In the *Hybrid Reasoning Workshop* at the International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina, July 26, 2015.

Mixing Non-Monotonic Logical Reasoning and Probabilistic Planning for Robots

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

Michael Gelfond
Department of Computer Science
Texas Tech University, USA
michael.gelfond@ttu.edu

Shiqi Zhang
Department of Computer Science
The University of Texas at Austin, USA
szhang@cs.utexas.edu

Jeremy Wyatt
School of Computer Science
University of Birmingham, UK
jlw@cs.bham.ac.uk

Abstract

This paper describes an architecture that combines the complementary strengths of probabilistic graphical models and declarative programming to represent and reason with qualitative and quantitative descriptions of domain knowledge and uncertainty. An action language is used for the architecture's low-level (LL) and high-level (HL) system descriptions, and the HL definition of recorded history is expanded to allow prioritized defaults. For any given objective, each action in the plan created in the HL using non-monotonic logical reasoning is executed probabilistically in the LL, refining the HL description to identify the relevant sorts, fluents and actions, and adding the corresponding action outcomes to the HL history. The HL and LL domain representations are translated into an Answer Set Prolog (ASP) program and a partially observable Markov decision process (POMDP) respectively. ASP-based inference provides a multinomial prior for POMDP state estimation, and populates a Beta density of priors for metareasoning and early termination. Robots equipped with this architecture reason with violation of defaults, noisy observations and unreliable actions in complex domains. The architecture is evaluated in simulation and on a mobile robot moving target objects to desired locations in an office domain.

1 Introduction

Robots deployed in assistive roles in complex domains have to represent knowledge and reason at both the sensorimotor level and the cognitive level. Although it is challenging for robots to operate in such domains without considerable domain knowledge, it is equally difficult to equip robots with complete and accurate domain knowledge. In addition, human participants may not have the time and expertise to interpret raw sensor data, or to provide comprehensive feedback. Our architecture combines the knowledge representation and non-monotonic logical reasoning capabilities of declarative programming with the planning and uncertainty modeling capabilities of probabilistic graphical models [Zhang *et al.*, 2014; 2015]. The architecture is a significant step towards addressing the fundamental challenge of representing, and reasoning with, qualitative and quantitative descriptions of uncertainty and domain knowledge obtained from different sources. The architecture's key features include:

- An action language is used for the system descriptions of the high-level (HL, symbolic) and low-level (LL, probabilistic) domain representations. The HL domain representation includes knowledge that holds in all but a few exceptional situations.
- For any given objective, tentative plans created in the HL using non-monotonic logical reasoning are implemented in the LL using probabilistic algorithms, with the observations of action outcomes added to the HL history.
- The result of inference in the HL is used to build a multinomial prior for state estimation in the LL, and to build a Beta density model of priors that includes historical data for metareasoning and early termination of appropriate tasks.

The HL and LL representations are translated into an Answer Set Prolog (ASP) program and a partially observable Markov decision process (POMDP) respectively, supporting reasoning with violation of defaults, noisy observations and unreliable actions. We illustrate the architecture's capabilities in simulation and on a mobile robot moving objects characterized by visual cues to specific places in an office domain.

2 Related Work

Probabilistic graphical models such as POMDPs have been used to plan sensing, navigation and interaction for robots [Rosenthal and Veloso, 2012]. However, these formulations (by themselves) make it difficult to perform commonsense reasoning. Research in classical planning has provided many algorithms for knowledge representation and logical reasoning, but these algorithms require prior knowledge about the domain, tasks, and actions. Many such algorithms also do not support merging of new, unreliable information with the current beliefs in a knowledge base. ASP, a declarative programming paradigm, is well-suited for representing and reasoning with commonsense knowledge [Gelfond and Kahl, 2014]. It has been used to enable robotics applications, e.g., planning and diagnosis by simulated robot housekeepers [Erdem *et al.*, 2012]. However, ASP does not support probabilistic analysis of uncertainty, whereas a lot of information obtained from sensors and actuators is represented probabilistically to quantitatively model the uncertainty.

