Towards an Architecture for Knowledge Representation and Reasoning in Robotics

Shiqi Zhang[1], Mohan Sridharan[2], Michael Gelfond[3], and Jeremy Wyatt[4]

[1] Department of Computer Science, The University of Texas at Austin, USA
[2] Department of Electrical and Computer Engineering, The University of Auckland, NZ
[3] Department of Computer Science, Texas Tech University, USA
[4] School of Computer Science, University of Birmingham, UK
`szhang@cs.utexas.edu, m.sridharan@auckland.ac.nz, michael.gelfond@ttu.edu, jlw@cs.bham.ac.uk`

Abstract. This paper describes an architecture that combines the complementary strengths of probabilistic graphical models and declarative programming to enable robots to represent and reason with qualitative and quantitative descriptions of uncertainty and domain knowledge. 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, tentative plans created in the HL using commonsense reasoning are implemented in the LL using probabilistic algorithms, and the corresponding observations are added to the HL history. Tight coupling between the levels helps automate the selection of relevant variables and the generation of policies in the LL for each HL action, and supports reasoning with violation of defaults, noisy observations and unreliable actions in complex domains. The architecture is evaluated in simulation and on robots moving objects in indoor domains.

1 Introduction

Robots deployed to collaborate with humans in homes, offices, and other domains, have to represent knowledge and reason at both the sensorimotor level and the cognitive/social level. This objective maps to the fundamental challenge of representing, revising, and reasoning with qualitative and quantitative descriptions of uncertainty and incomplete domain knowledge obtained from different sources. As a significant step towards addressing this challenge, our architecture combines the knowledge representation and commonsense reasoning capabilities of declarative programming with the uncertainty modeling capabilities of probabilistic graphical models. The architecture has two tightly coupled levels with the following key features:

1. An action language is used for the system descriptions and the definition of recorded history is expanded in the high-level (HL) to allow prioritized defaults.
2. For any given objective, tentative plans are created in the HL using commonsense reasoning, and implemented in the low-level (LL) using probabilistic algorithms, with the corresponding observations adding statements to the HL history.
3. Tight coupling between the system descriptions enables automatic selection of relevant variables and the creation of action policies in the LL for any HL action.

In this paper, the HL and LL domain representations are translated into an Answer Set Prolog (ASP) program and a partially observable Markov decision process (POMDP) respectively. The novel contributions, e.g., histories with defaults and the tight coupling between the levels, support reasoning with violation of defaults, noisy observations and unreliable actions in large, complex domains. The architecture is evaluated in simulation and on robots moving objects to specific places in an indoor domain.

2 Related Work

Probabilistic graphical models such as POMDPs have been used to plan sensing, navigation and interaction for robots [13]. 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 the set of actions. Many such algorithms also do not support merging of new, unreliable information with the current beliefs in a knowledge base. ASP, a non-monotonic logic programming paradigm, is well-suited for representing and reasoning with commonsense knowledge [2]. It has been used to enable applications such as simulated robot housekeepers and natural language human-robot interaction [4,5]. However, ASP does not support probabilistic analysis, whereas a lot of information available to robots is represented probabilistically to quantitatively model the uncertainty in sensing and acting.

Researchers have designed architectures and developed algorithms that combine deterministic and probabilistic algorithms for task and motion planning on robots [8,9]. Examples of principled algorithms that combine logical and probabilistic reasoning include probabilistic first-order logic [7], Markov logic network [12], Bayesian logic [10], and a probabilistic extension to ASP [3]. 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 uncertainty and 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 addition of (probabilistic) information; and reasoning with large probabilistic components [3]. As a step towards these capabilities, our novel architecture exploits the complementary strengths of declarative programming and probabilistic graphical models, enabling robots to plan actions in larger domains than was possible before.

3 KRR Architecture

The syntax, semantics and representation of the transition diagrams of our architecture's HL and LL domain representations are described in an *action language* AL [6]. 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 allows three types of statements:

a **causes** l_{in} **if** $p_0, \dots, p_m$ (Causal law)
l **if** $p_0, \dots, p_m$ (State constraint)
impossible $a_0, \dots, a_k$ **if** $p_0, \dots, p_m$ (Executability condition)

where a is an action, l is a literal, l_{in} is an inertial fluent literal, and $p_0, \dots, p_m$ are domain literals (any domain property or its negation). A collection of statements of AL forms a system description. As an illustrative example used throughout this paper, we consider a robot that moves objects of the sorts: *textbook*, *printer* and *kitchenware*, in a domain with four places: *office*, *main_library*, *aux_library*, and *kitchen*.

3.1 HL Domain Representation

The HL domain representation consists of a system description $\mathscr{D}_H$ and histories with defaults $\mathscr{H}$. $\mathscr{D}_H$ consists of a sorted signature (Σ_H) and axioms used to describe the HL transition diagram τ_H. Σ_H defines the names of objects, functions, and predicates available for use in the HL. The sorts in our example are: *place*, *thing*, *robot*, and *object*; *object* and *robot* are subsorts of *thing*. The sort *object* has subsorts: *textbook*, *printer* and *kitchenware*. The 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 predicates 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 domain dynamics are defined using axioms that consist of causal laws such as:

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

state constraints:

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

and executability conditions such as:

$$\begin{aligned} &\textbf{impossible } move(Robot, Pl) \textbf{ if } loc(Robot, Pl) \\ &\textbf{impossible } grasp(Robot, Ob) \textbf{ if } loc(Robot, Pl_1),\ loc(Ob, Pl_2), Pl_1 \neq Pl_2 \end{aligned} \tag{3}$$

Histories with Defaults. A dynamic domain's recorded history is usually a collection of records of the form $obs(fluent, boolean, step)$, i.e., a specific fluent observed to be true or false at a given step, and $hpd(action, step)$, i.e., a specific 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 [6] for formal semantics of defaults.

