What Happened and Why?
A Mixed Architecture for Planning and Explanation Generation in Robotics

Zenon Colaco and **Mohan Sridharan**
Department of Electrical and Computer Engineering
The University of Auckland, NZ
zncolaco@gmail.com; m.sridharan@auckland.ac.nz

Abstract

This paper describes a mixed architecture that couples the non-monotonic logical reasoning capabilities of a declarative language with probabilistic belief revision, enabling robots to represent and reason with qualitative and quantitative descriptions of knowledge and uncertainty. Incomplete domain knowledge, including information that holds in all but a few exceptional situations, is represented as a Answer Set Prolog (ASP) program. The answer set obtained by solving this program is used for inference, planning, and for jointly explaining (a) unexpected action outcomes; and (b) partial scene descriptions extracted from sensor input. For any given task, each action in the plan contained in the answer set is executed probabilistically. For each such action, observations extracted from sensor inputs perform incremental Bayesian updates to a probabilistic (belief) distribution over a relevant subset of the domain, committing high probability beliefs as statements to the ASP program. The architecture's capabilities are evaluated in simulation and on a mobile robot in scenarios that mimic a robot waiter assisting in a restaurant.

1 Introduction

Robots collaborating with humans in complex domains may receive raw data from different sensors. The information extracted from these sensor inputs can be represented probabilistically to quantitatively model the associated uncertainty (e.g., "I am 90% certain I saw a book on the table"). Robots also receive useful commonsense knowledge about the domain that is difficult to represent quantitatively, e.g., "Books are usually in the library". In addition, the human participants may not be able to provide elaborate and accurate feedback. To collaborate with humans, robots thus need to represent knowledge and reason at both the cognitive level and the sensorimotor level. This objective maps to fundamental challenges in knowledge representation and reasoning. Towards addressing these challenges, our previous architecture [Zhang *et al.*, 2014] coupled the non-monotonic logical reasoning capabilities of Answer Set Prolog (ASP), a declarative language, with the probabilistic planning and uncertainty modeling capabilities of partially observable Markov decision processes (POMDPs). For any given task, incomplete domain knowledge was represented as an ASP program, which was solved to obtain a plan of abstract actions. Each action in this plan was executed probabilistically using POMDPs, committing high probability beliefs as statements to the ASP program. While retaining the basic principle of coupling between logic programming and probabilistic reasoning, this paper expands the capabilities as follows:

- In addition to providing a plan for any given task, the *answer set* obtained by solving the ASP program is used for *jointly* (a) explaining unexpected action outcomes by reasoning about exogenous actions; and (b) identifying object occurrences that best explain partial scene descriptions extracted from sensor inputs.
- Representation and reasoning at a finer resolution in the ASP program improves the efficiency of probabilistic reasoning while still providing high reliability. Instead of using a POMDP to execute each action in the ASP-based plan, sensor observations perform incremental Bayesian updates to a probability distribution over a subset of the domain relevant to this action.

These capabilities are evaluated in simulation and on a mobile robot, in scenarios that mimic a robot waiter seating people and delivering orders in a restaurant.

The remainder of this paper is organized as follows. The proposed architecture is motivated by reviewing some related work in Section 2, followed by a description of the architecture in Section 3. Section 4 describes the result of experimental evaluation, followed by the conclusions in Section 5.

2 Related Work

Knowledge representation, planning and explanation generation are well-researched areas in robotics and artificial intelligence. Logic-based representations and probabilistic graphical models have been used to control sensing, naviga-

tion and interaction for robots and agents [Bai *et al.*, 2014; Galindo *et al.*, 2008]. Formulations based on probabilistic representations (by themselves) make it difficult to perform commonsense reasoning, while classical planning algorithms and logic programming tend to require considerable prior knowledge of the domain and the agent's capabilities, and make it difficult to merge new, unreliable information with an existing knowledge base. For instance, the non-monotonic logical reasoning capabilities of ASP [Gelfond and Kahl, 2014] have been used by an international research community for tasks such as reasoning by simulated robot housekeepers [Erdem *et al.*, 2012] and coordination of robot teams [Saribatur *et al.*, 2014]. However, ASP does not support probabilistic representation of uncertainty, whereas a lot of information extracted from sensors and actuators on robots is represented probabilistically.

Explanation generation systems (e.g., through abductive inference or plan diagnosis) use the system description and observations of system behavior to explain unexpected symptoms [Balduccini and Gelfond, 2003; Reiter, 1987], or use weaker system descriptions and depend on heuristic representation of intuition and past experience [Meadows *et al.*, 2013; Ng and Mooney, 1992]. Probabilistic and first-order logic representations have also been combined for abductive inference [Sindhu Raghavan and Raymond J. Mooney, 2011]. Researchers have also designed architectures that combine deterministic and probabilistic algorithms for task and motion planning [Kaelbling and Lozano-Perez, 2013], couple declarative programming and continuous-time planners for path planning in robot teams [Saribatur *et al.*, 2014], or combine a probabilistic extension of ASP with POMDPs for human-robot dialog [Zhang and Stone, 2015]. Recent work used a three-layered organization of knowledge (instance, default and diagnostic), and a three-layered architecture (competence, belief, and deliberative layers), which combines first-order logic and probabilistic reasoning for open world planning on robots [Hanheide *et al.*, 2015]. Some popular formulations that combine logical and probabilistic reasoning include Markov logic network [Richardson and Domingos, 2006], Bayesian logic [Milch *et al.*, 2006], and probabilistic extensions to ASP [Baral *et al.*, 2009; Lee and Wang, 2015]. However, algorithms based on first-order logic do not provide the desired expressiveness, e.g., it is not always possible to express degrees of belief quantitatively. Algorithms based on logic programming do not support one or more of the desired capabilities such as incremental revision of (probabilistic) information; reasoning as in causal Bayesian networks; and reasoning with large probabilistic components. Our prior work developed architectures that couple declarative programming and probabilistic graphical models for logical inference, deterministic and probabilistic planning on robots [Sridharan *et al.*, 2015; Zhang *et al.*, 2014; 2015]. This paper retains the coupling between logical and probabilistic reasoning but expands the existing capabilities