Researchers have designed architectures and algorithms that combine deterministic and probabilistic algorithms for task and motion planning on robots [Hanheide *et al.*, 2011; Kaelbling and Lozano-Perez, 2013]. Declarative programming and continuous-time planners have been used for path planning in robot teams [Saribatur *et al.*, 2014], and a probabilistic extension of ASP has been combined with POMDPs for logical inference and probabilistic planning in human-robot dialog [Zhang and Stone, 2015]. Examples of principled algorithms that combine logical and probabilistic reasoning include probabilistic first-order logic [Halpern, 2003], 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 for probabilistically modeling uncertainty do not provide the desired expressiveness for commonsense reasoning, e.g., it is not always possible to express degrees of belief quantitatively. Other algorithms based on logic programming that support probabilistic reasoning do not support one or more of the desired capabilities such as reasoning as in causal Bayesian networks; incremental revision of (probabilistic) information; and reasoning with large probabilistic components. To address these challenges, we have developed architectures that couple the complementary strengths of declarative programming and probabilistic graphical models to support logical inference, deterministic planning, and probabilistic planning on robots [Zhang *et al.*, 2014; 2015]. This paper summarizes and illustrates the default reasoning, probabilistic planning, and metareasoning capabilities of our current architecture.

3 Proposed Architecture

In the proposed architecture shown in Figure 1(a), the high-level (HL) representation is translated to an Answer Set Prolog (ASP) program, i.e., the *knowledge base* (KB), which includes default knowledge and is used for non-monotonic logical inference and deterministic planning. For any specific task, the execution of the each action in the HL plan is formulated as a partially observable Markov decision process (POMDP) in the LL, automatically refining the HL description to identify the sorts, fluents and actions that need to be modeled probabilistically. The *answer set* obtained by inference in the ASP program is used to construct (a) a multinomial prior for POMDP state estimation; and (b) a Beta density prior that includes historical data from comparable domains for metareasoning and early termination of suitable tasks. The POMDP *policy* is used to select and execute a sequence of LL actions, updating the probabilistic *belief state* based on the corresponding observations (and observations from algorithms always in use, e.g., for obstacle avoidance). Any resultant belief with high probability commits a statement to the ASP KB representing the outcome of the HL action's execution. Figure 1(b) shows this transfer of control in the context of locating desired objects. The architecture's components are described below.

The syntax and semantics of the transition diagrams of the HL and LL domain representations are described in an *action language* AL [Gelfond and Kahl, 2014]. AL has a sorted signature containing three *sorts*: *statics*, domain properties whose truth values cannot be changed by actions; *fluents*, properties whose values are changed by actions; and *actions*, elementary actions that can be executed in parallel. AL allows three types of statements:

$$\begin{array}{ll} a \;\textbf{causes}\; l_{in} \;\textbf{if}\; p_0,\ldots,p_m & \text{(Causal law)}\\ l \;\textbf{if}\; p_0,\ldots,p_m & \text{(State constraint)}\\ \textbf{impossible}\; a_0,\ldots,a_k \;\textbf{if}\; p_0,\ldots,p_m & \text{(Executability condition)} \end{array}$$

where a is an action, l is a literal, l_{in} is an inertial fluent, and $p_0,\ldots,p_m$ are domain literals (i.e., a property or its negation). A collection of AL statements forms a system description. As an illustrative example used throughout this paper, consider a robot moving objects of specific sorts and with visual attributes, in a domain with places such as $office$, $library$, and $kitchen$—we reason at the coarser resolution of places in the HL and at the finer resolution of grid cells in the LL.

3.1 HL domain representation

The HL domain representation consists of a system description $\mathscr{D}_H$ and a history with defaults $\mathscr{H}$. $\mathscr{D}_H$ has a sorted signature (Σ_H), which defines the names of objects, functions, and predicates in the HL, and axioms, to describe the transition diagram τ_H. The sorts in our example are: $place$, $thing$, $robot$, and $object$; $object$ and $robot$ are subsorts of $thing$; and $textbook$, $printer$, and $kitchenware$ are subsorts of $object$. Fluents of the domain are defined in terms of their arguments: $loc(thing, place)$ and $in\_hand(robot, object)$. The first predicate describes a thing's location, and the second states that a robot is holding an object. These are *inertial fluents* subject to the laws of inertia. The domain has three actions: $move(robot, place)$, $grasp(robot, object)$, and $putdown(robot, object)$. The axioms defining the domain dynamics consist of causal laws such as:

$$\begin{array}{l} move(Robot, Pl) \;\textbf{causes}\; loc(Robot, Pl) \\ grasp(Robot, Ob) \;\textbf{causes}\; in\_hand(Robot, Ob) \end{array} \tag{1}$$

