SN Computer Science (2021) 2:242
https://doi.org/10.1007/s42979-021-00573-0

ORIGINAL RESEARCH

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

Tiago Mota[1] · Mohan Sridharan[2] · Aleš Leonardis[2]

Received: 14 September 2020 / Accepted: 8 March 2021 / Published online: 29 April 2021


## Abstract

A robot's ability to provide explanatory descriptions of its decisions and beliefs promotes effective collaboration with humans. Providing the desired transparency in decision making is challenging in integrated robot systems that include knowledge-based reasoning methods and data-driven learning methods. As a step towards addressing this challenge, our architecture combines 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, the evolution of associated 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, preconditions, and effects of actions, 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

## 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 occluded if the view of any minimal fraction of its frontal face is hidden by another object, and an object 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. It also incrementally learns and revises previously unknown constraints, and preconditions and effects of actions, and responds to questions about its plans, actions, decisions, and beliefs. For instance, assume that the goal in Fig. 1b is to have the yellow ball on the orange block, and that the plan is to move the blue block to the table's surface before placing the ball on the orange block. When asked about a plan step, e.g., "why do you want to pick up the blue block first?", the robot answers "I have to put the ball on the orange block, and the blue block is on the orange block"; when asked, after plan execution, "why did you not pick up the pig?", the robot responds "Because the pig is not related to the goal".

The motivating scenario described above poses key knowledge representation, reasoning, learning, and control challenges. In this paper, we focus on enabling a robot to provide on-demand explanations of its decisions and beliefs

This article is part of the topical collection "Advances in Multi-Agent Systems Research: EUMAS 2020 Extended Selected Papers" guest edited by Nick Bassiliades and Georgios Chalkiadakis.

✉ Mohan Sridharan
m.sridharan@bham.ac.uk

Tiago Mota
tmot987@aucklanduni.ac.nz

Aleš Leonardis
a.leonardis@bham.ac.uk

[1] Electrical and Computer Engineering, The University of Auckland, Auckland, New Zealand

[2] School of Computer Science, University of Birmingham, Birmingham, United Kingdom

**(a)** Test scenario.

**(b)** Image from robot's camera.

**Fig. 1** **a** Motivating scenario of a Baxter robot arranging objects on a tabletop; **b** image from the camera on the robot's left gripper

in the form of descriptions comprising relations between relevant domain objects, object attributes, actions, and robot attributes. Such "explainability" will help human designers improve the underlying algorithms and establish accountability. Providing these explanations is particularly challenging in integrated robot systems that combine knowledge-based reasoning methods (e.g., for planning) and data-driven learning methods (e.g., for pattern recognition). Inspired by research in cognitive systems that highlights the superior capabilities provided by coupling different representations, reasoning methods, and learning methods [20, 43], our architecture provides transparency in decision making by integrating the principles of data-driven learning and knowledge-driven reasoning. Building on our prior work that combined non-monotonic logical reasoning and deep learning for classification tasks in simulated images [31], our architecture enables a robot to:

- Automatically learn axioms encoding previously unknown state constraints, and action preconditions and effects;
- Automatically trace the evolution of any given belief or the non-selection of any given action at a given time by inferring the relevant sequence of axioms and beliefs; and
- Exploit the interplay between representation, reasoning, and learning to describe decisions and beliefs related to computed or executed plans and hypothetical situations.

Our recent conference paper provided proof of concept evidence of our architecture's ability to learn previously unknown constraints and extract relevant information to construct descriptions of decisions and beliefs [33]. Here, we describe these capabilities in more detail, and introduce the ability to acquire action preconditions and effects and trace the evolution of beliefs. These capabilities are evaluated in the context of performing planning tasks and scene understanding tasks in simulated scenes and on a physical robot manipulating tabletop objects. Specifically, the robot: (1) computes and executes plans to arrange objects in desired configurations; and (2) estimates occlusion of scene objects and stability of object configurations. Experimental results indicate the ability to (1) incrementally reduce uncertainty in the scene by learning previously unknown state constraints, and preconditions and effects of actions; and (2) construct explanations reliably and efficiently by automatically identifying and reasoning with the relevant knowledge despite noisy sensing and actuation.

The remainder of this paper is organized as follows. "Related Work" discusses related work to motivate the architecture described in "Architecture". Experimental results and conclusions are discussed in "Experimental Setup and Results" and "Conclusions", respectively.

## Related Work

Early work in explanation generation drew on research in cognition, psychology, and linguistics to characterize explanations in terms of factors such as generality, objectivity, connectivity, relevance, and information content [11]. Subsequent studies involving human subjects have also indicated that the attributes of good explanations include coherence, simplicity, generality, soundness, and completeness [36]. In parallel, fundamental computational methods were developed for explaining unexpected outcomes by reasoning logically about potential causes [16].

With the increasing use of AI and machine learning methods in different domains, there is renewed interest in understanding the decisions of these methods[1]. This understanding can be used to improve the underlying algorithms, and to

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

make automated decision-making more acceptable or trustworthy to humans [25]. Recent work in *explainable AI* and *explainable planning* can be broadly categorized into two groups [29]. Methods in one group modify or map learned models or reasoning systems to make their decisions more interpretable, e.g., by mapping decisions to input data [17], explaining the predictions of classifiers by learning equivalent interpretable models [37], or biasing a planning system towards making decisions easier for humans to understand [45]. Methods in the other group provide descriptions that make a reasoning system's decisions more transparent, e.g., describing planning decisions [5], combining reasoning based on classical first order logic with interface design to help humans understand a plan [4, 26], describing why a particular solution was obtained for a given problem using non-monotonic logical reasoning [8], or using rules made of monotonic operators to define *proof trees* that provide a declarative view (i.e., explanation) of the trace of a computation [9]. Researchers have also explored explanations for non-monotonic rule-based systems in semantic web applications [2, 18]. Much of this research is agnostic to how an explanation is structured or assumes comprehensive domain knowledge. Also, they do not explore the interplay between learning, representation, and reasoning to improve the quality of the explanations.

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 [3]. As documented in a recent survey, these methods compute gradients and decompositions in a network's layers to obtain heatmaps of the relevant features [38]. 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 [35, 44]. However, these approaches do not (1) fully integrate reasoning and learning to inform and guide each other; or (2) 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.

There is a well-established literature methods in AI for learning logic-based representations of domain knowledge. Examples include the incremental revision of a first-order logic representation of action operators [14], the inductive learning of domain knowledge represented as an Answer Set Prolog program [15, 23], and work in our group on coupling of non-monotonic logical reasoning, inductive learning, and relational reinforcement learning to incrementally acquire actions and axioms [33, 41]. Our approach for learning domain axioms is inspired by work in interactive task learning, a general framework for acquiring domain knowledge using labeled examples or reinforcement signals

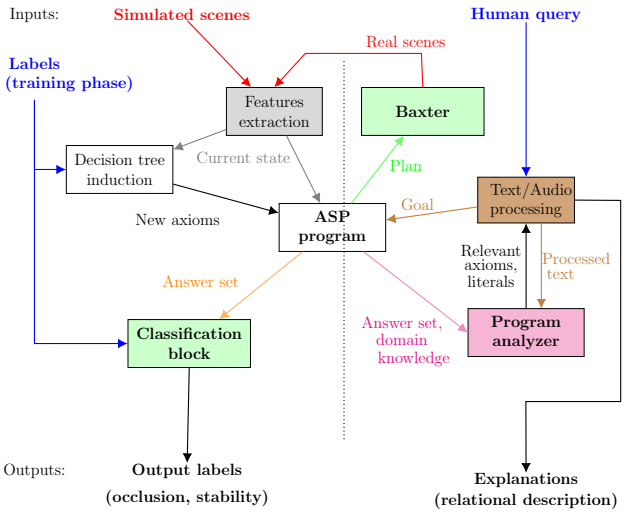



**Fig. 2** Architecture combines strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, deep learning, and inductive learning. Components to the left of the dashed vertical line establish this combination, and those to the right of the dashed line support the desired explainability

obtained from domain observations, demonstrations, or human instructions [6, 21]. However, unlike methods that learn from many training examples, our approach learns from limited training examples.