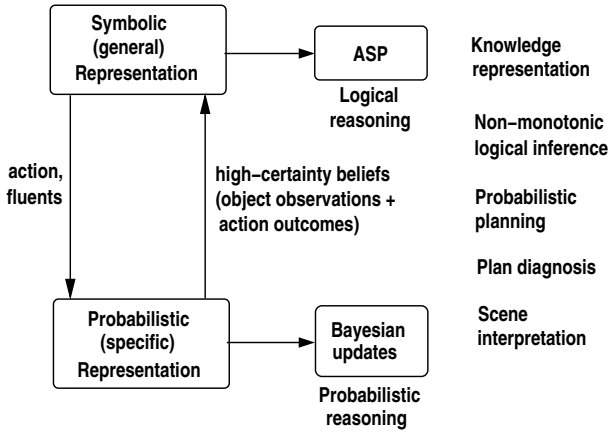


Figure 1: Overview of the mixed architecture that combines the strengths of declarative programming and Bayesian updates for inference, planning, and diagnosis.

to support: (1) joint explanation of unexpected action outcomes and partial descriptions extracted from sensor inputs; and (2) reasoning at a finer resolution using ASP, making the probabilistic reasoning more computationally efficient while still providing high reliability of task completion.

3 Proposed Architecture

Figure 1 is an overview of the mixed architecture. The symbolic (domain) representation is translated to an Answer Set Prolog (ASP) program used for non-monotonic logical inference and planning a sequence of actions for any given task, as described in Section 3.1. For each action in a plan created by inference in the ASP program, the relevant subset of the domain is identified [Sridharan *et al.*, 2015]. Sensor observations perform incremental Bayesian updates to a probability distribution over this domain subset, committing high probability beliefs (regarding action outcomes and observations of object attributes) as statements to the ASP program, as described in Section 3.2. Unexpected action outcomes are explained by reasoning about exogenous actions, and objects are identified to explain the partial descriptions extracted from visual cues. Representation and reasoning in ASP is performed at a resolution that provides high reliability, simplifies the (coupled) probabilistic reasoning and adapts it to the actions at hand. The individual components of the architecture are described below.

The syntax, semantics and representation of the transition diagram of the architecture's domain representation are described in an *action language* AL [Gelfond and Kahl, 2014]. AL has a sorted signature containing three *sorts*: *statics*, *fluents* and *actions*. Statics are domain properties whose truth values cannot be changed by actions, fluents are properties whose values are changed by actions, and actions are elementary actions that can be executed in parallel. AL al-

lows three types of statements:

a **causes** l_b **if** $p_0,\ldots,p_m$ (Causal law)

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

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

where a is an action, l is a literal, l_b is a basic fluent (also called inertial fluent) literal, and $p_0,\ldots,p_m$ are domain literals (any domain property or negation). A collection of statements of AL forms a system description.

Example domain: As an illustrative example used throughout this paper, consider a robot that seats people at tables, and delivers orders, in a restaurant. The domain's sorts are arranged hierarchically, e.g., *location* and *thing* are subsorts of *entity*; *animate* and *inanimate* are subsorts of *thing*; *person* and *robot* are subsorts of *animate*; *object* is a subsort of *inanimate*; and *room*, *area*, *door*, and *floor* are subsorts of *location*. We consider specific rooms in the domain, such as *kitchen* and *dining*; objects of sorts such as *table*, *chair* and *plate*, characterized by attributes *size*, *color*, *shape*, and *location*; and dishes of sorts such as *pasta*, *noodles* and *pizza*. We also include the sort *step* for temporal reasoning.

3.1 ASP Domain Representation

The ASP program is based on a domain representation that includes a system description $\mathscr{D}_H$ and a history with defaults $\mathscr{H}$. $\mathscr{D}_H$ has a sorted signature that defines the names of objects, functions, and predicates available for use, and axioms that describe a transition diagram τ_H. Fluents are defined in terms of the sorts of their arguments, e.g.:

$$\begin{aligned} &has\_location(thing,location) \\ &in\_hand(robot,object) \\ &is\_open(door) \\ &can\_move(robot,location) \\ &has\_state(thing,state) \end{aligned} \tag{1}$$

