Studies with human subjects supported these findings (Read and Marcus-Newhall 1993), and computational methods were developed for explaining unexpected outcomes (de Kleer and Williams 1987).

There is much interest in understanding the operation of AI and machine learning methods, and making automation more acceptable (Miller 2019). Recent work on *explainable AI/planning* can be broadly categorized into two groups. Methods in one group modify or transform learned models or reasoning systems to make their decisions more interpretable, e.g., by tracing decisions to inputs (Koh and Liang 2017), learning equivalent interpretable models (Ribeiro, Singh, and Guestrin 2016), or biasing a planning system towards making decisions easier for humans to understand (Zhang et al. 2017). Methods in the other group provide descriptions that make decisions more transparent, e.g., describing planning decisions (Borgo, Cashmore, and Magazzeni 2018), combining logical reasoning with interface design to help humans understand a plan (Bercher et al. 2014), or using rules associated with monotonic operators to define *proof trees* that provide a declarative view (i.e., explanation) of computation (Ferrand, Lessaint, and Tessier 2006). There has also been work on describing solutions obtained through non-monotonic logical reasoning (Fandinno and Schulz 2019). These methods are often agnostic to how an explanation is structured or assume comprehensive domain knowledge. Methods are also being developed to make the operation of deep networks more interpretable, e.g., by computing gradients and *heat maps* of relevant features (Assaf and Schumann 2019; Samek, Wiegand, and Mller 2017), or with deep networks trained to answer questions about images of scenes (Yi et al. 2018).

A key requirement for providing explanations is the ability to identify and use the information relevant to the query or request. The robot can address any ambiguity in human input by formulating effective follow up questions and soliciting clarification. AI researchers have evaluated how the type of question posed by an agent affects the quality of human responses (Rosenthal, Veloso, and Dey 2012; Gervasio, Yeh, and Myers 2011). Other approaches studied how the type of questions posed by the agent affects the the agent's ability to learn from the answers (Gonzalez-Pacheco et al. 2018) or the ability to learn from human demonstration (Cakmak and Thomaz 2012). These methods focused on measuring the accuracy of the information obtained from the human or to learn from the human response. There has also been work on building questions that minimize the potential ambiguity of the human's response(Myagmarjav and Sridharan 2015). These methods do not reason effectively with incomplete domain knowledge to construct questions or improve the quality of the explanations.

Our work focuses on integrated systems that use a combination of knowledge-based and data-driven algorithms to represent, reason with, and learn from incomplete commonsense domain knowledge and noisy observations. We seek to enable such robots to generate accurate relational descriptions of decisions, beliefs, and hypothetical situations, capabilities that are not supported by existing systems (Anjomshoae et al. 2019; Miller 2019). Towards this objective, this paper describes the automatic tracing of beliefs and the construction of disambiguation questions.

Architecture

Figure 2 depicts the overall architecture for knowledge representation, explainable reasoning, and interactive learning. This architecture first attempts to use non-monotonic logical reasoning with incomplete commonsense domain knowledge to complete any given visual scene understanding task. If it is unable to do so, reasoning automatically identifies relevant regions of relevant images to inform and guide the adaptation of deep network models for this task. Also, the examples processed by the deep network are used to induce previously unknown axioms that are used for subsequent reasoning. The program analyzer takes the parsed human (verbal or text) input, and triggers reasoning and/or constructs an explanation of the desired decisions and beliefs. In this paper, we introduce a component that enables the robot to trace and reason with beliefs, and to automatically pose disambiguation questions input. For completeness, we summarize all components below but focus primarily on the disambiguation module, and recent changes in existing modules, in the context of the following running example.

Example Domain 1 *[Assistive Robot (AR) Domain]* Consider an assistive robot analyzing cluttered scenes of objects stacked in different configurations on a flat surface. The objective of the robot is to: (i) rearrange object structures according to human requirements; and (ii) provide on-demand, relational descriptions of decisions and beliefs before, during, or after action execution. The robot's prior domain knowledge includes some object attributes such as $size$ (small, medium, large), $surface$ (flat, irregular) and $shape$ (cube, apple, duck), and the spatial $relation$ between objects (above, below, front, behind, right, left, close). The agent can visually observe and move the objects to achieve the desired object configurations. Domain knowledge also includes axioms governing domain dynamics but some ax-