In this paper, we focus on integrated robot systems that use knowledge-based methods and data-driven methods to represent, reason with, and learn from incomplete domain knowledge and noisy observations. We enable such robots to generate relational descriptions of decisions and beliefs, including hypothetical or counterfactual situations that are often used by humans and computer systems to infer causal relations [27]. Recent surveys indicate that these capabilities are not supported by existing systems [1, 29]. Our architecture builds on knowledge representation tools, our prior work on integrating non-monotonic logical reasoning and deep learning for classification tasks in simulated images [31], and work in our group on explainable agency [22] and a theory of explanations [42] that has shown that the non-monotonic logical reasoning paradigm used in this paper can be used to present information at the level of abstraction, verbosity, and specificity desired by the human participant [42].

## Architecture

Figure 2 shows the overall architecture. Components to the left of the dashed vertical line combine non-monotonic logical reasoning and deep learning for classification tasks; an initial version of these components were described in a context of classification tasks in simulated images in

our conference paper [31]. Components to the right of the dashed line expand reasoning capabilities and answer questions about decisions and beliefs before, during, or after reasoning and learning. An initial version of some of these components were introduced in our recent conference paper [33]. Here, we describe all components, focusing primarily on the tracing of beliefs and construction of explanations, and highlighting recent changes and extensions in other components. We do so using the following illustrative domain.

***Example Domain 1*** [Robot Assistant (RA) Domain] A Baxter (see Fig. 1a): (1) estimates occlusion of scene objects and stability of object structures, and arranges objects in desired configurations; and (2) 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); position and distance-based spatial relations between objects (above, below, front, behind, right, left, in); other 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 other objects blocking its frontal face causes this object to be not occluded.
- An object that is positioned below another object cannot be picked up.

This knowledge may need to be revised, e.g., some actions, axioms, and the values of some attributes may be unknown, or the robot may observe that placing certain objects on an object with an irregular surface results in a stable configuration.

## Knowledge Representation, Reasoning, and Learning

We first describe the knowledge representation, reasoning, and learning capabilities, i.e., the components to the left of the dashed vertical line in Fig. 2.

*Feature extraction* In our architecture, the sensor inputs are RGB images of simulated scenes, or noisy top and front views of scenes from the robot's cameras; our prior work also considered RGB-D images of simple simulated scenes [31]. From each image, the "*Feature extraction*" component in Fig. 2 uses a probabilistic algorithm to extract objects and their attributes. Also, the spatial relations between objects in the image are computed using our prior work that learned the *grounding*, i.e., the meaning in the physical world, for seven position-based prepositional words (*in, above, below, front, behind, right, left*) and three distance-based prepositional words (*touching, non-touching, far*). This grounding is modeled in the form of 2D and 1D histograms, which are learned from labeled image data and revised over time based on human feedback. Given an input image, a measure of similarity computed between the histograms extracted from this image and the learned models is used to label the spatial relations between pairs of objects in the image. For more details about this grounding, please see [30].

*Non-monotonic logical reasoning* To represent and reason with domain knowledge, the "*ASP program*" component in Fig. 2 uses 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 "¬ a", which implies that "*a is believed to be false*", "not a" only implies that "*a is not believed to be true*" [13]. 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 inferred consequences, which helps recover from errors caused by reasoning with incomplete domain knowledge. This is an appealing capability for robotics domains characterized by incomplete knowledge, dynamic changes, and noisy observations. ASP and other knowledge-based reasoning paradigms are often criticized for requiring comprehensive prior knowledge, and for being unwieldy in large, complex domains. However, ASP has been used by an international research community to reason with incomplete domain knowledge in many applications, and modern ASP solvers have demonstrated the ability to reason efficiently with a large knowledge base [7].

A domain's description in ASP comprises a *system description* $\mathcal{D}$ and a *history* $\mathcal{H}$. $\mathcal{D}$ comprises a *sorted signature* $\Sigma$ and axioms encoding the domain's dynamics. In [31] we explored spatial relations between objects in the image for classification tasks; the $\Sigma$ included *basic sorts*, e.g., *object*, *robot*, *location size*, *relation*, and *surface*; *statics*, i.e., domain attributes that do not change over time, e.g., $\text{obj}_{\text{size}}$(object, size) for object size and $\text{obj}_{\text{surface}}$(obj, surface) for object surface; and *fluents*, i.e., domain attributes whose values can be changed, e.g., *loc*(*object*, *location*) implies that a particular object is at a particular location, and *obj_relation*(*above*, *object*, *object*) implies that a particular

object is above another particular object. In [33] and this paper, *the robot also plans and executes actions that cause changes in the domain*. We model the corresponding domain dynamics by first describing the expanded $\Sigma$ and transition diagram in action language $\mathcal{AL}_d$ [12] and then automatically translating this description to ASP statements. Action languages are formal models of parts of natural language used for specifying transition diagrams of dynamic domains. $\mathcal{AL}_d$ supports three types of statements: causal law, state constraint, and executability condition, which are encoded as:

$a$ **causes** $l_{in}$ **if** $p_0, \ldots, p_m$
$l$ **if** $p_0, \ldots, p_m$
**impossible** $a_0, \ldots, a_k$ **if** $p_0, \ldots, p_m$

where $a$ is an action, $l$ is a literal, $l_{in}$ is an inertial literal, and $p_0, \ldots, p_m$ are domain literals. For the RA domain, $\Sigma$ now also includes the sort *step* for temporal reasoning, fluents such as $\text{in}_{\text{hand}}$(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 time step.

Given a signature, axioms in the system description capture the domain's dynamics. For the RA domain, the axioms would include $\mathcal{AL}_d$ statements such as:

$$\text{putdown}(\text{rob}_1, \text{Ob}_1, \text{Ob}_2) \textbf{ causes } \text{obj\_relation}(\text{on}, \text{Ob}_1, \text{Ob}_2) \tag{1a}$$

$$\text{obj\_relation}(\text{above}, \text{Ob}_1, \text{Ob}_2) \textbf{ if } \text{obj\_relation}(\text{below}, \text{Ob}_2, \text{Ob}_1) \tag{1b}$$

$$\textbf{impossible } \text{pickup}(\text{rob}_1, \text{Ob}_1) \textbf{ if } \text{obj\_relation}(\text{below}, \text{Ob}_1, \text{Ob}_2) \tag{1c}$$

where Statement 1(a) is a causal law implying that putting an object down on another object causes the first object to be on the second one; Statement 1(b) is a state constraint linking the spatial relations *above* and *below* between two objects; and Statement 1(c) is an executability condition implying that the robot cannot try to pick up an object that is below another object. The domain axioms also encode constraints that hold unless there is evidence to the contrary, e.g., "larger objects placed on smaller objects are unstable unless stated otherwise" is encoded as:

$$\neg\text{stable}(A) \textbf{ if } \text{obj\_relation}(\text{above}, A, B), \text{size}(A, \text{large}), \text{size}(B, \text{small}), \text{not stable}(A) \tag{2}$$

where "not" denotes default negation. In addition to these axioms, information extracted from the input images (e.g., spatial relations, object attributes) with sufficiently high probability is converted to ASP statements.