The first three are *basic fluents* that obey the laws of inertia and can be changed directly by actions; the last one is a *defined fluent* that is not subject to inertia and cannot be changed directly by an action. Statics such as $connected(location,location)$ and $belongs(location,location)$ specify connections between locations, relation $holds(fluent,step)$ implies that a specific fluent holds at a specific timestep, and $occurs(action,step)$ (hypothesizes) that a specific action occurs at a specific timestep. We then include *actions* such as:

$$\begin{aligned} &move(robot,location) \\ &seat\_person(robot,person,table) \\ &search\_person(robot,area) \\ &pickup(robot,object) \\ &putdown(robot,object) \end{aligned} \tag{2}$$

and define domain dynamics using causal laws such as:

$$\begin{aligned} move(R,L) &\ \textbf{causes}\ has\_location(R,L) \\ pickup(R,O) &\ \textbf{causes}\ in\_hand(R,O) \\ open(R,D) &\ \textbf{causes}\ is\_open(D) \\ seat\_person(R,P,T) &\ \textbf{causes}\ has\_location(P,L) \\ &\ \textbf{if}\ has\_location(T,L) \end{aligned} \tag{3}$$

state constraints such as:

$$\begin{aligned} has\_location(O,L) &\ \textbf{if}\ has\_location(R,L),\ in\_hand(R,O) \\ \neg has\_location(Th,L_2) &\ \textbf{if}\ has\_location(Th,L_1),\ L_1 \neq L_2 \\ has\_location(Th,L_2) &\ \textbf{if}\ has\_location(Th,L_1), \\ &\quad belongs(L_1,L_2) \\ can\_move(R,L_2) &\ \textbf{if}\ has\_location(R,L_1), \\ &\quad connected(L_1,L_2) \end{aligned} \tag{4}$$

and executability conditions such as:

$$\begin{aligned} &\textbf{impossible}\ move(R,L)\ \textbf{if}\ has\_location(R,L) \\ &\textbf{impossible}\ pickup(R,O)\ \textbf{if}\ has\_location(R,L_1), \\ &\quad has\_location(O,L_2), L_1 \neq L_2 \\ &\textbf{impossible}\ open(R,D)\ \textbf{if}\ is\_open(D) \end{aligned} \tag{5}$$

Since robots frequently receive *default* domain knowledge that is true in all but a few exceptional situations, the domain history $\mathscr{H}$, in addition to $hpd(action,step)$ and $obs(fluent,boolean,step)$, the occurrence of specific actions and the observation of specific fluents at specific time steps, contains prioritized defaults describing the values of fluents in their initial states. For instance, it may be initially believed that, unless told otherwise, dishes to be delivered are on a table in a specific area in the kitchen—if they are not there, they are in another area in the kitchen. This information can be represented as:

$$\begin{aligned} &\textbf{init default}\ has\_location(X,a12)\ \textbf{if}\ dish(X), \\ &\quad not\,\neg has\_location(X,a12) \\ &\textbf{init default}\ has\_location(X,a11)\ \textbf{if}\ dish(X), \\ &\quad \neg has\_location(X,a12), \\ &\quad not\,\neg has\_location(X,a11) \end{aligned} \tag{6}$$

where "not" denotes *default negation* (details below). ASP allows us to encode weak exceptions that make defaults inapplicable, or strong exceptions that falsify defaults. Existing definitions of entailment are used to reason with such histories, and any observed exceptions [Zhang *et al.*, 2014].

The domain representation is translated into a program $\Pi(\mathscr{D}_H,\mathscr{H})$ in CR-Prolog that incorporates consistency restoring rules in ASP [Gelfond and Kahl, 2014]. Π includes the causal laws of $\mathscr{D}_H$, inertia axioms, closed world assumption for actions and defined fluents, reality checks,

and records of observations, actions and defaults from $\mathscr{H}$. Every default is turned into an ASP rule and a consistency-restoring (CR) rule that allows us to assume the default's conclusion is false to restore Π's consistency. ASP is based on stable model semantics, and represents 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 introduces concepts such as default negation (i.e., negation by failure) and epistemic disjunction, e.g., unlike "$\neg$a", which implies that "*a is believed to be false*", "not a" only implies that "*a is not believed to be true*"; and unlike "p $\vee \neg$p" in propositional logic, "p or $\neg$p" is not a tautology. It supports non-monotonic reasoning (i.e., adding a statement can reduce the set of inferred consequences), reasoning in large knowledge bases, and reasoning with quantifiers. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π—statements that hold in all such answer sets are program consequences. Inference and planning can be reduced to computing answer sets of program Π by adding a goal, a constraint stating that the goal must be achieved, and a rule generating possible future actions.

In addition to the basic planning capabilities used in the previous architecture, our architecture supports reasoning about exogenous actions to explain the unexpected (observed) outcomes of actions [Balduccini and Gelfond, 2003]. For instance, to reason about a door between the kitchen and the dining room being locked by a human, and to reason about a person moving away from a known location, we introduce exogenous actions $locked(door)$ and $moved\_from(person, location)$ respectively, and add (or revise) axioms:

$$
\begin{aligned}
&is\_open(D) \leftarrow open(R,D),\ \neg ab(D) \qquad (7)\\
&ab(D) \leftarrow locked(D)\\
&\neg has\_location(P,L) \leftarrow moved\_from(P,L),\ has\_location(P,L)
\end{aligned}
$$

where a door is considered *abnormal* (i.e., $ab(D)$) if it has been locked, say by a human. Actions and suitable axioms are included for other situations in a similar manner, e.g., for a dish being identified incorrectly, we include exogenous action: $mixed\_up(dish, state)$, which implies the dish's actual state has been mixed up. We also introduce an *explanation generation* rule and a new relation *expl*:

$$
\begin{aligned}
occurs(A,I) \mid \neg\, occurs(A,I) &\leftarrow exogenous\_action(A) \qquad (8)\\
&\quad I < n\\
expl(A,I) &\leftarrow action(exogenous, A),\\
&\quad occurs(A,I),\ not\ hpd(A,I)
\end{aligned}
$$

where *expl* holds if an exogenous action is hypothesized but there is no matching record in the history. We also include *awareness* axioms and *reality check* axioms:

$$
\begin{aligned}
&\%\ \text{awareness axiom} \qquad (9)\\
&holds(F,0)\ or\ \neg\, holds(F,0) \leftarrow fluent(basic,F)\\
&occurs(A,I) \leftarrow hpd(A,I)\\
&\%\ \text{reality checks}\\
&\leftarrow\ obs(fluent,true,I),\ \neg\, holds(fluent,I)\\
&\leftarrow\ obs(fluent,false,I),\ holds(fluent,I)
\end{aligned}
$$