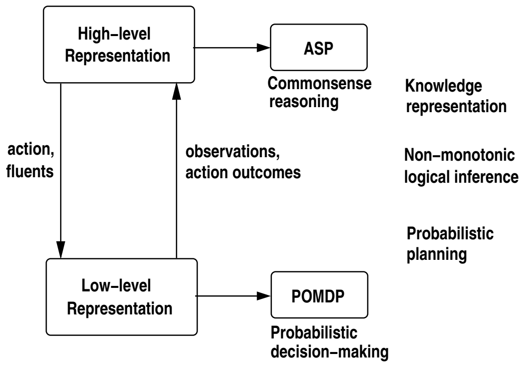
state constraints such as:

$$\begin{array}{l} loc(Ob, Pl) \;\textbf{if}\; loc(Robot, Pl),\; in\_hand(Robot, Ob) \\ \neg loc(Th, Pl_1) \;\textbf{if}\; loc(Th, Pl_2),\; Pl_1 \neq Pl_2 \end{array} \tag{2}$$

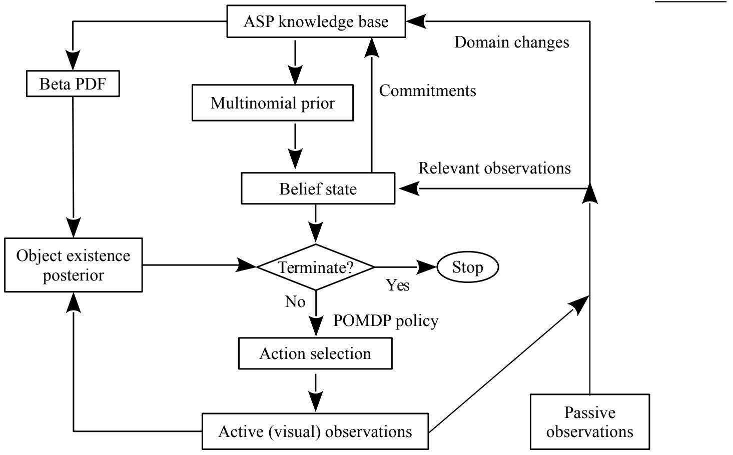
and executability conditions such as:

$$\begin{array}{l} \textbf{impossible}\; move(Robot, Pl) \;\textbf{if}\; loc(Robot, Pl) \\ \textbf{impossible}\; grasp(Robot, Ob) \;\textbf{if}\; loc(Robot, Pl_1) \\ \qquad\qquad\qquad\qquad\qquad loc(Ob, Pl_2), Pl_1 \neq Pl_2 \end{array} \tag{3}$$

A dynamic domain's recorded history is usually a collection of records of the form $obs(fluent, boolean, step)$, i.e.,


High–level Representation
ASP
Commonsense reasoning
Knowledge representation
action, fluents
observations, action outcomes
Non–monotonic logical inference
Probabilistic planning
Low–level Representation
POMDP
Probabilistic decision–making

(a) Architecture overview.



(b) Belief merging.

Figure 1: (a) Architecture combines the complementary strengths of declarative programming and probabilistic graphical models; (b) ASP-based inference provides multinomial prior for POMDP state estimation, and supports metareasoning.

a fluent observed to be true or false at a given step, and $hpd(action, step)$, i.e., a action happened at a given step; we abbreviate $obs(f, true, 0)$ and $obs(f, false, 0)$ as $init(f, true)$ and $init(f, false)$ respectively. *We expand on this view by allowing histories to contain (prioritized) defaults describing the values of fluents in their initial states.* We provide some illustrative examples below—see [Gelfond and Kahl, 2014] for formal semantics of defaults.