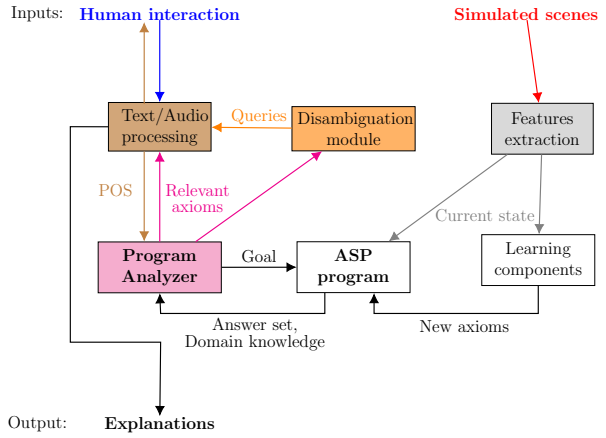

Figure 2: Architecture integrates complementary strengths of non-monotonic logical reasoning, deep learning, and inductive learning. Disambiguation module automatically constructs and poses questions to identify relevant information and provide responses to human queries and requests.

ioms may be unknown, e.g.:

- Placing an object on top of an object with an irregular surface causes instability;
- Removing all objects in front of an object causes this object to be not occluded.

The previously unknown axioms can be learned, merged with existing axioms, and used for subsequent reasoning.

Knowledge Representation and Reasoning

To represent and reason with domain knowledge, we use CR-Prolog, an extension to Answer Set Prolog (ASP); we use the terms CR-Prolog and ASP interchangeably in this paper. ASP is a declarative language that can represent recursive definitions, defaults, causal relations, and language constructs that occur frequently in non-mathematical domains, and are difficult to express in classical logic formalisms. ASP is based on the stable model semantics and supports concepts such as *default negation* (negation by failure) 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*" (Gelfond and Kahl 2014). Each literal can thus be true, false or unknown and the *robot only believes that which it is forced to believe*. ASP supports non-monotonic logical reasoning, i.e., adding a statement can reduce the set of inferred consequences, aiding in the recovery from errors due to reasoning with incomplete knowledge. ASP and other knowledge-based paradigms are often criticized for requiring considerable prior knowledge, and for being unwieldy in large, complex domains. However, modern ASP solvers support efficient reasoning in large knowledge bases with incomplete knowledge, and are used by an international research community (Erdem, Gelfond, and Leone 2016; Erdem and Patoglu 2018).

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. Σ includes *sorts* arranged hierarchically; *statics*, i.e., domain attributes that do not change over time; and *fluents*, i.e., domain attributes whose values can be changed. In the AR domain, sorts include $object$, $robot$, $size$, $relation$, $surface$, and $step$ for temporal reasoning. Statics include some object attributes such as *obj_size(object, size)* and *obj_surface(obj, surface)*. The fluents *obj_relation(relation, object, object)* model relations between objects in terms of their arguments' sorts, e.g., *obj_relation(above, A, B)* implies object A is *above* object B—the last argument in these relations is the reference object. Since the robot in the AR domain also plans and executes actions, the set of fluents also includes relations describing other aspects of the domain, e.g., *in_hand(robot, object)*. Actions of the AR domain include *pickup(robot, object)* and *putdown(robot, object, location)*, and *holds(fluent, step)* is a predicate implying that a particular fluent holds true at a particular time step.

Given the sorted signature, the domain's transition diagram is first described using statements in action language $\mathcal{AL}_d$ (Gelfond and Inclezan 2013); these statements are then translated to statements in ASP. For simplicity, we directly describe the axioms as ASP statements; for the AR domain, this includes ASP statements such as:

$$holds(in\_hand(robot, object), I+1) \leftarrow occurs(pickup(robot, object), I) \quad (1a)$$

$$holds(obj\_relation(above, A, B), I) \leftarrow holds(obj\_relation(below, B, A), I) \quad (1b)$$

$$\neg occurs(pickup(robot, object), I) \leftarrow holds(in\_hand(robot, object), I) \quad (1c)$$

where Statement 3(a) is a causal law, 3(b) is a state constraint, and 3(c) is an executability condition. The spatial relations extracted from images (more detail below) are also converted to ASP statements. In addition, we include axioms that encode generic (commonsense) knowledge about the domain, e.g., statements of the form "larger objects placed on smaller objects are typically unstable".

$$\neg holds(stable(A), I) \leftarrow holds(obj\_relation(above, A, B), I), size(A, large),\ size(B, small),\ not\ holds(stable(A), I) \quad (2)$$

Finally history $\mathcal{H}$ includes records of observations received and actions executed by the robot at particular time steps. We expand it to represent initial state defaults, i.e., statements that are initially assumed to be true in all but a few exceptional circumstances. For instance, we encode "a book is usually in the library; if not there, it is usually found in the office", and exceptions, e.g., cookbooks are in the kitchen.