The awareness axioms guarantee that an inertial fluent's value is always known, and that reasoning takes into account actions that actually happened. Similarly, the reality check axioms cause a contradiction when observations do not match expectations, and the explanation for such unexpected symptoms can be reduced to finding (and extracting suitable statements from) the answer set of the corresponding program [Gelfond and Kahl, 2014]. The new knowledge is included in the ASP program and used for planning and explanation generation. This approach provides *all* explanations of an unexpected symptom—using a CR rule instead of the explanation generation rule (first rule in Statement 8) provides the minimal explanation:

$$occurs(A,I) \overset{+}{\leftarrow} exogenous\_action(A),\ I < n \qquad (10)$$

where, the robot is allowed to assume the occurrence of an exogenous action to restore consistency.

A robot processing sensor inputs (e.g., camera images) is able to extract useful partial descriptions of objects even when the entire object is not visible. To make use of this information, the proposed architecture identifies object occurrences that best explain these partial descriptions. In our illustrative example, we introduce static relations to establish object class membership[1] and introduce relations to capture ideal definitions of object attributes, e.g., for a table:

$$
\begin{aligned}
has\_color(O,white) &\leftarrow member(O,table) \qquad (11)\\
has\_size(O,medium) &\leftarrow member(O,table)\\
has\_wheels(4) &\leftarrow member(O,table),\\
&\quad \neg\, has\_location(O,kitchen)
\end{aligned}
$$

where, tables are white and medium-sized, and have wheels except when they are in the kitchen. Similarly, for a chair:

$$
\begin{aligned}
&has\_color(O,white) \leftarrow member(O,chair) \qquad (12)\\
&has\_size(O,medium) \leftarrow member(O,chair)\\
&\neg has\_wheels(O) \leftarrow member(O,chair)
\end{aligned}
$$

Other objects and object attributes are encoded similarly. As before, a reality check axiom causes an inconsistency when

[1] Relation $member(object, class)$ is applied recursively in a class hierarchy to establish class membership; $is\_a(object, class)$ denotes an instance of a specific class; and $class\_known(object)$ holds for any object whose class label is known.

an object does not have a class label due to incomplete information, and a CR rule restores consistency by allowing the robot to assume the assignment of class labels:

$$\leftarrow \ object(O),\ not\ class\_known(O) \quad \%\ reality\ check$$
$$is\_a(O,C) \stackrel{+}{\leftarrow} \ object(O) \quad \%\ CR\ rule \qquad (13)$$

This assignment of a class label to an object is based on the smallest number of rules that need to be relaxed to support the assignment. This information is also added to the ASP program and be used for subsequent reasoning. However, both planning and object recognition are based on processing sensor inputs and moving to specific locations—these tasks are accomplished probabilistically, as described below.

3.2 Probabilistic Domain Representation

In our recent work, ASP-based reasoning for any specific task was at a coarse resolution (e.g., rooms). For each action in the corresponding plan, the relevant subset of the sorted signature was identified and *refined* by including new sorts (e.g., cells in rooms), fluents and actions, to define a fine-resolution transition diagram. A probabilistic version of this fine-resolution transition diagram was used to construct a POMDP for planning a sequence of primitive actions to implement the abstract action in the ASP-based plan [Sridharan *et al.*, 2015; Zhang *et al.*, 2014]. For instance, to move between two rooms, probabilistic reasoning with a POMDP only considered the cells in these rooms.

Constructing and solving a POMDP becomes computationally intractable, even with state of the art approximate solvers, as the state space increases, e.g., large rooms with many cells, connected to other rooms and corridors. The proposed architecture seeks to address this problem by incorporating three key revisions that modify the resolution at which the robot reasons using ASP and thus probabilistically.

1. First, representation and reasoning in ASP is at a finer resolution (e.g., areas in rooms). As a result, the plans obtained by ASP-based inference consist of less abstract actions, and consider a smaller subset of the domain at any time, e.g., *move* actions cause the robot to move between contiguous areas in rooms.
2. Second, to make this planning tractable, we do selective grounding, i.e., the values that variables in the ASP program can take is based on the task, e.g., to locate a dish known to be in the kitchen, we only consider this dish and areas in the kitchen during planning.
3. Third, with ASP being used to plan at finer resolution, we do not use (or need) the POMDP for probabilistic reasoning. Instead, the robot maintains a probabilistic belief distribution over the relevant subset of the domain. For instance, to move between two contiguous areas, the robot maintains a probability distribution that represents the likelihood that it is in each of these areas—other areas, or beliefs regarding other objects in the domain, are not related to this action.

Such a representation simplifies the computational requirements of probabilistic reasoning. These revisions are motivated by the observation that, for any given task, the robot typically needs to reason at fine resolution, and thus probabilistically, over only a very small subset of the domain. Even for tasks requiring accurate positioning, e.g., picking up or placing a dish, the robot needs to initiate routines for accurate positioning (or control) only after it is in a suitable location.