Example 1 *[Example of defaults]*
Textbooks are usually in the main library. If a textbook is not there, it is in the office. These statements about the locations of textbooks in the initial state can be represented as:

$$\begin{aligned} &\textbf{initial default}\ loc(X, library)\ \textbf{if}\ \ textbook(X) \qquad (4)\\ &\textbf{initial default}\ loc(X, office)\ \textbf{if}\ \ textbook(X),\\ &\qquad\qquad\qquad\qquad\qquad\qquad \neg loc(X, library) \end{aligned}$$

A history $\mathscr{H}_a$ with the above statements entails $holds(loc(Tb_1, library), 0)$ for textbook Tb_1. History $\mathscr{H}_b$ that adds observation $init(loc(Tb_1, library), false)$ to $\mathscr{H}_1$ renders the first default inapplicable; it entails $holds(loc(Tb_1, office), 0)$ based on the second default. History $\mathscr{H}_c$ that adds $obs(loc(Tb_1, library), false, 1)$ to $\mathscr{H}_a$ defeats the first default because if this default's conclusion is true in the initial state, it is also true at step 1 (inertia), which contradicts our observation. The second default will conclude that this book was initially in the $office$; inertia axioms will propagate this information to entail $holds(loc(Tb_1, office), 1)$.

To define the entailment relation with respect to $\mathscr{D}_H$, we define a *state* of τ_H *compatible* with a description $\mathscr{I}$ of the initial state of $\mathscr{H}$, i.e., a collection of defaults and the set of observations of $\mathscr{H}$ at time step 0. We also define *models* of $\mathscr{H}$, i.e., paths of τ_H compatible with $\mathscr{H}$. For formal definitions of compatible states, models and entailment, and to see that our notion of entailment captures the intuition corresponding to histories from Example 1, see [Zhang *et al.*, 2014].

The HL domain representation is translated into program $\Pi(\mathscr{D}_H, \mathscr{H})$ in CR-Prolog that introduces consistency restoring (CR) rules in ASP [Balduccini and Gelfond, 2003]. Π consists of causal laws of $\mathscr{D}_H$, inertia axioms, closed world assumption for actions and defined fluents, reality checks, records of observations, actions and defaults from $\mathscr{H}$, and special axioms for $init$: $holds(F, 0) \leftarrow init(F, true)$ and $\neg holds(F, 0) \leftarrow init(F, false)$. Every initial state default, which states that elements of class c satisfying property b typically have property p, is turned into an ASP rule and a CR rule as follows:

$$\begin{aligned} &holds(p(X), 0) \leftarrow c(X), holds(b(X), 0),\ not\ \neg holds(p(X), 0)\\ &\neg holds(p(X), 0) \stackrel{+}{\leftarrow} c(X), holds(b(X), 0) \quad \%\ \text{CR rule} \qquad (5) \end{aligned}$$

where the CR rule allows us to assume the default's conclusion is false to restore Π's consistency—see [Gelfond and Kahl, 2014]. ASP is based on stable model semantics and non-monotonic logics; it can represent recursive definitions, defaults, causal relations, and language constructs that are difficult to express in classical logic formalisms [Baral, 2003]. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π. Program consequences are statements that are true in all such belief sets—the following discussion assumes that inference produces only one answer set. We define the relation between models and answer sets to reduce (a) the computation of models of $\mathscr{H}$ to computing answer sets of a CR-Prolog program; and (b) a planning task to computing answer sets of a program obtained by adding to $\Pi(\mathscr{D}_H, \mathscr{H})$ a goal definition, a constraint stating that the goal must be achieved, and a rule generating possible future actions [Zhang *et al.*, 2014].

3.2 LL domain representation

The LL system description $\mathscr{D}_L$ has a sorted signature Σ_L and axioms that describe transition diagram τ_L. The LL representation is considered as a *refinement* of the HL representation. Signature Σ_L includes sorts from Σ_H and additional sorts that are subsorts of those in Σ_H, e.g., *cell* and *room* are subsorts of *place*, and satisfy static relation $part\_of(cell, room)$. The set of actions in Σ_L includes HL actions represented at a finer resolution, and new actions for sensor input processing, e.g., $search(cell, object)$. Fluents of Σ_L include those of Σ_H and additional fluents to model knowledge producing actions, e.g., to model the effect of *search*, we have an inertial fluent: $searched(cell, object)$—a cell was searched for an object—and two defined fluents: $found(object, place)$ and $continue\_search(room, object)$. The causal laws and constraints in the LL are defined appropriately—see [Zhang *et al.*, 2014]. In this action theory that describes τ_L, states are viewed as extensions of states of τ_H by physically possible fluents and statics defined in the language of the LL. Moreover, for every HL state transition $\langle\sigma, a, \sigma'\rangle$ and every LL state s compatible with σ, there is a path in the LL from s to some state compatible with σ'.