The $\mathcal{H}$ of a dynamic domain typically comprises records of the form *obs*(*fluent*, *boolean*, *step*), i.e., fluents observed to be true or false at a particular time step, and *hpd*(*action*, *step*), i.e., an action's execution at a particular time step. In [40], other work in our group expanded this notion 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 domain knowledge, our architecture constructs the CR-Prolog program $\Pi(\mathcal{D}, \mathcal{H})$, which includes $\Sigma$ and axioms of $\mathcal{D}$, inertia axioms, reality checks, closed world assumptions for actions, and observations, actions, and defaults from $\mathcal{H}$. For instance, Statements 1(a–c) would be translated into ASP statements such as:

$$\text{holds}(\text{obj\_relation}(\text{on}, \text{Ob}_1, \text{Ob}_2), I+1) \leftarrow \text{occurs}(\text{putdown}(\text{rob}_1, \text{Ob}_1, \text{Ob}_2), I) \tag{3a}$$

$$\text{holds}(\text{obj\_relation}(\text{above}, \text{Ob}_1, \text{Ob}_2), I) \leftarrow \text{holds}(\text{obj\_relation}(\text{below}, \text{Ob}_2, \text{Ob}_1), I) \tag{3b}$$

$$\neg\text{occurs}(\text{pickup}(\text{rob}_1, \text{Ob}_1), I) \leftarrow \text{holds}(\text{obj\_relation}(\text{below}, \text{Ob}_1, \text{Ob}_2), I) \tag{3c}$$

In addition, 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 ASP statement:

$$\neg\text{loc}(X, \text{library}) \xleftarrow{+} \text{book}(X) \tag{4}$$

is a CR rule that is only triggered under exceptional circumstances to assume a book is not in the library, e.g., as an explanation for an unexpected observation of a book outside the library. CR rules can also be used for diagnostics, i.e., to explore the reasons for any unexpected outcomes and to trigger the learning and revision of axioms. We do not discuss it here to avoid confusion with our axiom induction approach described below; see [13, 41] for complete details.

Once $\Pi$ is constructed, planning, diagnostics, and inference can be reduced to computing *answer sets* of $\Pi$ after introducing some helper relations and axioms [13]. Any answer set represents the beliefs of the robot associated with $\Pi$; 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 domain *state* at that time step. The program for our RA domain is available in the "*Explanations*" folder of our open-source online repository [34].

Note that incorrect inferences can be drawn due to incomplete knowledge, noisy sensor input, or when probabilistic information is elevated 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 heuristic thresholds. Also, although we do not describe it here, our architecture supports the modeling of non-determinism (e.g., in action outcomes). 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 [40]. For ease of understanding and to focus on the interplay between reasoning and learning in the context of constructing explanations, we limit ourselves to logical reasoning at one resolution in this paper.

*Classification* Similar to our prior work [31], the "*Classification block*" in Fig. 2 first tries to estimate the occlusion of objects and the stability of object configurations in any given image using ASP-based reasoning. If an answer is not found, or an incorrect answer is found for 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 [24], AlexNet [19]) 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 automatically identify and ground only the relevant axioms and relations in the image under consideration to determine the ROIs to be analyzed further [31]. In this paper, we build on this notion of relevance and reason over a sequence of steps to provide explanations, as described in "Relational Descriptions as Explanations".

*Axiom induction* Images used to train the CNNs are considered to contain information about missing or incorrect constraints related to occlusion and stability. In the "*Decision tree induction*" component in Fig. 2, 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 with less than a minimal level of support (i.e., $< 5\%$) on a separate set of 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. The thresholds can be revised to achieve other outcomes, e.g., they can be lowered significantly to identify default constraints.

Unlike our prior work [31, 33], we introduce new strategies to process noisy images of more complex scenes. First, we use a homogeneous ensemble learning approach, retaining only axioms that are identified over a number of cycles of applying the decision tree induction approach for learning and validation on different subsets of data. Second, different versions of the same axiom are merged to remove over-specifications. As an example, consider the statements:

$$\neg\text{stable}(A) \leftarrow \text{obj\_relation}(\text{above}, A, B), \text{obj\_surface}(B, \text{irregular}) \tag{5a}$$

$$\neg\text{stable}(A) \leftarrow \text{obj\_relation}(\text{above}, A, B), \text{obj\_surface}(B, \text{irregular}), \text{obj\_size}(B, \text{large}) \tag{5b}$$

where Statement 5(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 at least one common literal 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. Each such ASP program is used to classify ten labeled scenes, retaining the 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 eventually removed.

Unlike our prior work that only learned state constraints [33], the robot now also learns previously unknown causal laws and executability conditions if there is any mismatch between the expected and observed state after an action is executed. Any expected but unobserved fluent literal indicates missing executability condition(s), and any observed unexpected fluent literal indicates missing causal law(s). Given the distributed representation of axioms in our architecture, axioms of any particular type (e.g., causal law) are learned by constructing decision trees in a suitable format for each candidate action. The learning is based on the following methodology:

1. To explore missing executability conditions, the robot simulates the execution of the action (that caused the inconsistency) in different initial states and stores: (a) the relevant information from the initial state; (b) the executed action; and (c) a label indicating the presence

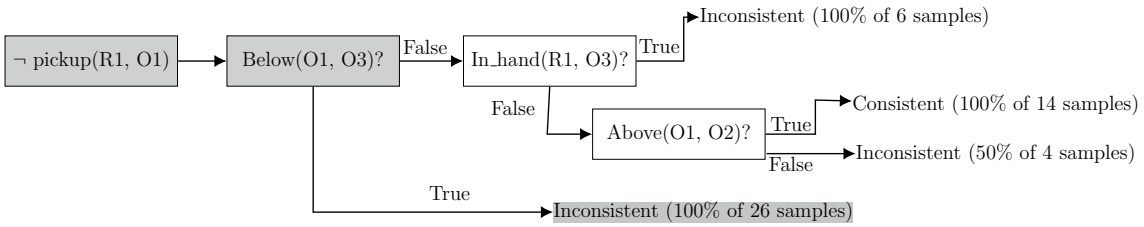



**Fig. 3** Example illustrating part of the decision tree created for the missing executability condition encoded by Statement 3(c)

or absence of inconsistency, which is used as the output label for the sample in the decision tree. Any fluent literal that exists in the answer set or initial state, and has an object constant that occurs in the action under consideration, is considered to be relevant; it is stored with variables replacing ground terms. These simulations correspond to the execution of the action in some initial state based on the model provided by the ASP program.

2. To explore a missing causal law, training samples are collected as in Step 1, but the output label is the observed, unexpected fluent literal from the resultant state.
3. Separate decision trees are created with the relevant information from the initial state as the attributes (i.e., nodes); the output label is the presence or the absence of inconsistency for any executability conditions, and the unexpected fluent for any causal law. The root node of the tree is the executed action.

Figure 3 shows part of a decision tree obtained using the method described above when the executability condition in Statement 3(c) is missing; the learned axiom is along the path shaded grey. The label *consistent* (*inconsistent*) implies that for $\geq 50\%$ of the examples at the leaf, observations match (do not match) the expected outcomes. It is possible for the robot to observe a transition to an unexpected state due to hardware problems (e.g., mechanical failure), but such failures are assumed to be infrequent and are considered to be exceptions that do not trigger or influence the axiom learning approach. When such failures occur, the robot is expected to create a new plan to achieve the goal once the failure has been fixed.

## Relational Descriptions as Explanations

The components of our architecture on the right side of the dashed line in Fig. 2 exploit the interplay between representation, reasoning, and learning to provide the desired on-demand relational descriptions of decisions and beliefs.

*Interaction interface and control loop* Human interaction with our architecture through speech or text input is handled by the "*Text/audio processing*" component in Fig. 2. Existing software implementations and a controlled (domain-specific) vocabulary are used to parse human verbal input. 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 [39] and related synonyms and antonyms from WordNet [28]. The processed text helps identify the type of request, which may correspond to a desired goal or a question/command about decisions and beliefs. In the former case, the goal is sent to the ASP program for planning. The robot computes and executes the plan, replanning when unexpected action outcomes cannot be explained, until the desired goal is achieved or learning is triggered. In the latter case, i.e., when given a question about the decisions and beliefs, the "Program Analyzer" considers the current knowledge (including inferred beliefs) and parsed 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.