To reason with the incomplete domain knowledge, the robot automatically constructs the CR-Prolog program $\Pi(\mathcal{D}, \mathcal{H})$. Planning, diagnostics and inference tasks can then be reduced to computing *answer sets* of Π, with each answer describing a possible world comprising beliefs of the robot associated with Π. We use the SPARC system (Balai, Gelfond, and Zhang 2013) to compute answer set(s) of ASP programs, and extract literals corresponding to plans as needed.

In other work we have combined such ASP-based non-monotonic logical reasoning with probabilistic reasoning for more precise action execution at a finer granularity. For ease of understanding and to focus on the interplay between reasoning, learning, and explainability, we limit ourselves to logical reasoning at a coarser resolution in this paper.

Features Extraction and Learning

Next we describe the extraction of features from input images and their use in performing scene understanding tasks.

Feature extraction: The main input to our architecture are RGB images of simulated scenes, e.g., Figure 1, with different object configurations. For any such image, we first extract the spatial relations between scene objects using an existing approach for incrementally revising the grounding (i.e., meaning in physical world) for spatial relations represented by prepositional words such as "above", "behind", and "in" (Mota and Sridharan 2018). In addition, we also extract the other desired object attributes such as color, shape, and size; these are based on probabilistic algorithms, with the most likely outcome encoded as ASP statements associated with complete certainty.

Classification and Learning: The classification block comprises three sub-components, and encodes a processing strategy. For any given image, the agent first attempts to address the classification task (e.g.,, estimate object occlusion and stability of object structures) using ASP-based reasoning with domain knowledge. If an answer is not found, or an incorrect answer is found (during training), the robot automatically extracts relevant regions of interest (ROIs) from the corresponding image. Information from these ROIs is used to train and use a deep (convolutional) neural network, the second sub-component, for the classification task.

Images that need to be processed using deep networks are considered to contain information that is missing (or incorrect) in the existing knowledge. Image features and spatial relations extracted from the ROIs selected from each such image, along with the ground truth label for occlusion and stability, are used by the third sub-component to incrementally learn a decision tree (during training) that summarizes the corresponding experiences of state transitions. Branches in this decision tree that have sufficient support among the training examples are used to induce axioms that are merged with the existing ASP program and used for reasoning.

Answering Explanatory Questions

Next, we describe the components that provide explanatory descriptions of decisions and beliefs.

Text and audio interface: To answer explanatory questions, the robot first needs to interpret the questions correctly. A human's verbal input (of such a question) is processed using existing software. Specifically, verbal input is transcribed using speech recognition software (Zhang 2017), labeled using a part-of-speech tagger, and normalized with the lemma list (Someya 1998) and their synonyms and antonyms retrieved from WordNet (Miller 1995). The processed text helps identify the type of request, which may be the execution of a task or an explanation. In the former case, the related goal is passed to the ASP program for planning. In the latter case, the "Program Analyzer" module (described below) automatically infers and extracts the relevant literals to compose a suitable answer. These literals are placed in generic templates for sentences, resulting in human-understandable (textual) explanations. If needed, the response is converted to synthetic speech (Bhat 2018).

Beliefs tracing: A key ability of the "Program Analyzer" (described below), which constructs the relational (explanatory) descriptions, is to infer the sequence of axioms that explain the evolution of any given belief. Our approach adapts existing methods for generating "proof trees" (Ferrand, Lessaint, and Tessier 2006) to our non-monotonic (logical) reasoning formulation. For any given belief of interest, which could be a positive or negative literal of a fluent or an action, we proceed as follows:

1. Select axioms whose head matches the belief of interest.
2. Ground the literals in the body of each selected axiom and check whether these are supported by the answer set.
3. Create a new branch in a proof tree (with target belief as root) for each selected axiom supported by the answer set, and store the axiom and the related supporting ground literals in suitable nodes.
4. Repeats Steps 1-3 with the supporting ground literals in Step 3 as target beliefs in Step 1, until all branches reach a leaf node with no further supporting axioms.

The paths from the root to the leaves in these proof trees help construct the desired explanations.