Action outcomes and observations in the LL are only known with some degree of probability. The execution of each action in an HL plan is formulated as a POMDP, defined by the tuple $\langle S, A, Z, T, O, R\rangle$. The first three elements are the set of states, actions and values of the observable fluents. Since the LL states are partially observable, the POMDP considers *belief states*, probability distributions over the set of states. The function $T : S\times A\times S' \rightarrow [0,1]$ defines the LL state transition probabilities, and the function $O : S\times A\times Z \rightarrow [0,1]$ defines the probability of obtaining specific observations by executing specific actions in specific states. Both T and O are computed from prior knowledge or statistics collected by repeatedly executing different actions in different states. They describe a probabilistic transition diagram over belief states, and support iterative Bayesian update of the belief state B_t: $B_{t+1}(s_{t+1}) \propto O(s_{t+1}, o_{t+1}, a_{t+1}) \sum_s T(s, a_{t+1}, s_{t+1}) \cdot B_t(s)$. The POMDP tuple also includes a reward function $R : S\times A\times S' \rightarrow \Re$ that specifies the relative *utility* of different actions in different states with regard to the HL action to be executed. Planning in the LL involves computing a *policy* that maximizes the cumulative reward over a planning horizon to map belief states to actions: $\pi : B_t \mapsto a_{t+1}$—we use an approximate solver to compute this policy [Ong *et al.*, 2010]. Plan execution uses the policy to repeatedly choose an action in the current belief state, updating the belief state after executing the action and receiving an observation. The LL history thus stores observations and actions over one time step. We call this algorithm "POMDP-1".

Due to the refinement-based architecture, *the POMDP for any specific HL action is constructed by considering only the relevant sorts, fluents and actions*, significantly improving the computational efficiency and (indirectly) the accuracy. For instance, to execute an HL action to move between two places, the LL formulation only considers the cells in these two places, actions that move the robot between these cells, and fluents that can influence or are affected by these actions.

The combination of logical and probabilistic beliefs includes some interesting contributions. First, consider the computation of a multinomial prior for POMDP state estimation from an answer set. For any HL action that is to be executed, the relevant literals in the answer set are identified to create the POMDP. Then, *Fechner's law*, a logarithmic law in Psychophysics that relates visual perception to sensory stimulus, and some postulates motivated by commonsense knowledge of visual perception, are used to convert the count of these literals to a multinomial prior [Zhang *et al.*, 2015]. For instance, if the task is to locate an object in a set of rooms, this prior is the probability of the object being in each room: b_i^{KB}. The multinomial prior probability for each room is distributed over cells in the room.

Second, a Bayesian treatment is provided for computing the posterior distribution from the multinomial prior and POMDP belief distribution. Since the answer set is subject to non-monotonic inference, using a new prior to revise the posterior computed with the previous prior is challenging. The fact that actions in the domain do not change object locations is exploited to maintain the likelihood of the sequence of observations received by the robot. When a new multinomial prior is obtained, Bayes rule can then be used to compute the revised posterior belief from the prior and the observation sequence likelihood, e.g., for localizing a target object:

$$b'_{i,t} \propto b_{i,t}^{Ob} \cdot b_i^{KB} \tag{6}$$

where B_t^{Ob} is the observation sequence likelihood and B'_t is the posterior. Since this strategy does not reset the observation likelihood after each commit to the ASP KB, observational information may be reused. While this is strictly incorrect in Bayesian terms, we use it because it (experimentally) improves task completion accuracy and time.

Finally, the answer set is used for metareasoning in the context of localizing specific target objects. It comprises three steps: (1) using a Beta (meta) density to model prior knowledge from historical data and the answer set about an object's existence in the domain; (2) maintaining the likelihood of the observation sequence ($o_{1:t}$) given the existence or non-existence of the object in the domain; and (3) using the prior and the likelihood to obtain the posterior of object's existence in the domain. The prior probability that the target exists in the current domain is $\theta = P(E)$, the parameter of a Bernoulli distribution. A Beta PDF models the meta density over θ, i.e., the conjugate prior—the count of relevant literals (identified by refinement during POMDP creation) in the answer set and the count of similar objects from comparable domains set the Beta PDF's parameters. This PDF is used with the observation sequence likelihood to compute the posterior of the target's existence in tne domain:

$$p(E|o_{1:t}) = \int_\theta p_\theta(E|o_{1:t})\, p(\theta) d\theta \tag{7}$$

where $p(\theta)$ is modeled by the Beta PDF, E is the event that the object exists in the domain, and $p_\theta(E|o_{1:t})$ is computed iteratively (using Bayes rule) based on $p(o_t|E)$ and $p(o_t|\neg E)$, shorthand for $p(o_t|E, a_t, b_t)$ and $p(o_t|\neg E, a_t, b_t)$ respectively, i.e., the observation likelihoods at a specific time step given the object exists or does not exist in the domain. Since computing this posterior (i.e., integral) in closed form is difficult, we explore three approximations: (1) using the expectation of the Beta PDF as the prior probability of existence of the target; (2) using the 90% upper bound of the value of the prior probability; and (3) using Monte Carlo sampling to estimate the integral. In all three strategies, if the posterior falls below a preset threshold, the corresponding trial (to compute the location of the target object) is terminated—for details, see [Zhang *et al.*, 2015].

4 Experimental Setup and Results

This section describes the experimental setup, and the result of evaluation in simulation and on a mobile robot.

Experimental setup: In an initial training phase, the robot acquired data, e.g., computational time and accuracy, by executing different actuation and sensor input processing algorithms. The data was used to define the probabilistic components of the LL domain representation, including models that represent objects as probabilistic functions of attributes extracted from images, and models of the robot's motion—these models also make the simulator more realistic [Zhang *et al.*, 2015].

The goal in each experimental trial, unless stated otherwise, was to move specific objects to specific places—the robot's location, target object, and locations of objects were chosen randomly. An action sequence extracted from an answer set of the ASP program provides an HL plan, e.g., the plan to move textbook Tb_1 from the *library* to the *office*: *move(robot, library)*, $grasp(robot, Tb_1)$, $move(robot, office)$, $putdown(robot, Tb_1)$. An object's location in the LL is known with certainty if the belief (in a cell) exceeds a threshold (0.85). Our architecture, henceforth referred to as "PA", was compared with: (1) POMDP-1; and (2) POMDP-2, which revises POMDP-1 by assigning high probability values to defaults to bias the initial belief. We evaluated three hypotheses: (H1) PA achieves goals more reliably and efficiently than POMDP-1; (H2) our representation of defaults improves reliability and efficiency in comparison with not using defaults or assigning high probability values to defaults; and (H3) metareasoning with domain-specific observations and historical data reliably and efficiently determines when a trial should be terminated. During the evaluation of hypotheses H1 and H2, metareasoning was not included.

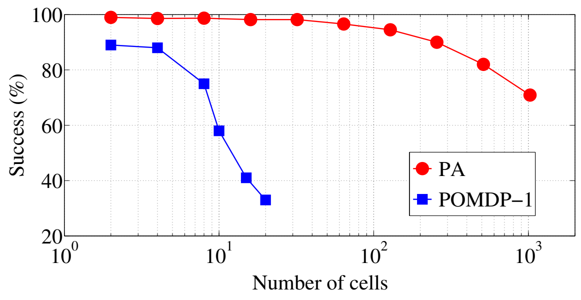
Simulation Experiments: To evaluate H1, we first compared PA with POMDP-1 in trials in which the robot's initial position is known but the position of the object to be moved is unknown. The solver used in POMDP-1 is given a fixed amount of time to compute action policies. Figure 2(a) summarizes the ability to successfully achieve the assigned goal, as a function of the number of cells in the domain. Each data point in Figure 2(a) is the average of 1000 trials, and each room is set to have four cells (for ease of interpretation). PA significantly improves the robot's ability to achieve the assigned goal in comparison with POMDP-1. As the number of cells in the domain increases, it becomes computationally difficult to generate good POMDP action policies that, in conjunction with incorrect observations (e.g., false positives), significantly impacts the ability to complete the trials. PA directs the robot's attention to appropriate regions, e.g., specific rooms and cells in the domain. As the domain size increases, the creation of multiple plans of similar cost may (with incorrect observations) affect the ability to achieve desired goals—the impact is, however, much less pronounced.