*Tracing beliefs/axioms* A key capability of our architecture is to infer the sequence of axioms and beliefs that explain the evolution of any given belief or the non-selection of any given ground action at a given time. This tracing of beliefs and axioms is done by the "*Program Analyzer*" component in Fig. 2, which uses the inferred sequence of axioms and beliefs for building explanations. Others have constructed such *proof trees* in the context of monotonic (i.e., classical first order) logic statements used to explain observations [9]. We adapt this method to our domain representation based on non-monotonic logic, and use the following methodology to explain the evolution of any given belief and the non-selection of any given action.

1. Select axioms whose head matches the belief or action of interest.

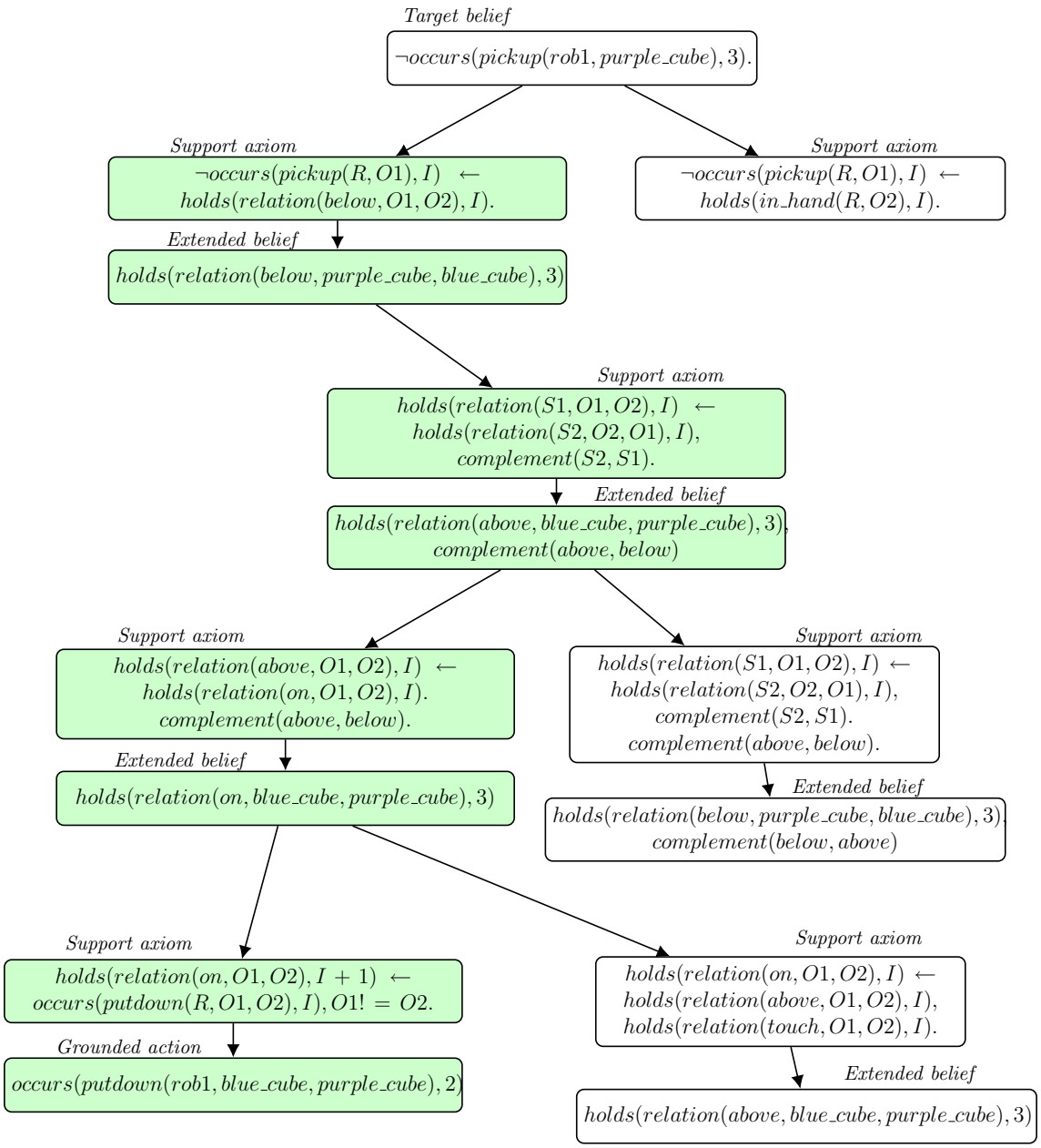



**Fig. 4** Example of belief tracing to explain non-selection of a particular action

2. Ground the literals in the body of each selected axiom and check whether these are supported by the answer set under consideration.
3. Create a new branch in a proof tree (with the target belief or action as the root node) 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 without further supporting axioms.

The paths from the root to the leaves in these proof trees help construct the desired explanations. If multiple such supporting branches exist, they are not compared with each other for choosing the one that best explains a target belief. Instead, the algorithm randomly selects any branch to compose the required answers, leaving a comparison of the available options as a direction for future work. As an example, consider the initial scenario in Fig. 1b with the goal of having the yellow ball on the orange cube. The robot computes a plan that has in it move the blue cube on top of the purple cube (behind the pig) and then move the