For a given probabilistic representation, belief revision is performed incrementally based on sensor observations using Bayesian updates. For instance, to update the belief about the location of a specific dish in the dining room:

$$p(E_i|O_i) = \frac{p(O_i|E_i)p(E_i)}{(p(O_i|E_i)p(E_i) + p(O_i|\neg E_i)p(\neg E_i))} \qquad (14)$$

where O_i is the event the dish was observed in area i, E_i is the event the dish exists in area i, making $p(O_i|E_i)$ and $p(E_i|O_i)$ are the observation likelihood and the posterior probability of existence of the dish in each area in the dining room. The initial knowledge, i.e., prior: $p(E_i)$, is based on domain knowledge or statistics collected in an initial training phase (see Section 4). Note that this representation eliminates the need to construct and solve an appropriate POMDP during execution time, significantly improving the computational efficiency of the probabilistic component. Although not described here in detail, a Bayesian state estimation approach is used by the robot to estimate its own position (e.g., *particle filters*), navigate, and to process sensor inputs to extract information about objects being observed.

In summary, for any given task, ASP planning provides a plan with deterministic effects. The first action in the plan and relevant information (from ASP inference) identify the subset of the domain to be represented probabilistically, and set the initial (probabilistic) belief distribution. The action is executed probabilistically, updating the belief distribution based on observations until a high belief indicates action completion or a time limit is exceeded, and adding relevant statements to the ASP program. Any unexpected action outcomes and partial scene descriptions are explained by reasoning about exogenous actions and possible objects. Once these explanations restore consistency, either the next action (in the plan) is selected for execution or a new plan is created. In what follows, we refer to the proposed architecture as the "mixed architecture", and compare it with two algorithms: (1) ASP-based reasoning for completing the assigned tasks; and (2) a (greedy) probabilistic approach that maintains a probabilistic belief distribution and heuristically selects actions (and makes decisions) based on the most likely state.

The mixed architecture raises some subtle issues. First, committing probabilistic beliefs above a specific threshold (e.g., 0.85) as fully certain statements to the ASP program may introduce errors, but the non-monotonic reasoning capability of ASP helps the robot recover. Second, with previous work that reasoned at a coarser resolution with ASP

and used POMDPs for probabilistic planning (a) computing POMDP policies for each ASP action is computationally expensive; and (b) there may be improper reuse of information if the probabilistic belief distribution is not reset between trials [Zhang *et al.*, 2015]. Third, the mixed architecture presents an interesting trade-off between the resolution of symbolic representation and probabilistic representation. Moving most of the reasoning to a symbolic representation can have a negative impact on accuracy and also be computationally expensive—the mixed architecture thus also represents a trade-off between accuracy and efficiency.

4 Experimental Setup and Results

Experiments were conducted in simulation and on a mobile robot in scenarios that mimic a robot waiter in a restaurant. The robot's tasks include finding, greeting and seating people at tables, and delivering orders appropriately.

4.1 Experimental Setup and Hypotheses

The experimental trials used existing implementations of relevant algorithms for control, sensor input processing (including basic speech processing), and simultaneous localization and mapping. In an initial training phase, the robot collected statistics of executing these algorithms to compute: (a) motion error models that estimate errors likely to occur when specific motion commands are executed; and (b) observation likelihood models that provide probability of different observations when specific sensor input processing algorithms are executed. These models were also used to make the simulation trials more realistic.

The experimental trials considered three hypotheses, evaluating whether the mixed architecture: (H1) generates plans for different tasks, and explains unexpected outcomes and partial descriptions extracted from sensor inputs; (H2) significantly improves the task completion accuracy in comparison with just ASP-based reasoning; and (H3) significantly improves task completion time and provides similar accuracy in comparison with a purely probabilistic approach (i.e., with no ASP-based reasoning). To evaluate H1, we primarily used execution traces—some examples are provided in Section 4.2. To evaluate H2 and H3, we conducted experimental trials comparing the mixed architecture with two baseline approaches. The first baseline approach used only ASP-based reasoning. It assumes deterministic action outcomes and generates explanations for the failure of specific plan steps, followed by re-planning to achieve the desired goal. The second baseline approach does not use ASP. Instead, it maintains a probability distribution over the relevant state space, which has been identified in advance for different actions, i.e., before experimental trials are conducted. Observations are used to perform probabilistic belief updates, and a greedy policy is used for action selection, e.g., if a target object is to be found, the robot will look in an area that currently has the largest probability of containing the target. In the results described in Section 4.3, we use "ASP only" and "Probabilistic" (respectively) to refer to these two baseline approaches.

4.2 Execution Traces

We first illustrate the planning and diagnosis capabilities of the architecture by discussing the execution traces for the following example scenarios.

Execution Example 1 *[Unexpected action outcome]*
The task is to return dish $ds1$ to the kitchen from the dining room, e.g., from a table in area $a7$ to the kitchen counter in $a12$ in Figure 3(a). Unknown to the robot, the door *din_kit* between the dining room and kitchen has been locked.

- Some statements describing initial state:

$$holds(has\_location(robot1, dining), 0)$$
$$hold(has\_location(robot1, a7), 0)$$
$$holds(in\_hand(robot1, ds1), 0)$$

The following plan is computed:

$$occurs(move(robot1, a8), 1)$$
$$occurs(move(robot1, din\_kit), 2)$$
$$occurs(open(robot1, din\_kit), 3)$$
$$occurs(move(robot1, a12), 4),$$
$$occurs(putdown(robot1, ds1), 5).$$

- Each step in this plan is executed probabilistically.
- The attempt to open door *din_kit* produces an unexpected observation—$obs(is\_open(din\_kit), false, 3)$ is added to history.
- An explanation $expl(locked(din\_kit), 3)$ obtained by diagnosis invokes Statement 7 to restore consistency.
- The robot solicits human help to unlock the door, and replans to successfully return $ds1$ to the kitchen.