Next, we computed the time taken by PA to generate a plan as the number of rooms and objects increases. We conducted three sets of experiments in which the robot reasons with: (1) all available knowledge of objects and rooms; (2) only knowledge relevant to the assigned goal—e.g., if the robot knows an object's default location, it need not reason about other objects and rooms to locate the object; and (3) relevant knowledge and knowledge of an additional 20% of randomly selected objects and rooms. Figure 2(b) shows that PA generates appropriate plans for domains with a large number of rooms and objects. Using only the knowledge relevant to the goal significantly reduces the planning time—this knowledge is identified when the HL representation is refined to obtain the LL representation. Furthermore, it soon becomes computationally intractable to generate a plan with POMDP-1 for domains with many objects and rooms, even with hierarchical decompositions—these results are not shown in Figure 2(b).

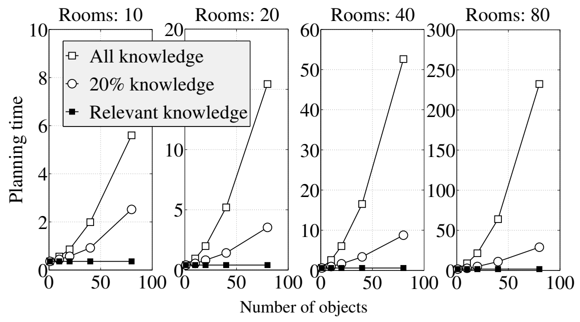
To evaluate H2, we first compared PA with PA^*, a version that does not include any default knowledge. Figure 3(a) summarizes the average number of actions executed per trial as a function of the number of rooms—each data point is the average of 10000 trials. We observe that the principled use of default knowledge significantly reduces the number of actions (and thus time) required to achieve the assigned goal. Next PA was compared with POMDP-2, which assigns high probability values to default information and revises the initial belief. The results with POMDP-2 can vary depending on: (a) the numerical value chosen; and (b) whether the ground truth matches the default information. For instance, *if a large probability is assigned to the default knowledge that books are typically in the library, but the book the robot has to move is an exception, POMDP-2 takes a large amount of time to recover from the initial belief.* PA, on the other hand, can encode and reason with exceptions to initial defaults.

To evaluate H3, we conducted trials in which the robot had to localize specific objects in the domain. The KB was static in each trial to isolate the effect of using observations, and the target was randomly selected to be present or absent (unknown to the robot). The "baseline" strategy terminated a trial when the probability of one of the grid cells exceeded the preset threshold ($\tau^+ = 0.8$) or a time limit was exceeded. This strategy was compared with the action selection policies corresponding to the three approximation strategies for computing Equation 7 in Section 3.2: "expectation", "upper bound" and "sampling". Each of these three policies terminated a trial early if the posterior of the target's non-existence in the domain exceeded a threshold (τ^-) that was set experimentally. All four policies used the multinomial prior based on ASP inference for POMDP state estimation.

Figure 3(b) summarizes the results ($\tau^- = 0.7$), with the localization time and accuracy on the x-axis and y-axis respectively. The black plot with plus-shaped markers depicts the average results with the baseline strategy and specific time limits; the robot can localize the target more accurately if given more time. However, in trials in which the target object does not exist in the domain, the baseline strategy cannot terminate trials early. The action selection policies based on the approximation strategies enable early termination by updating the belief of the target's existence in the domain using historical data and observations. All three approximation strategies provide significantly lower target localization time in comparison with the baseline—an indirect consequence is the increase in localization accuracy. For instance, to obtain a

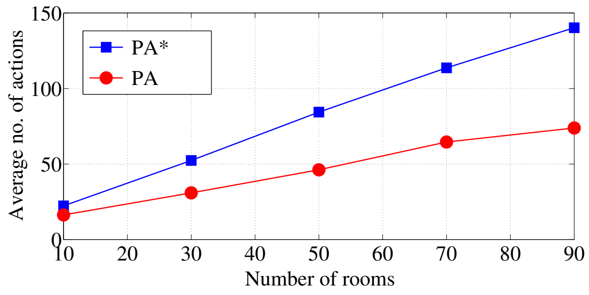


(a) Success rate.

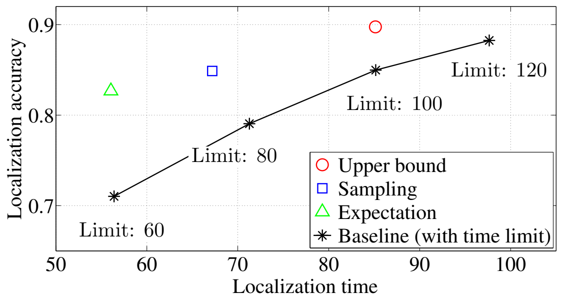


(b) Planning time.