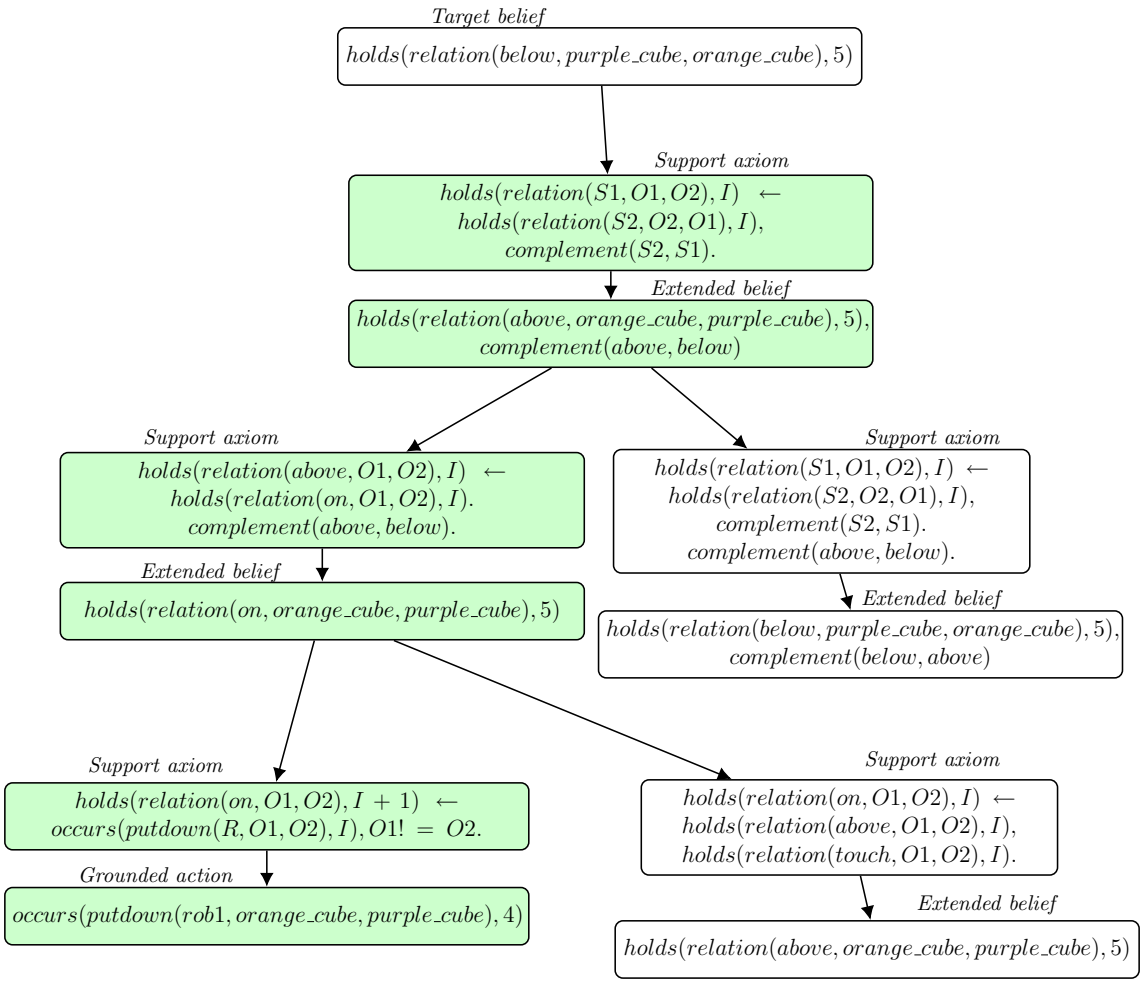



**Fig. 5** Example of belief tracing to explain a particular belief

ball on top of the orange cube. If the robot is asked (after plan execution) why it did not pick up the purple block at time step 3, part of the corresponding proof tree would be as shown in Fig. 4; the path highlighted in green contains the information needed for the answer; in this example, the purple cube could not be picked up because the blue cube had just been placed on top of it. As another example consider the goal of having the pig on the red cube. The plan is to move the blue cube to the table and then the orange cube on top of the purple cube before moving the pig on top of the red cube. Now, if the robot is asked to explain why it believed that the purple cube was below the orange cube at time step 5, part of the corresponding proof tree would be as shown in Fig. 5, with the path highlighted in green providing the information needed to construct the answer.

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

**Input :** Question literal; $\Pi(\mathcal{D}, \mathcal{H})$; answer templates.
**Output:** Answer to questions.

```
// Compute answer set
1  AS = AnswerSet(Π)
2  if question = plan description then
       // Retrieve 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 actions
6      relevant_axioms = Retrieve(Π, head = ¬next_actions)
       // Extract relevant literals
7      relevant_literals = Retrieve(AS, Body(relevant_axioms) ∈ I ∧ ∉ 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(Π, head = ¬occurs(X))
       // Extract relevant literals
11     answer_literals = Retrieve(AS, Body(relevant_axioms) ∈ I ∧ ∉ I + 1)
12 else if question = "why belief Y at step I?" then
       // Extract relevant axioms
13     relevant_axioms = Retrieve(Π, head = Y)
       // Examine body of axioms
14     answer_literals = Recursive_Examine(AS, Body(relevant_axioms))
15 Construct_Answer(answer_literals, answer_templates)
```

*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. It does so in the context of four types of *explanatory* requests or questions. The first three were introduced in prior work as questions to be considered by any explainable planning system [10]; in addition, we consider a question about the robot's beliefs at any given point in time.

1. *Plan description* When asked to describe a particular plan, the robot parses the related answer set(s) to extract a sequence of actions occurs(action$_1$, step$_1$), …, occurs(action$_N$, step$_N$) (line 3, 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, the robot considers the actions or states that this action enables, and proceeds as follows:

   (a) For each action that occurred after step $I$, the robot examines relevant axioms (e.g., executability conditions, causal laws) and identifies literal(s) that would prevent this action's execution (lines 5–7, Algorithm 1). For example, assume that for the goal of placing the orange_block on the table in Fig. 1b, the following plan is executed:

   occurs(pickup(robot, blue_block), 0),
   occurs(putdown(robot, blue_block, table), 1),
   occurs(pickup(robot, orange_block), 2).

   If asked to justify the execution of the first *pickup* action in this plan, i.e., if posed the question "Why did you pick up the blue block at time step 0?", one of the relevant axioms is the following executability condition related to the second *pickup* action in the plan (line 6):

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

   which is ground in the image or scene under consideration to obtain obj_relation(below, orange_block, blue_block) as a literal of interest in this example.

   (b) If any such 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 time step, it is taken to be a reason for executing the action under consideration (line 7).

   (c) The identified reason is paired with the subsequent action to construct the answer to the question (line 8). In the current example, the robot's answer includes "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 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 a particular action not included in the plan and not considered for execution at a particular time step, the robot proceeds as follows to identify conditions that would have prevented it from considering the action:

   (a) The robot identifies executability conditions with the hypothetical action in the head, i.e., conditions that would 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.

In the plan described above for the goal of having the orange block on the table in Fig. 1b, action putdown(robot, blue_block, table) occurred at step 1. For the question "Why did you not put the blue cube on the tennis ball at time step 1?", the following executability condition is identified as being relevant:

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

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". Note that to answer to this question, the robot has to use the approach described earlier to trace a sequence of related axioms and beliefs.

4. *Belief query: Why belief Y at step I?* To explain any particular belief held at a particular time step, the robot uses the *belief tracing* approach described earlier to trace the sequence of axioms that were activated. These supporting axioms and relevant literals are used to construct the answer.

For instance, if the robot is asked the question "why do you believe object $ob_1$ is unstable at step I?", it finds the following support axiom:

$$\neg\text{holds}(\text{stable}(\text{ob}_1), I)\ \leftarrow\ \text{holds}(\text{small\_base}(\text{ob}_1), I)$$

Assume that the robot's beliefs includes a statement about $\text{ob}_1$ having a small base. Searching for why $\text{ob}_1$ is believed to have a small base identifies the following relevant axiom:

$$\text{holds}(\text{small\_base}(\text{ob}_1), I)\ \leftarrow \\ \text{holds}(\text{relation}(\text{below},\ \text{ob}_2,\ \text{ob}_1),\ I), \\ \text{has\_size}(\text{ob}_2,\ \text{small}),\ \text{has\_size}(\text{ob}_1,\ \text{big})$$

As a result, the robot provides the following answer "Because object $\text{ob}_2$ is below object $\text{ob}_1$, $\text{ob}_2$ is small, and $\text{ob}_1$ is big".

*Robot platform* As stated earlier, in addition to images of simulated scenes, this paper considers a physical robot that plans and executes actions to achieve the desired goals. For the robot experiments, we use a Baxter robot that manipulates objects on a tabletop; this is the "*Baxter*" component in Fig. 2. 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 Fig. 1b. The Baxter also uses built-in probabilistic motion planning algorithms to execute primitive manipulation actions, e.g., to grasp, pick up, and move objects. Observations obtained with a high probability (e.g., $\geq 0.9$) are elevated to literals associated with complete certainty in the ASP program.

Recall that non-monotonic logical reasoning enables the robot to identify and recover from errors caused by incomplete or incorrect information. For instance, consider the situation in which robot $\text{rob}_1$ has been asked to move book $\text{book}_2$ from the library to the *office*. Since the sensor in the robot's arm is unreliable in detecting when an object is in its grip, the value of the fluent $\text{in\_hand}(\text{rob}_1, \text{book}_2)$ is unknown after the robot has executed $\text{pickup}(\text{rob}_1, \text{book}_2)$ and $\text{book}_2$ is actually in the robot's hand. This outcome does not match the expected outcome and would create an inconsistency. Even if the sensor in the robot's arm provides an incorrect observation when the robot is about to put $\text{book}_2$ down in the *office*, it would be prevented from doing so by the following executability condition:

$$\neg\text{occurs}(\text{putdown}(\text{robot}, \text{object}, \text{location}), I)\ \leftarrow \\ \text{not holds}(\text{in\_hand}(\text{robot}, \text{object}), I)$$

which uses default negation (i.e., *not*) in the body to encode a stronger constraint than the use of classical negation (i.e., ¬); it implies that it is impossible for the robot to put a particular object down in a particular location if it does not know whether the object is in its hand or not, and not just when it is sure that it is not in its hand. In such situations, the robot can perform diagnostics to figure out the cause for the observed inconsistency by reasoning or executing actions to gather more information.

## Experimental Setup and Results

In this section, we discuss the results of evaluating our architecture's ability to reason with incomplete knowledge, learn previously unknown axioms, and construct relational descriptions of decisions and beliefs. Specifically, "Experimental Setup" describes the setup for different experiments, "Execution Trace" describes some execution traces, and "Experimental Results" discusses quantitative results.