Program Analyzer: We describe the approach for automatically identifying and reasoning with the relevant information to construct relational descriptions for four types of *explanatory* questions or requests.The first three were introduced as question types to be considered by any explainable planning system (Fox, Long, and Magazzeni 2017); we also consider a question about the robot's beliefs at any point in time.

1. **Plan description** When asked to describe a plan, the robot parses the related answer set(s) and extract a sequence of actions such as *occurs(action1, step1), ..., occurs(actionN, stepN)* to construct the response.

2. **Action justification: Why action X at step I?** To justify the execution of an 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. For the goal of picking up the red_block in Figure 3, assume that the executed actions are *occurs(pickup(robot, green_mug), 0)*, *occurs(putdown(robot, green_mug, table), 1)*, and *occurs(pickup(robot, red_block), 2)*. If the focus is on the first $pickup$ action, an executability condition related to the second $pickup$ action:

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

is ground in the scene to obtain *obj_rel(below, red_block, green_mug)* as a literal of interest.

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

(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. The question "Why did you pick up the green mug at time step 0?", receives the answer "I had to pick up the red block, and the red block was below the green mug".

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

3. **Hypothetical actions: Why not action X at step I?** For questions about actions not selected:

(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.

(b) For each such executability condition, the robot checks if literals in the body are satisfied by the corresponding answer set. If yes, these literals form the answer.

Suppose action *putdown(robot, green_mug, table)* occurred at step 1 in Figure 3. For the question "Why did you not put the green mug on the yellow duck at time step 1?", the following 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 states that the yellow duck has an irregular surface and the robot answers "Because the yellow duck has an irregular surface". This process uses the *belief tracing* approach.

4. **Belief query: Why belief Y at step I?** To explain any particular belief, the robot uses the *belief tracing* approach described earlier. The supporting axioms and relevant literals identified 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. Tracing this belief identifies the axiom:

$$holds(small\_base(ob_1), I) \leftarrow holds(relation(below,\ ob_2,\ ob_1),\ I),\ has\_size(ob_2,\ small),\ has\_size(ob_1,\ big)$$

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".

Disambiguation Queries

Questions or requests posed by humans may be ambiguous in terms of the objects and time that they reference. Here we describe a method for construction of disambiguation questions that try to address such ambiguity. Consider a number of attributes that characterize the objects in a scene, and a robot able to identify part these features. Different disambiguation queries can be formed by combining such features. In our approach, which is inspired by findings in psychology and cognitive science (Friedman 1974; Read and Marcus-Newhall 1993), the robot constructs the query most likely to address the ambiguity based on three heuristic measures applied in the following sequence:

1. **Unambiguity:** this measure selects attributes that match with a minimum number of ambiguous objects in the context of the query and scene under consideration.
2. **Human confusion:** based on the understanding that queries with many attributes are more likely to confuse a human, this measure is biased towards selecting questions with the minimum number of features in it.
3. **Attribute/feature rank:** this measure seeks to select candidate questions comprising more "useful" attributes. This measure is a linear combination of **human preference** and the **detection complexity** of each feature, which are determined by the robot's domain interactions and algorithms, and is calculated as follows:

$$\textbf{Feature rank} = \alpha \times (human\ preference) + \beta \times (detection\ complexity).$$

where the values of α and β are dynamically updated to reflect the relative importance assigned to the two measures. Here, **human preference** expresses the degree of predilection humans towards using certain attributes for describing certain objects, whereas the **detection complexity** reflects the level of difficulty a robot has in detecting the specific feature. These are domain-specific measures whose values are determined from statistics collected during an initial semi-supervised training phase. For instance, suppose a robot is able to detect **color**, **size** and **shape** of objects, and the current optimal values for α and β are 0.6 and 0.4, respectively. Illustrative examples of the values of **human preference** and **detection complexity**, computed experimentally, and the resulting **rank**, are summarized in Table 1. Although we use the values computed in the training phase for our experiments, these can be revised over time based on the robot's experiences. values of this metrics were fixed for the experiments. We are considering to incrementally update them based on the agent's experiences.

Table 1: Example of computing ranks of attributes.

features	human preference	detection complexity	rank
Color	0.5	0.9	0.66
Size	0.3	0.8	0.5
Shape	0.2	0.6	0.36

A crude method for constructing disambiguating queries would consider all possible combinations of attributes (not included in the human input) to construct candidate queries. It would apply the three measures, and stop when only one candidate query remains or all three measures have been applied. Such an approach would construct and consider a

large number of queries in a complex domain. To address this problem, our architecture introduces a notion of relevance (different from that used to identify ROIs for deep learning) to identify and use contextual knowledge to construct relevant queries. To do so, the robot uses the belief tracing algorithm (Section) to identify information that can be used to address the current ambiguity.