Figure 2: (a) With a limit on the time provided to compute policies, PA significantly increases accuracy in comparison with POMDP-1 as the number of cells increases; (b) Planning time as a function of the number of rooms and the number of objects in the domain—*PA* scales to larger number of rooms and objects.



(a) Default knowledge.



(b) Metareasoning strategies.

Figure 3: (a) Principled representation of defaults significantly reduces the number of actions (and thus time) for achieving assigned goal; (b) Metareasoning with historical data reduces target localization time (by early termination) and (indirectly) increases the accuracy. Paired trials establish statistical significance of the results.

target localization accuracy of 0.85, the sampling-based strategy takes ≈ 67 time units while the baseline strategy needs ≈ 85 units. The three proposed strategies also result in different trade-offs between computational efficiency and target localization accuracy (and time). For instance, the expectation-based strategy provides the lowest localization time, but the localization accuracy is also the lowest among the approximation strategies. The upper bound strategy, on the other hand, has the highest localization time but provides the highest localization accuracy. The sampling-based strategy provides a trade-off between accuracy and time. Overall, the sampling-based and upper bound strategies result in better performance because they better exploit the variance of the Beta PDF.

Robot Experiments: Finally, PA was compared with POMDP-1 on a wheeled robot over 50 trials on two floors. The robot recognizes objects by processing images from a camera. It uses input data from range finders for simultaneously building (and revising) a map of the domain and estimating its location in the map. Since manipulation is not a focus of this work, the robot asks for the desired object to be placed in its gripper once it is next to it. All algorithms are implemented using the Robot Operating System (ROS). On each floor, we considered ≈ 15 rooms, including offices, labs, common areas and corridors. To use POMDP-1 in such large domains, we used a hierarchical decomposition based on our prior work [Zhang *et al.*, 2013]. The experiments included paired trials, e.g., over 15 trials (each), POMDP-1 takes 1.64 as much time as PA to move specific objects to specific places—this 39% reduction in execution time is statistically significant, with *p-value* $= 0.0023$ at 95% level of significance. A video of a robot trial can be viewed online: `http://youtu.be/8zL4R8te6wg`

5 Conclusions and Future Work

This paper describes a knowledge representation and reasoning architecture that combines the complementary strengths of declarative programming and probabilistic graphical models. The architecture's high-level (HL) and low-level (LL) system descriptions are provided using an action language, and recorded history in the HL is expanded to allow prioritized defaults. Tentative plans created in the HL are implemented in the LL using probabilistic algorithms, generating observations that add to the HL history. ASP-based inference is used to generate a multinomial prior for POMDP state estimation, and to populate a Beta PDF for metareasoning. Experimental results indicate that the architecture supports reasoning with violation of defaults, noisy observations and unreliable actions, and scales well to large, complex domains. Future work will investigate a tighter coupling between logical and probabilistic reasoning for robots collaborating with humans in complex application domains.

References

[Balduccini and Gelfond, 2003] Marcello Balduccini and Michael Gelfond. Logic Programs with Consistency-Restoring Rules. In *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, pages 9–18, 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.

[Baral, 2003] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[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.

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

[Halpern, 2003] Joseph Y. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003.

[Hanheide *et al.*, 2011] Marc Hanheide, Charles Gretton, Richard Dearden, Nick Hawes, Jeremy Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Gobelbecker, and Hendrik Zender. Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, July 16-22, 2011.

[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.

[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.

[Ong *et al.*, 2010] Sylvie C. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning Under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068, July 2010.

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

[Rosenthal and Veloso, 2012] Stephanie Rosenthal and Manuela Veloso. Mobile Robot Planning to Seek Help with Spatially Situated Tasks. In *National Conference on Artificial Intelligence*, Toronto, Canada, July 2012.

[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)*, pages 2923–2930, Chicago, USA, 2014.

[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.*, 2013] Shiqi Zhang, Mohan Sridharan, and Christian Washington. Active Visual Planning for Mobile Robot Teams using Hierarchical POMDPs. *IEEE Transactions on Robotics*, 29(4):975–985, 2013.

[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 Uncertain Worlds. *IEEE Transactions on Robotics*, 31(3):699–713, 2015.