### Experimental Setup

We evaluated the following hypotheses:

- ***H1*** : our architecture enables the robot to accurately learn previously unknown axioms;
- ***H2*** : reasoning with incrementally learned axioms improves the quality of plans generated;
- ***H3*** : the beliefs tracing approach accurately retrieves the supporting axioms and beliefs; and
- ***H4*** : exploiting the links between reasoning and learning improves the accuracy of the explanatory descriptions.

These hypotheses and the underlying capabilities were evaluated considering the four types of explanatory requests and questions described earlier: (1) describing the plan; (2) justifying the execution of an action at a given time step; (3) justifying not choosing an action at a given time step; and (4) justifying any given belief. As stated earlier, the same methodology can also be adapted to address other types of requests and 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 (i.e., "ground truth") response 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 Fig. 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. For real scenarios, the states were measured using the robot's cameras before and after the execution of its actions. 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.

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 providing the robot knowledge of some domain constraints. The robot equipped with our architecture learned and revised the missing constraints over time as described in "Knowledge Representation, Reasoning, and Learning", whereas the missing 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 fails to achieve the desired goal.

To test hypothesis **H1** we removed five axioms (three executability conditions and two causal laws) from the agent's knowledge, and ran the learning algorithm 20 times. One of these axioms is the executability condition encoded by Statement 3c, and the rest by the following statements:

$$\neg occurs(pickup(rob_1, Ob_1), I) \leftarrow holds(in\_hand(rob_1, Ob_2), I) \tag{6a}$$

$$\neg occurs(putdown(rob_1, Ob_1, Ob_2), I) \leftarrow \neg holds(in\_hand(rob_1, Ob_1), I) \tag{6b}$$

$$holds(in\_hand(rob_1, Ob_1), I+1) \leftarrow occurs(pickup(rob_1, Ob_1), I) \tag{6c}$$

$$\neg holds(in\_hand(rob_1, Ob_1), I+1) \leftarrow occurs(putdown(rob_1, Ob_1, Ob_2), I) \tag{6d}$$

The robot executed actions to learn all the missing axioms each time. Each run was terminated if the robot executed a number of actions without detecting any inconsistency, or if a maximum number of decision trees were constructed. The overall precision and recall of learning the missing axioms were then computed. Before we describe the details

of other experiments and discuss the quantitative results of these experimental trials, we first describe some execution traces illustrating our architecture's operation.

## Execution Trace

The following execution traces illustrate our architecture's ability to construct relational descriptions of its decisions and beliefs during reasoning and learning.

**Execution Example 1** [Plans, actions, and beliefs] Consider a robot that starts with objects as shown in Fig. 1b. The robot is assigned the goal of achieving a state in which the red cube is on top of the orange cube, i.e., holds(relation(on, $\text{red}_{\text{cube}}$, $\text{orange}_{\text{cube}}$), I). The following interaction takes place *after* the robot has executed a plan and successfully achieved the assigned goal.

- *Human* "Please describe the plan."
- *Baxter* "I picked up the blue cube. I put the blue cube on the table. I picked up the orange cube. I put the orange cube on the table. I picked up the red cube. I put the red cube on the orange cube."
- The human may ask the robot to justify a particular action in the executed plan.
- *Human* "Why did you pick up the blue cube at step 0?"
- *Baxter* "Because I had to pick up the red cube, and it was below the blue cube."

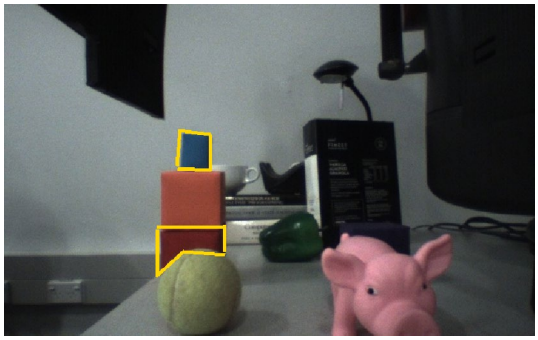
  The answer constructed in response to the question is also used to automatically highlight the relevant image regions that influenced this answer, as illustrated in Fig. 6a.
- The human now may ask about particular actions that were not considered.

  *Human* "Why did you not put down the orange cube on the blue cube?"

  *Baxter* "Because the blue cube is small." In the absence of any reference to a particular time step, the robot answers the question based on the single instance (in the executed plan) of putting the orange cube on another cube or surface. The answer is also based on learned default knowledge that any large object with a small base, i.e., when placed on a small object, is typically unstable.
- The human may also ask the robot to justify particular beliefs.

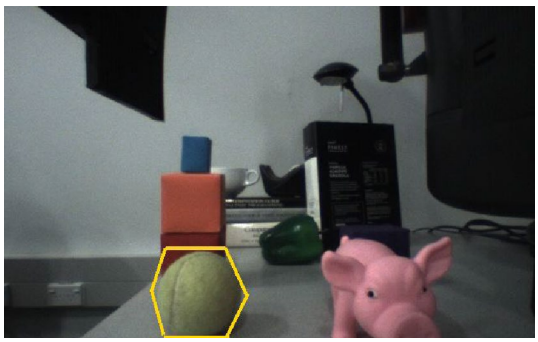
  *Human* "Why did you believe that the red cube was below the blue cube in the initial state?"

  *Baxter* "Because I observed the red cube below the blue cube in step zero."



**(a)** Execution Example 1.

**(b)** Execution Example 3.



**(c)** Additional example.

**Fig. 6** **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 a trial (not described in this paper) in which a tennis ball plays an important role in the explanation constructed

**Execution Example 2** [Reasoning and explanation generation] 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_cube), $I$) is encoded in the ASP program for planning. The default axiom about configurations with small bases being unstable acts as a constraint that prevents the robot from placing objects on the blue cube. At the same time, the robot has to compute plans to achieve the assigned goal. This

causes an inconsistency that is resolved by invoking the corresponding CR rule and planning to place the ball on the blue cube. The following interaction takes place after this plan is executed.

– *Human* "Please describe the plan that you executed."
  *Robot* "I picked up the ball. I put the ball on the blue cube."
– The human may now explore the robot's belief related to the exception to default knowledge that the robot had to invoke:
  *Human* "Why do you believe that the ball is on the blue cube?"
  *Robot* "Because I observed the ball on the blue cube in step 2."

Combining reasoning with the approach for 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 execute the human request because a learned constraint makes it impossible to achieve the desired object configuration or belief. Even in such cases, our architecture enables the robot to answer questions about the decisions. For instance, consider the simulated scene in Fig. 6b, with the following interaction:

– *Human* "Please put the pitcher on the duck."
  This action is not executed because a constraint learned earlier implies that any object configuration with an object on another with an irregular surface is unstable.
– The robot can justify 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(s) relevant to the construction of the robot's answer to the human query is (are) automatically highlighted in the corresponding image, as illustrated by Fig. 6b for the current example. This example also illustrates how integrating reasoning and learning helps justify the decision to not execute a requested action because it will have an unfavorable outcome.

Continuing with the scenario illustrated in the Fig. 6b, the robot is now asked to move the duck on top of the red cube. A possible plan to achieve this goal would be: pick up the green cylinder, put it on the table, pick up the white cube, put it on the top of the green cylinder, pick up the duck, and put it down on the top of the red cube. Considering that each action is executed in one time step, this plan contains six time steps. Consider the following interaction after the execution of such a plan:

– *Human:* "Why did you not pick up green cylinder at step 5?"
  Since this question is about a hypothetical action not actually executed by the robot, it explores the related scenario by creating a proof tree, as described in "Relational Descriptions as Explanations", and provides the following answer:
  *Robot* "Because the white cube was on the green cylinder."
  The human may ask for further details:
  *Human* "Why did you believe the white cube was on the green cylinder?"
  To answer this question the robot has to know the causal relationship between the action *putdown* and the spatial relation *on*—see Statement 3(a). Since the robot has learned this causal law, it constructs the correct answer:
  *Robot* "Because I put the white cube on the green cylinder at time step 4."