As described earlier, any human query or request is translated to literals compatible with the information in the knowledge base using the *text and audio interface* and the *program analyzer* components. For ease of understanding, assume that the human query maps to a single literal; this is grounded for as many each entity that matches the query in the current scene. The negation of such literals are used as the initial beliefs in the beliefs tracing algorithm. The negated literals not supported by the knowledge base receive higher attention. For instance, in the scene depicted in Figure 3, suppose the human request is "Put the green mug on the top of the yellow object". Since there are three yellow objects in the scene, the request is ambiguous. The following negated action literals are then used as input to the beliefs tracing algorithm:

$$\neg occurs(putdown(rob1, mug, yellow\_duck), I)$$
$$\neg occurs(putdown(rob1, mug, yellow\_cylinder), I)$$
$$\neg occurs(putdown(rob1, mug, yellow\_block), I)$$

The first two literals are supported by the knowledge base, i.e., the robot knows these actions cannot be executed in the current state given the axioms encoded by the human. So the robot prioritizes the yellow cube as being the object of interest and the disambiguation question is biased towards confirming this intuition, with the candidate query being: "Do you want the mug on top of the yellow block?". Section includes an example of the use of this algorithm.

Experimental Setup and Results

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

Experimental Setup

The reasoning and learning capabilities of our baselines architecture have been described in our priir work (Mota and Sridharan 2019; Mota, Sridharan, and Leonardis 2020). Here we focus on belief tracing and generation of disambiguation queries, and evaluate the following hypotheses:

H1 : The proposed disambiguation approach reduces the number of queries posed (i.e., interactions initiated) by the robot and the features used in the queries, and increases the accuracy of the robot's explanatory responses after the first disambiguation question.; and

H2 : The contextual information retrieved by belief tracing enables the robot to construct queries better suited to address the ambiguity in the human query or request.

Experimental trials considered simulated images of the AR domain. We used a real-time physics engine (Bullet) to create 200 simulated images, each with $7-15$ objects (stacked or spread on a flat surface). Objects included cylinders, spheres, cubes/blocks, a duck, and five household objects from the Yale-CMU-Berkeley dataset (apple, pitcher, mustard bottle, mug, and box of crackers). We considered questions containing $2-10$ ambiguous objects, with $2-10$ attributes/features available for disambiguation. One hundred images containing up to 10 objects were used for questions containing up to six ambiguous entities whereas the other 100 were used for question with more than six ambiguous entities. We registered the number of features and interactions required, and the accuracy in the robot's responses after posing the disambiguation queries, for each ambiguous question. We compared the proposed algorithm with the baseline algorithm that randomly and incrementally selects features for disambiguation until there in no more ambiguity or no more features available to use in a disambiguation query. The baseline initially uses the same number of features as the proposed algorithm, and then adds one feature at a time. Each extra feature explored in a disambiguation query is considered as an additional interaction.

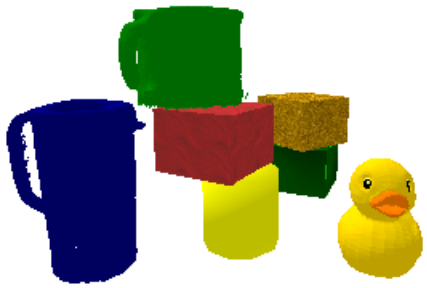

Figure 3: Example a a simulated scene used in the experimental evaluation.

Execution Trace

The following execution traces illustrate our architecture's ability to construct and use disambiguation queries, and to provide relational descriptions in response to the human queries and requests.

Execution Example 1 *[Disambiguation example 1]*
Consider the scenario shown in Figure 3, and assume that objects are characterized by color, shape, and size. A human may pose the following request to the robot:

- **Human**: "Please pick up the yellow object."
  This is an ambiguous request because it is unclear which yellow object the human is referring to.
- The baseline disambiguation strategy would randomly choose and use one of the two unused attributes to ask a follow up question. This could take the form of:
  Robot: "What is the size of the yellow object?"
  In this case, the three yellow objects are of comparable size (medium), so the robot would need at least one more question for disambiguation.