Execution Example 2 *[Observation error]*
The task is to deliver dish *pasta* to *person*1 seated at a table in area $a5$ in Figure 3(a). Unknown to the robot, it picks up the incorrect dish due to an observation error.

- Some statements describing the initial state:

$$holds(has\_location(person1, a5), 0)$$
$$holds(has\_location(robot1, a12), 0)$$
$$holds(has\_state(ds1, pasta), 0)$$
$$holds(has\_state(ds2, pasta), 0)$$

The following plan is computed; for simplicity, we omit intermediate steps through some doors and areas:

$$occurs(pickup(robot1, ds1), 1)$$
$$occurs(move(robot1, a5), 2)$$
$$occurs(putdown(robot1, ds1), 3).$$

- Each step in this plan is executed probabilistically.
- After the dish is placed on the table, the robot observes (from human or camera) that the dish is *noodles*, and $obs(has\_state(ds1, noodles), true, 3)$ is added to history.
- An explanation $expl(mixed\_up(ds1, noodles), 1)$ is obtained through diagnosis to restore consistency.
- The robot then creates a new plan (some steps omitted, once again, for simplicity):

$$occurs(move(robot1, a12), 4)$$
$$occurs(pickup(robot1, ds2), 5)$$
$$occurs(move(robot1, a5), 6)$$
$$occurs(putdown(robot1, ds2), 7)$$

This time the robot delivers the dish ordered.

Execution Example 3 *[Partial scene description]*
The robot delivering a dish sees a medium-sized white object from a distance, but is unable to assign a class label in the absence of any further information.

- The initial knowledge consists of:

$$has\_size(ob_1, medium)$$
$$has\_color(ob_1, white)$$

- The search for explanations generates two possible interpretations using the rules in Statements 11–13:

$$(1)\ \ is\_a(ob_1, table)$$
$$(2)\ \ is\_a(ob_1, chair)$$

- As the robot gets closer, it observes: $has\_wheel(ob_1, 4)$, $has\_location(ob_1, dining)$, i.e., the object has wheels and is in the dining room.
- Statements 11,12 provide the (correct) interpretation: $is\_a(ob_1, table)$.

Execution Example 4 *[Inconsistencies due to defaults]*
The robot has delivered an order to $a8$ and is asked to pickup a dish $ds1$ whose location is not provided. The primary default is that dishes as usually on the table in $a12$, while the secondary default is that it is in $a11$, another area in the kitchen.

- The initial knowledge consists of:

$$holds(has\_location(robot1, a8), 0)$$
$$holds(has\_location(ds1, a12), 0) \quad \%defaultknowledge$$

- The plan steps consist of (some steps involving the door are omitted):

$$occurs(move(robot1, a12, 1))$$
$$occurs(pickup(robot1, ds1), 2)$$

It is assumed that the dish is on the kitchen counter and will be found once the robot reaches $a12$.

- However, when the robot reaches $a12$, it does not observe $ds1$ there. It explains this observation by assuming that the initial state default's conclusion was false.
- After including this knowledge, a plan is obtained based on the secondary default:

$$occurs(move(robot1, a11, 1))$$
$$occurs(pickup(robot1, ds1), 2)$$

- This time, the robot finds and picks up $ds1$.

This example illustrates that inconsistencies that occur when initial state defaults do not hold true, can also be explained by appropriate CR rules.

4.3 Experimental Results

Although experimental trials considered many scenarios, the results reported below are based on scenarios that build on Examples 2 and 4 above. These scenarios are generated by varying the initial location of the robot, the destination, the target object(s), and the locations of objects in the domain. The task completion time and accuracy were used as the evaluation metrics. As stated earlier, we compared the mixed architecture with two baselines: (1) using only ASP-based reasoning; and (2) using a probabilistic approach that selects actions greedily.

Since the positions of the robot and the objects can vary between trials, we cannot compute the average task completion time across different trials. We therefore conducted paired trials—in each paired trial, for each approach being compared, the initial location of the robot and the location of domain objects are the same. In each paired trial, the task completion time using each baseline is then expressed as a ratio of the time taken by the mixed architecture. The results obtained over many such trials are evaluated for statistical significance using the *Mann-Whitney* test.

Simulation Experiments: For running the simulated experiments, we designed a simulated environment that used the error models and observation models developed by collecting statistics from the robot (Section 4.1). It uses a graph-based representation for the physical locations (Figure 2 shows an example), with nodes representing areas with "properties"—this representation supports graphical visualization of the plan steps for debugging.

First, Tables 1-2 summarize the results obtained by considering scenarios based on Examples 2 and 4 respectively—each entry is the average of 500 trials. In Table 1, the accuracy of the probabilistic approach is much lower than that of the mixed architecture or a purely ASP-based approach. This is because the probabilistic approach does not include ASP-based reasoning and thus does not support the diagnosis and replanning capabilities needed to recover from observation failures. In Table 2, the task completion time of the probabilistic approach is much higher than the other two approaches because it does not use the default information. In