This example illustrates the benefit of exploiting the interplay between reasoning and learning to justify particular beliefs.

**Execution Example 4** [Belief tracing and explanation generation] We continue with our previous example:

- *Human* "Why did you not pick up the white cube at step 0?"
  The robot uses belief tracing to construct a proof tree with the relation ¬occurs(pickup(rob1, white_cube), 0) as the root. For each axiom in which this ground literal matches the head, it checks if its body is supported by the answer set. If yes, ground literals in the body are used to expand the tree. Based on the axiom encoded by Statement 3(c), one of the beliefs identified as being relevant is holds(obj_rel(below, white_cube, green_cylinder), 0). These steps are repeated until no further supporting axioms are found. The ground literal holds(relation(on, white_cube, green_cylinder), 0) is output as the leaf of the proof tree, and the robot answers the query.
- *Robot* "Because I observed the green cylinder on the white cube at step 0."

Overall, these examples illustrate the ability to focus on relevant knowledge, incrementally learn and revise axioms, trace relevant beliefs, and identify attributes and actions relevant to a given scenario. They also support hypothesis H3. Since the same samples are used to learn axioms and train the deep networks, our approach also helps understand the behavior of the deep networks.

**Table 1** Precision and recall of learning previously unknown axioms using decision tree induction, as described in "Relational Descriptions as Explanations"

| Missing axioms | Precision (%) | Recall (%) |
|---|---|---|
| Strict | 69.2 | 78.3 |
| Relaxed | 96 | 95.1 |

## Experimental Results

The first set of experiments evaluated H1. We removed five axioms (two causal laws and three executability conditions, as described above) from the robot's knowledge, and ran the learning algorithm 20 times. We measured the precision and recall of learning the missing axioms in each run, and Table 1 summarizes the results. The row labeled "Strict" provides results when any variation in the target axiom is considered an error. In this case, even over-specified axioms, i.e., axioms that have some additional irrelevant literals, are considered to be incorrect. The following is an example axiom in which the second literal in the body is irrelevant.

$$\begin{aligned} \neg\text{holds}(\text{in\_hand}(R1, O1), I+1) \leftarrow \\ \text{occurs}(\text{putdown}(R1, O1, O2), I), \\ \neg\text{holds}(\text{in\_hand}(R1, O5), I). \end{aligned} \tag{7}$$

The row labeled "Relaxed" reports results when over-specifications are not considered errors; the high precision and recall support H1.

The second set of experiments was designed to evaluate hypothesis **H2**.

1. As stated earlier, 40 initial object configurations were created. The Baxter automatically extracted information (e.g., attributes, spatial relations) from images corresponding to top and frontal views (i.e., images from 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 chosen randomly. The robot reasoned with the existing knowledge to create plans for these 200 combinations (40 initial states, five goals).
3. The computed plans were evaluated to determine the number of optimal, sub-optimal, and incorrect plans, and planning time.
4. Trials were repeated with and without including the learned axioms for reasoning.

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 axioms being included in the ASP program used for reasoning. The initial conditions and goal were identical in each paired trial, but differed between paired trials. Then, we expressed the number of plans and the planning time with the learned axioms included for reasoning as a fraction of the corresponding values obtained by not using the learned axioms for reasoning. The average of these fractions over all trials is reported in Table 2. We also computed the number of optimal, sub-optimal, and incorrect plans in each trial as a fraction of the total number of plans; this too was done with and without using the learned axioms for reasoning, and the average over all trials is summarized in Table 3.

**Table 2** Number of plans and planning time with the learned axioms used for reasoning expressed as a fraction of the values without using the learned axioms for reasoning

| Measures | Ratio (with/without) | |
|---|---|---|
| | Real scenes | Simulated scenes |
| Number of steps | 1.15 | 1.23 |
| Number of plans | 0.81 | 1.08 |
| Planning time | 0.96 | 1.02 |

**Table 3** 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

| Plans | Real Scenes | | Simulated scenes | |
|---|---|---|---|---|
| | Without | With | Without | With |
| Optimal | 0.4 | 0.9 | 0.14 | 0.3 |
| Sub-optimal | 0.11 | 0.1 | 0.46 | 0.7 |
| Incorrect | 0.49 | 0 | 0.4 | 0 |

These results indicate that for images of real scenes, using the learned axioms for reasoning significantly reduced the search space, resulting in a much smaller number of plans and a reduction in the planning time. The use of the learned axioms does not seem to make any significant difference with the simulated scenes. This is understandable because simulated images have more objects with several of them being small objects. This increases the number of possible plans to achieve any given goal. In addition, when the robot used the learned axioms for reasoning, it reduced the number of sub-optimal plans and eliminated all incorrect plans. Also, almost every sub-optimal plan was created when the corresponding goal could not be achieved without creating an exception to a default. Without the learned axioms, a larger fraction of the plans are sub-optimal or incorrect. Note that the number of sub-optimal plans is higher for simulated scenes that have more objects to consider. These results support

**Table 4** (*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

| Query type | Precision | | Recall | |
|---|---|---|---|---|
| | Without (%) | With (%) | Without (%) | With (%) |
| Plan description | 78.54 | 10 | 67.52 | 100 |
| Why X? | 76.29 | 95.25 | 66.75 | 95.25 |
| Why not X? | 96.61 | 96.55 | 64.04 | 100 |
| Belief | 96.67 | 99.02 | 95.6 | 100 |

**Table 5** (*Simulated scenes*) Precision and recall of retrieving relevant literals for constructing answers to questions with and without reasoning with the learned axioms. Learned axioms significantly improve the accuracy of the explanations

| Query type | Precision | | Recall | |
|---|---|---|---|---|
| | Without (%) | With (%) | Without (%) | With (%) |
| Plan description | 70.78 | 100 | 57.98 | 100 |
| Why X? | 65.63 | 93.0 | 57.75 | 93.0 |
| Why not X? | 90.53 | 96.38 | 65.15 | 100 |
| Belief | 92.73 | 98.44 | 90.27 | 99.21 |

hypothesis H2 but also indicate the need to explore complex scenes further.

The third set of experiments was designed as follows to evaluate hypothesis **H4**:

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 with the expected literals in the "ground truth" response, with the average precision and recall scores shown in Table 4.
4. Similar experiments were performed with simulated images; results are in Table 5.

Tables 4 and 5 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 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. The experimental results summarized above thus indicate the ability of our architecture to provide good performance in the presence of noise in sensing and actuation on physical robots. These results also indicate that reasoning and learning inform and guide each other to provide accurate relational descriptions of decisions, beliefs, and the outcomes of hypothetical actions. Overall, these results provide evidence in support of hypothesis **H4**. For additional examples of images, and experimental results of classification and explanation generation, please see our open source repository [34].

## Conclusions

This paper described an approach inspired by cognitive systems research for an integrated robot system to explain its decisions and beliefs, including the outcomes of hypothetical actions. The explanations are constructed on-demand before, during, or after reasoning or learning, in the form of descriptions of relations between relevant objects, actions, and attributes of the domain. We 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 information relevant to the tasks at hand to efficiently construct the desired explanations. Also, both the planning and explanation generation performance improves significantly when the robot incrementally learns and uses previously unknown axioms for reasoning.

Our architecture opens up multiple avenues for further research. First, we will extend the ability to learn other kinds of axioms and consider actions with delayed rewards.

We will do so by building on the architecture developed by others in our group by combining non-monotonic logical reasoning and relational reinforcement learning [41]. Second, we will explore more complex domains, tasks, and explanations, reasoning with logic-based and probabilistic representations of relevant knowledge at different tightly-coupled resolutions for scalability [40]. 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), and scenarios in which the human user wants the explanation at a different level of abstraction, specificity, or verbosity. We will do so by building on our proof of concept work on disambiguation [32], and work in our group on a related theory of explanations [42]. Third, we will use our architecture to better understand the behavior of deep networks. The key advantage of our architecture is that it uses reasoning to guide learning; unlike "end-to-end" data-driven 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 reliable and efficient operation in complex, dynamic domains.