- As stated earlier, our approach uses the three measures to choose the best attributes to construct the queries. Assume that the algorithm considers all possible combinations of the two unused features (size and shape) to construct candidate disambiguation queries, i.e., it considers size, shape, and size and shape respectively.
- Using the **unambiguity** measure, the robot chooses attribute(s) resulting in the least number of matching entities. Since yellow objects are of a similar size (medium), no candidate query is constructed based just on size.
- Based on the **human confusion** measure, the robot seeks to construct queries based on the minimum number of attributes. In our example, the candidate query containing only the shape attribute is preferred over the other combining size and shape. As a result, only one disambiguation question is constructed:
  Robot: "What is the shape of the yellow object?"
- Note that only two measures were used for constructing a disambiguation query in this example. However, when two or more queries are constructed in more complex situations, the third measure **Attribute/feature rank** measure will be applied to select the most useful candidate query to be posed to the human.

Execution Example 2 *[Disambiguation and Axioms trace]* Continuing with the previous example, assume that the human now requests:

- **Human**: "Please move mug on top of the yellow object."
  This request is also ambiguous because similar to Execution Example 1, the robot is unsure which of the three yellow objects the human is referring to.
- Unlike Execution Example 1, we now consider the existing axioms in the ASP program to provide contextual information that reduces the ambiguity and the search space during the construction of the disambiguation queries.
- Assume that that robot knows the following axioms:

$$\neg holds(stable(Ob1), I) \leftarrow \quad (3a)$$
$$holds(obj\_relation(above, Ob_1, Ob_2), I),$$
$$has\_surface(Ob_2, irregular)$$
$$\neg occurs(putdown(rob_1, Ob_1, Obj2), I) \leftarrow \quad (3b)$$
$$holds(obj\_relation(below, Ob_2, Ob_3), I)$$

  Statement 3(a) eliminates the duck as a possible place for the mug since it is known to have an irregular surface. This reduces the number of ambiguous entities to two. Statement 3(b) favors the yellow block (on top of the green block) as the possible supporting place for the mug.
- It is possible to place the mug on top of the yellow cylinder after removing the red block, but the yellow block offer a simpler solution (for the unambiguity measure) and is thus preferred. The following disambiguation query is thus constructed:
  Robot: "Should I move mug on top of the yellow block?"

Execution Example 3 *[Disambiguation and Explanation]* For the same scenario above, consider now that the human wants the robot to move the yellow cylinder to the top of the green cube. The human may request:

- **Human**: "Please move the yellow object on top of the green cube."
  This request is ambiguous because similar to Execution Example 1, the robot is unsure which of the three yellow objects the human is referring to. As the yellow cube is already in the desired position, and the yellow cylinder is below other objects, the yellow duck would be the simpler solution for the ambiguity. As a result, the robot could formulate the following disambiguating query:
  Robot: "Should I move the yellow duck on top of the green cube?"
  Human: "No. Please move the yellow cylinder on top of the green cube."
- To attend to this request, the robot may compute and execute the plan: *pick up the mug; put down the mug on the table; pick up the red cube; put down the red cube on the table; pick up the yellow cube; put down the yellow cube on the table; pick up the yellow cylinder; put down the yellow cylinder on the top of the green cube*.
- The human may now require an explanation related to such a plan:
  Human: "Why did you put down the cube on the table?"
  This is an ambiguous question because the robot has moved the red and yellow cubes to the table in different time steps. Since these two cubes have similar size and shape, the disambiguation algorithm would use color for disambiguating as follows:
  Robot: "What is the color of the cube?"
  Human: "Yellow."
- The explanation approach then provides the following answer to the initial question that is no longer ambiguous:
  Robot: "I had to put the yellow cylinder on top of the green cube. The green cube was below the yellow cube.

Experimental Results

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

1. A hundred initial object configurations were constructed randomly (similar to that in Figure 3). The information extracted from each such image (e.g., object attributes, spatial relations) was encoded in the corresponding ASP program as the initial state.
2. For each initial state, we considered questions in which $2-10$ objects were ambiguous, and $2-10$ attributes were available for the construction of disambiguation queries.
3. The total number of attributes used for disambiguation was registered for the baseline algorithm and for the proposed algorithm. When a sufficient number of attributes were not available for disambiguation, the number of attributes used was considered to be the same as the number of attributes available.
4. We ran the baseline for the same 100 scenes mentioned above, and considered any extra feature needed in addition to the number of features required by the disambiguation algorithm as an extra interaction.