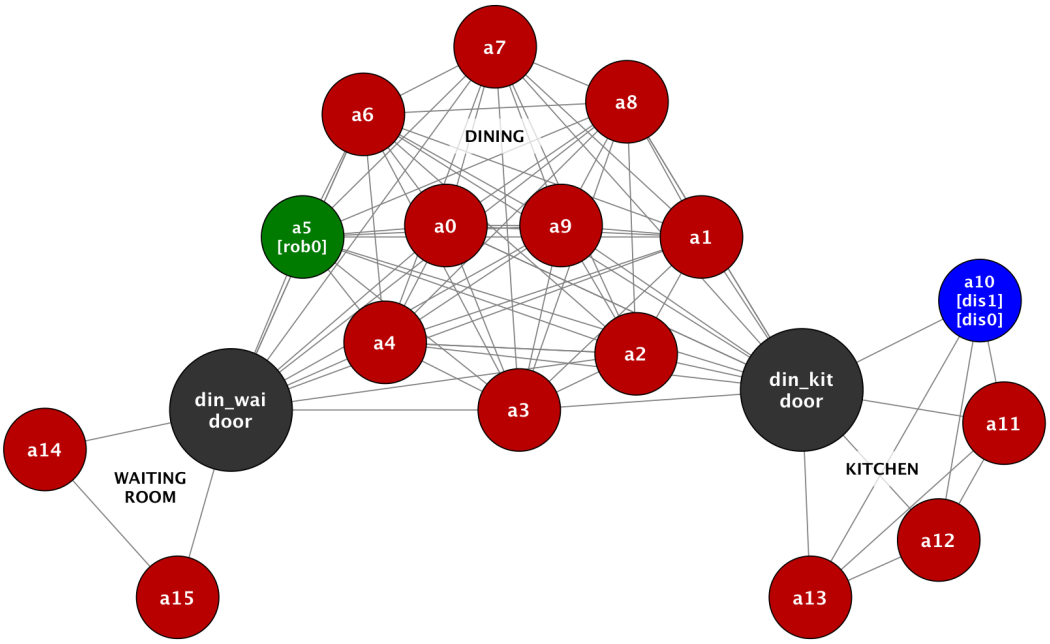


Figure 2: Graph-based representation of the domain's locations in the simulator. Figure shows connections between specific areas, doors, and one possible initial location (green) and final location (blue) for a task to be completed by the robot.

Table 1: Task completion accuracy and time for scenarios based on Example 2. In paired trials, task completion times using only ASP, or a probabilistic approach, are expressed as a factor of the values provided by the mixed architecture. Values that are *statistically significantly* in comparison with the mixed architecture, are indicated in bold font.

Algorithms	Evaluation metrics	
	Accuracy	Time
ASP only	**0.82**	$\mathbf{1.06 \pm 0.6}$
Probabilistic	**0.78**	$\mathbf{1.1 \pm 0.36}$
Mixed approach	0.97	1

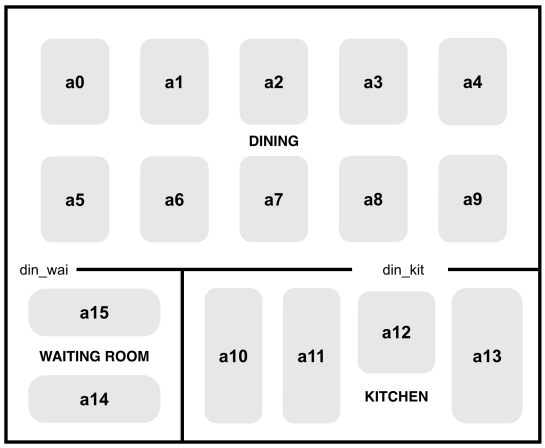
Table 2: Task completion accuracy and time for scenarios based on Example 4. In paired trials, task completion times using only ASP, or a probabilistic approach, are expressed as a factor of the values provided by the mixed architecture. Values that are *statistically significantly* in comparison with the mixed architecture, are indicated in bold font.

Algorithms	Evaluation metrics	
	Accuracy	Time
ASP only	**0.81**	$\mathbf{1.1 \pm 0.5}$
Probabilistic	0.97	$\mathbf{2.29 \pm 1.9}$
Mixed approach	0.96	1

both tables, values that are statistically significant in comparison with the mixed architecture are indicated using bold font. Overall, the results indicate that the mixed architecture (a) improves the accuracy in comparison with ASP-based reasoning; and (b) improves the task completion time and provides similar accuracy in comparison with the probabilistic approach. Furthermore, the planning and execution time are reduced significantly, in comparison with approaches that combined ASP with probabilistic graphical models [Zhang *et al.*, 2014], while providing comparable accuracy. One direction of further research is to consider other domains to better understand the choice of resolution for symbolic and probabilistic representations for a given task and domain.

Robot Experiments: Trials were conducted on a *Turtlebot* (Figure 3(b)) equipped with a Kinect (RGB-D) sensor, range sensors, and an on-board processor running Ubuntu Linux. Our architecture and algorithms were implemented using the Robot Operating System (ROS). Trials included instances of the domain introduced in Section 3, each with one or more tables in specific areas, people and other objects—see Figure 3(a) for an example. The robot was equipped with probabilistic algorithms to determine the attribute values of objects (e.g., color and shape) from camera images, revise the map of the domain, determine its location in the map, and to parse and understand verbal input from humans.

Table 3 summarizes the results of trials based on scenarios drawn from Example 4. A purely ASP-based approach was



(a) Example domain.

(b) Turtlebot robot.

Figure 3: (a) Example map of illustrative domain used for experimental evaluation, with rooms, areas and doors (people and robot not shown); for simplicity, each area is assumed to have one table, but this assumption can be easily relaxed; and (b) the Turtlebot mobile robot platform.