**Funding** This work was supported in part by the Asian Office of Aerospace Research and Development award FA2386-16-1-4071, the U.S. Office of Naval Research Science of Autonomy Awards N00014-17-1-2434 and N00014-20-1-2390, and the UK Engineering and Physical Sciences Research Council Award EP/S032487/1. Opinions and conclusions reported in this paper are those of the authors alone.

## Declarations

**Conflict of interest** The authors (Tiago Mota, Mohan Sridharan, Aleš Leonardis) do not have any conflict of interest to declare.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any 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. Antoniou G, Bikakis A, Dimaresis N, Genetzakis M, Georgalis G, Governatori G, Karouzaki E, Kazepis N, Kosmadakis D, Kritsotakis M, et al. Proof explanation for a nonmonotonic semantic web rules language. Data Knowl Eng. 2008;64(3):662–87.
3. Assaf R, Schumann A. Explainable deep neural networks for multivariate time series predictions. In: International joint conference on artificial intelligence, Macao, China, pp. 6488–6490. 2019.
4. Bercher P, Biundo S, Geier T, Hoernle T, Nothdurft F, Richter F, Schattenberg B. Plan, repair, execute, explain - how planning helps to assemble your home theater. In: Twenty-fourth international conference on automated planning and scheduling. 2014
5. Borgo R, Cashmore M, Magazzeni D. Towards providing explanations for AI planner decisions. In: IJCAI workshop on explainable artificial intelligence, pp. 11–17. 2018.
6. Chai JY, Gao Q, She L, Yang S, Saba-Sadiya S, Xu G. Language to action: towards interactive task learning with physical agents. In: International joint conference on artificial intelligence. 2018.
7. Erdem E, Patoglu V. Applications of ASP in robotics. Kunstliche Intelligenz. 2018;32(2–3):143–9.
8. Fandinno J, Schulz C. Answering the "Why'' in answer set programming: a survey of explanation approaches. Theory and Practice of Logic Programming. 2019;19(2):114–203.
9. Ferrand G, Lessaint W, Tessier A. Explanations and proof trees. Comput Inform. 2006;25:1001–21.
10. Fox M, Long D, Magazzeni D. Explainable planning. In: IJCAI workshop on explainable AI. 2017.
11. Friedman M. Explanation and scientific understanding. Philosophy. 1974;71(1):5–19.
12. Gelfond M, Inclezan D. Some properties of system descriptions of $AL_d$. J Appl Non Class Logics Spec Issue Equilib Logic Answ Set Programm. 2013;23(1–2):105–20.
13. Gelfond M, Kahl Y. Knowledge representation, reasoning and the design of intelligent agents. Cambridge: Cambridge University Press; 2014.
14. Gil Y. Learning by experimentation: incremental refinement of incomplete planning domains. In: International conference on machine learning, pp. 87–95. 1994.
15. Katzouris N, Artikis A, Paliouras G. Online learning of event definitions. Theory Pract Logic Programm. 2016;16(5–6):817–33.
16. de Kleer J, Williams BC. Diagnosing multiple faults. Artif Intell. 1987;32:97–130.
17. Koh PW, Liang P. Understanding black-box predictions via influence functions. In: International conference on machine learning. pp. 1885–1894. 2017.
18. Kontopoulos E, Bassiliades N, Antoniou G. Visualizing semantic web proofs of defeasible logic in the dr-device system. Knowl Based Syst. 2011;24(3):406–19.
19. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Neural information processing systems, pp. 1097–1105. 2012.
20. Laird JE. The soar cognitive architecture. Cambridge: The MIT Press; 2012.
21. Laird JE, Gluck K, Anderson J, Forbus KD, Jenkins OC, Lebiere C, Salvucci D, Scheutz M, Thomaz A, Trafton G, Wray RE, Mohan S, Kirk JR. Interactive task learning. IEEE Intell Syst. 2017;32(4):6–21.

22. Langley P, Meadows B, Sridharan M, Choi D. Explainable agency for intelligent autonomous systems. In: Innovative applications of artificial intelligence. Cambridge: AAAI Press; 2017.
23. Law M, Russo A, Broda K. The ILASP system for inductive learning of answer set program. Technical report on arXiV. 2020. https://arxiv.org/abs/2005.00904.
24. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
25. Lewandowsky S, Mundy M, Tan G. The dynamics of trust: comparing humans to automation. J Exp Psychol Appl. 2000;6(2):104.
26. McGuinness DL, Glass A, Wolverton M, Da Silva PP. Explaining task processing in cognitive assistants that learn. In: AAAI spring symposium: interaction challenges for intelligent assistants, pp. 80–87. 2007.
27. Menzies P, Beebee H. Counterfactual theories of causation. In: Zalta EN, editor. The Stanford encyclopedia of philosophy. 2020th ed. Stanford: Stanford University; 2020.
28. Miller GA. WordNet: a lexical database for English. Commun ACM. 1995;38(11):39–41.
29. Miller T. Explanations in artificial intelligence: insights from the social sciences. Artif Intell. 2019;267:1–38.
30. Mota T, Sridharan M. Incrementally grounding expressions for spatial relations between objects. In: International joint conference on artificial intelligence, pp. 1928–1934. 2018.
31. Mota T, Sridharan M. Commonsense reasoning and knowledge acquisition to guide deep learning on robots. In: Robotics science and systems. 2019.
32. Mota T, Sridharan M. Answer me this: constructing disambiguation queries for explanation generation in robotics. In: Workshop of the UK planning and scheduling special interest group. 2020.
33. Mota T, Sridharan M. Commonsense reasoning and deep learning for transparent decision making in robotics. In: European conference on multiagent systems. 2020.
34. Mota T, Sridharan M. Scene understanding, reasoning, and explanation generation. 2020. https://github.com/tmot987/Scenes-Understanding
35. Norcliffe-Brown W, Vafeais E, Parisot S. Learning conditioned graph structures for interpretable visual question answering. In: Bengio S, Wallach H, Larochelle H,Grauman K, Cesa-Bianchi N, Garnett R. editors. Advances in neural information processing systems, vol. 31. Montreal, Canada. 2018.
36. Read SJ, Marcus-Newhall A. Explanatory coherence in social explanations: a parallel distributed processing account. Pers Soc Psychol. 1993;65(3):429.
37. 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.
38. Samek W, Wiegand T, Müller KR. Explainable artificial intelligence: understanding, visualizing and interpreting deep learning Models. ITU J ICT Discov Impact Artif Intell Commun Netw Serv. 2017;1:1–10.
39. Someya Y. Lemma list for English language. 1998.
40. Sridharan M, Gelfond M, Zhang S, Wyatt J. REBA: a refinement-based architecture for knowledge representation and reasoning in robotics. J Artif Intell Res. 2019;65:87–180.
41. Sridharan M, Meadows B. Knowledge representation and interactive learning of domain knowledge for human-robot collaboration. Adv Cogn Syst. 2018;7:1–20.
42. Sridharan M, Meadows B. Towards a theory of explanations for human-robot collaboration. Kunstliche Intelligenz. 2019;33(4):331–42.
43. Winston PH, Holmes D. The genesis enterprise: taking artificial intelligence to another level via a computational account of human story understanding. In: Computational models of human intelligence report 1. Cambridge: Massachusetts Institute of Technology; 2018.
44. Yi K, Wu J, Gan C, Torralba A, Kohli P, Tenenbaum JB. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, and Garnett R. editors. Advances in neural information processing systems. Montreal, Canada. 2018.
45. Zhang Y, Sreedharan S, Kulkarni A, Chakraborti T, Zhuo HH, Kambhampati S. Plan explicability and predictability for robot task planning. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R editors. International conference on robotics and automation, vol. 31, pp. 313–1320. 2017.