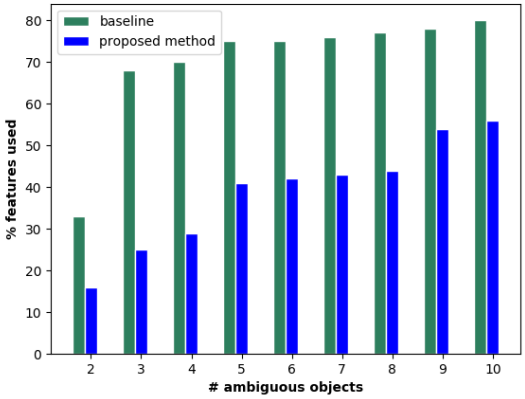

Figure 4: Percentage of features used in disambiguation queries over a number of objects.

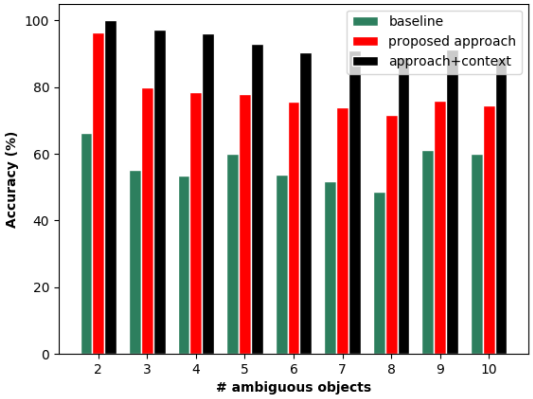

Figure 6: Accuracy in disambiguating for the agent using the baseline, the proposed algorithm with and without contextual knowledge.

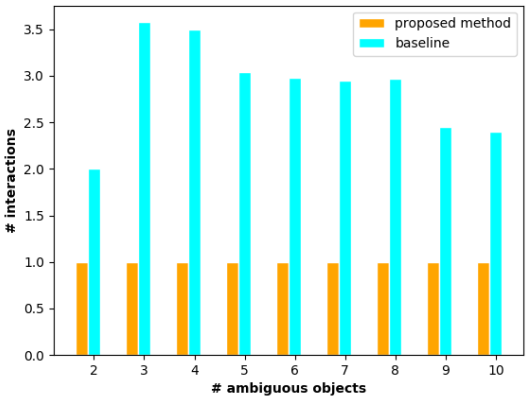

Figure 5: Average number of interactions required by the baseline for disambiguation.

The average values of the measures as a function of the number of ambiguous objects, are shown in Figure 4. The average number of interactions as a function of the number of ambiguous objects is plotted in Figure 5. Figure 4 indicates that using the proposed method reduces the number of features required for disambiguation. Figure 5 shows that the baseline approach requires at least two interactions to achieve the expected response whereas the proposed method requires only one. These results support **H1**.

The second set of experiments was designed as follows to evaluate hypotheses **H1** and **H2**:

1. A hundred initial object configurations were constructed randomly (similar to that in Figure 3). The information extracted from each such image (e.g., object attributes, spatial relations) was encoded in the corresponding ASP program as the initial state.
2. For each initial state, we considered questions in which $2-10$ objects were ambiguous, and $2-10$ attributes were available for the construction of disambiguation queries.
3. The accuracy for the answers provided by the robot after asking the disambiguation question is computed for the baseline methods, and the proposed algorithm with and without contextual knowledge. The results plotted in Figure 6 as *baseline*, *proposed approach*, and *approach+context*, respectively.

Figure 6 indicates that the use of the proposed algorithm improves accuracy of the potential responses provided, which provides further support to **H1**. The figure also shows that the related information extract from the knowledge base helps improve accuracy of human responses to disambiguating queries, which supports **H2**.

Conclusion

The architecture described in this paper is a step towards explainable reasoning and learning for integrated robot systems that include methods for learning from data and reasoning with incomplete commonsense domain knowledge. The architecture supports non-monotonic logical reasoning, data-driven deep learning from a limited set of examples, and inductive learning of previously unknown axioms governing domain dynamics. We also described a simple interactive strategy that traces beliefs, and constructs and poses suitable disambiguation queries to result in more accurate relational descriptions of decisions, beliefs, and hypothetical events. Future work will further explore the interplay between reasoning and learning in the context of explaining decisions and beliefs in more complex domains.