Table 3: Task completion accuracy and time for robot experiments in scenarios based on Example 4. In paired trials, task completion times of a probabilistic approach are expressed as a factor of the values provided by the mixed architecture. The proposed mixed architecture *significantly* reduces the task completion time while providing high accuracy.

Algorithms	Evaluation metrics	
	Accuracy	Time
Probabilistic	0.9	$\mathbf{2.3 \pm 1.72}$
Mixed approach	0.9	1

not included in these experiments because the results can vary arbitrarily based on the uncertainty in sensing and actuation. Results indicate that the robot was able to use the mixed architecture to successfully complete the assigned tasks in all such scenarios, with results of paired trials being similar to those obtained in simulation (for similar scenarios). The overall trend, i.e., the mixed architecture significantly reducing task completion time while providing high accuracy, was the same in many other scenarios as well.

Finally, we provide a video of an experimental trial illustrating the planning and diagnostics capabilities. In this video, the task is to fetch a bag of crisps for a human, and to deal with an unexpected failure (similar to Example 2 in Section 4.2). Since object manipulation is not a focus of this work, it is abstracted away during trials on the physical robot (they are included in the simulated trials)—once the robot reaches the desired location and locates the desired object or human, it verbally solicits human help to complete the desired interaction, e.g., to place the object on the robot. The video can be viewed online: `https://vimeo.com/136990534`

5 Conclusions

This paper described an architecture that mixes the complementary strengths of declarative programming and probabilistic belief updates. Plans created using ASP-based nonmonotonic logical reasoning are implemented probabilistically, with high probability observations and action outcomes adding statements to the ASP program. The architecture enables a robot to explain unexpected action outcomes by reasoning about exogenous actions, and to identify objects that best explain partial scene descriptions. These capabilities have been demonstrated through experimental trials in simulation and on a mobile robot in scenarios that mimic a robot waiter assisting in delivering orders in a restaurant. Future work will further investigate the tight coupling and transfer of control between the logical and probabilistic representations, with the long-term objective of equipping robots with commonsense reasoning capabilities to collaborate with humans in complex application domains.

Acknowledgments

The authors thank Michael Gelfond and Rashmica Gupta for discussions related to the architecture described in this paper. This work was supported in part by the US Office of Naval Research Science of Autonomy award N00014-13-1-0766. All opinions and conclusions described in this paper are those of the authors.

References

[Bai *et al.*, 2014] Haoyu Bai, David Hsu, and Wee Sun Lee. Integrated Perception and Planning in the Continuous Space: A POMDP Approach. *International Journal of Robotics Research*, 33(8), 2014.

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

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, 9(1):57–144, January 2009.

[Erdem *et al.*, 2012] Esra Erdem, Erdi Aker, and Volkan Patoglu. Answer Set Programming for Collaborative Housekeeping Robotics: Representation, Reasoning, and Execution. *Intelligent Service Robotics*, 5(4):275–291, 2012.

[Galindo *et al.*, 2008] Cipriano Galindo, Juan-Antonio Fernandez-Madrigal, Javier Gonzalez, and Alessandro Saffioti. Robot Task Planning using Semantic Maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

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

[Hanheide *et al.*, 2015] Marc Hanheide, Moritz Gobelbecker, Graham Horn, Andrzej Pronobis, Kristoffer Sjoo, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, Hendrik Zender, Geert-Jan Kruijff, Nick Hawes, and Jeremy Wyatt. Robot Task Planning and Explanation in Open and Uncertain Worlds. *Artificial Intelligence*, 2015.

[Kaelbling and Lozano-Perez, 2013] Leslie Kaelbling and Tomas Lozano-Perez. Integrated Task and Motion Planning in Belief Space. *International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

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

[Meadows *et al.*, 2013] Ben Meadows, Pat Langley, and Miranda Emery. Seeing Beyond Shadows: Incremental Abductive Reasoning for Plan Understanding. In *AAAI Workshop on Plan, Activity and Intent Recognition*, Bellevue, USA, March 25-27, 2013.

[Milch *et al.*, 2006] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic Models with Unknown Objects. In *Statistical Relational Learning*. MIT Press, 2006.

[Ng and Mooney, 1992] Hwee Tou Ng and Raymond J. Mooney. Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In *Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 499–508. Morgan Kaufmann, 1992.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–95, 1987.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, February 2006.

[Saribatur *et al.*, 2014] Zeynep Saribatur, Esra Erdem, and Volkan Patoglu. Cognitive Factories with Multiple Teams of Heterogeneous Robots: Hybrid Reasoning for Optimal Feasible Global Plans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[Sindhu Raghavan and Raymond J. Mooney, 2011] Sindhu Raghavan and Raymond J. Mooney. Abductive Plan Recognition by Extending Bayesian Logic Programs. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 629–644, September 2011.

[Sridharan *et al.*, 2015] Mohan Sridharan, Michael Gelfond, Shiqi Zhang, and Jeremy Wyatt. A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Technical report, CoRR abstract: `http://arxiv.org/abs/1508.03891`, August 2015.

[Zhang and Stone, 2015] Shiqi Zhang and Peter Stone. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *AAAI Conference on Artificial Intelligence*, pages 1394–1400, Austin, USA, 2015.

[Zhang *et al.*, 2014] Shiqi Zhang, Mohan Sridharan, Michael Gelfond, and Jeremy Wyatt. Towards An Architecture for Knowledge Representation and Reasoning in Robotics. In *International Conference on Social Robotics (ICSR)*, pages 400–410, Sydney, Australia, October 27-29, 2014.

[Zhang *et al.*, 2015] Shiqi Zhang, Mohan Sridharan, and Jeremy Wyatt. Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.