References

Anjomshoae, S.; Najjar, A.; Calvaresi, D.; and Framling, K. 2019. Explainable agents and robots: Results from a systematic literature review. In *International Conference on Autonomous Agents and Multiagent Systems*. Montreal, Canada.

Assaf, R.; and Schumann, A. 2019. Explainable Deep Neural Networks for Multivariate Time Series Predictions. In *International Joint Conference on Artificial Intelligence*, 6488–6490. Macao, China.

Balai, E.; Gelfond, M.; and Zhang, Y. 2013. Towards Answer Set Programming with Sorts. In *International Conference on Logic Programming and Nonmonotonic Reasoning*.

Bercher, P.; Biundo, S.; Geier, T.; Hoernle, T.; Nothdurft, F.; Richter, F.; and Schattenberg, B. 2014. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Bhat, N. M. 2018. pyttsx3 - Text-to-speech x-platform.

Borgo, R.; Cashmore, M.; and Magazzeni, D. 2018. Towards Providing Explanations for AI Planner Decisions. In *IJCAI Workshop on Explainable Artificial Intelligence*, 11–17.

Cakmak, M.; and Thomaz, A. L. 2012. Designing robot learners that ask good questions. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 17–24. IEEE.

de Kleer, J.; and Williams, B. C. 1987. Diagnosing Multiple Faults. *Artificial Intelligence* 32: 97–130.

Erdem, E.; Gelfond, M.; and Leone, N. 2016. Applications of Answer Set Programming. *AI Magazine* 37(3): 53–68.

Erdem, E.; and Patoglu, V. 2018. Applications of ASP in Robotics. *Kunstliche Intelligenz* 32(2-3): 143–149.

Fandinno, J.; and Schulz, C. 2019. Answering the "Why" in Answer Set Programming: A Survey of Explanation Approaches. *Theory and Practice of Logic Programming* 19(2): 114–203.

Ferrand, G.; Lessaint, W.; and Tessier, A. 2006. Explanations and Proof Trees. *Computing and Informatics* 25: 1001–1021.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *IJCAI Workshop on Explainable AI*.

Friedman, M. 1974. Explanation and scientific understanding. *Philosophy* 71(1): 5–19.

Gelfond, M.; and Inclezan, D. 2013. Some Properties of System Descriptions of AL_d. *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming* 23(1-2): 105–120.

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

Gervasio, M.; Yeh, E.; and Myers, K. 2011. Learning to ask the right questions to help a learner learn. In *Proceedings of the 16th international conference on Intelligent user interfaces*, 135–144.

Gonzalez-Pacheco, V.; Malfaz, M.; Castro-Gonzalez, A.; Castillo, J. C.; Alonso, F.; and Salichs, M. A. 2018. Analyzing the impact of different feature queries in active learning for social robots. *International Journal of Social Robotics* 10(2): 251–264.

Koh, P. W.; and Liang, P. 2017. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*, 1885–1894.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. The MIT Press.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11): 39–41.

Miller, T. 2019. Explanations in Artificial Intelligence: Insights from the Social Sciences. *Artificial Intelligence* 267: 1–38.

Mota, T.; and Sridharan, M. 2018. Incrementally Grounding Expressions for Spatial Relations between Objects. In *International Joint Conference on Artificial Intelligence*, 1928–1934.

Mota, T.; and Sridharan, M. 2019. Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots. In *Robotics Science and Systems*.

Mota, T.; Sridharan, M.; and Leonardis, A. 2020. Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics. In *European Conference on Multiagent Systems*.

Myagmarjav, B.; and Sridharan, M. 2015. Incremental knowledge acquisition for human-robot collaboration. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 809–814. IEEE.

Read, S. J.; and Marcus-Newhall, A. 1993. Explanatory coherence in social explanations: A parallel distributed processing account. *Personality and Social Psychology* 65(3): 429.

Ribeiro, M.; Singh, S.; and Guestrin, C. 2016. Why Should I Trust You? Explaining the Predictions of Any Classifier. In *International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Rosenthal, S.; Veloso, M.; and Dey, A. K. 2012. Acquiring accurate human responses to robots questions. *International journal of social robotics* 4(2): 117–129.

Samek, W.; Wiegand, T.; and Mller, K.-R. 2017. 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.

Someya, Y. 1998. Lemma List for English Language.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. B. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Neural Information Processing Systems*. Montreal, Canada.

Zhang, A. 2017. Speech Recognition (version 3.8).

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan explicability and predictability for robot task planning. In *International Conference on Robotics and Automation*, 1313–1320.