Example 1 *[Example of defaults]*
Consider the following statements about the locations of textbooks in the initial state in our illustrative example. *Textbooks are typically in the main library. If a textbook is not there, it is in the auxiliary library. If a textbook is checked out, it can be found in the office.* These defaults can be represented as:

$$\begin{array}{ll} default(d_1(X)) & default(d_2(X)) \\ head(d_1(X), loc(X, main\_library)) & head(d_2(X), loc(X, aux\_library)) \\ body(d_1(X), textbook(X)) & body(d_2(X), textbook(X)) \\ & body(d_2(X), \neg loc(X, main\_library)) \end{array} \tag{4}$$

$$\begin{array}{l} default(d_3(X)) \\ head(d_3(X), loc(X, office)) \\ body(d_3(X), textbook(X)) \\ body(d_3(X), \neg loc(X, main\_library)),\ body(d_3(X), \neg loc(X, aux\_library)) \end{array} \tag{5}$$

where the literal in the "head" is true if all literals in the "body" are true. A history $\mathscr{H}_1$ with the above statements entails: $holds(loc(Tb_1, main\_library), 0)$ for textbook Tb_1. History $\mathscr{H}_2$ that adds observation: $init(loc(Tb_1, main\_library), false)$ to $\mathscr{H}_1$ renders default d_1 inapplicable; it entails: $holds(loc(Tb_1, aux\_library), 0)$ based on d_2. A history $\mathscr{H}_3$ that adds observation: $init(loc(Tb_1, aux\_library), false)$ to $\mathscr{H}_2$ should entail: $holds(loc(Tb_1, office), 0)$. History $\mathscr{H}_4$ that adds: $obs(loc(Tb_1, main\_library), false, 1)$ to $\mathscr{H}_1$ defeats default d_1 because if this default's conclusion is true in the initial state, it is also true at step 1 (by inertia), which contradicts our observation. Default d_2 will conclude that this book is initially in the $aux\_library$; the inertia axiom will propagate this information to entail: $holds(loc(Tb_1, aux\_library), 1)$.

The following terminology is used to formally define the entailment relation with respect to a fixed $\mathscr{D}_H$. A set S of literals is *closed under a default d* if S contains the head of d whenever it contains all literals from the body of d and does not contain the literal contrary to d's head. S is *closed under a constraint* of $\mathscr{D}_H$ if S contains the constraint's head whenever it contains all literals from the constraint's body. A set U of literals is the *closure of S* if $S \subseteq U$, U is closed under constraints of $\mathscr{D}_H$ and defaults of $\mathscr{H}$, and no proper subset of U satisfies these properties.

Definition 1. *[Compatible initial states]*
A state σ of τ_H is *compatible* with description $\mathscr{I}$ of the initial state of history $\mathscr{H}$ if:

1. σ satisfies all observations of $\mathscr{I}$,
2. σ contains the closure of the union of statics of $\mathscr{D}_H$ and the set $\{f : init(f, true) \in \mathscr{I}\} \cup \{\neg f : init(f, false) \in \mathscr{I}\}$.

Let $\mathscr{I}_k$ describe the initial state of history $\mathscr{H}_k$. In Example 1 above, states compatible with $\mathscr{I}_1$, $\mathscr{I}_2$, $\mathscr{I}_3$ must contain $\{loc(Tb_1, main\_library)\}$, $\{loc(Tb_1, aux\_library)\}$, and $\{loc(Tb_1, office)\}$ respectively. Since $\mathscr{I}_1 = \mathscr{I}_4$, they have the same compatible states.

Definition 2. *[Models]*
A path P of τ_H is a *model* of history $\mathscr{H}$ with description $\mathscr{I}$ of its initial state if there is a collection E of *init* statements such that:

1. If $init(f, true) \in E$ then $\neg f$ is the head of a default of $\mathscr{I}$. Similarly, for $init(f, false)$.
2. The initial state of P is compatible with the description: $\mathscr{I}_E = \mathscr{I} \cup E$.
3. Path P satisfies all observations in $\mathscr{H}$.
4. There is no collection E_0 of *init* statements which has less elements than E and satisfies the conditions above.

We refer to E as an *explanation* of $\mathscr{H}$. Models of $\mathscr{H}_1$, $\mathscr{H}_2$, and $\mathscr{H}_3$ are paths consisting of initial states compatible with $\mathscr{I}_1$, $\mathscr{I}_2$, and $\mathscr{I}_3$; the corresponding explanations are empty. For $\mathscr{H}_4$, the predicted and observed locations of Tb_1 are different. Adding $E = \{init(loc(Tb_1, main\_library), false)\}$ to $\mathscr{I}_4$ resolves this problem.

Definition 3. *[Entailment and consistency]*
- Let $\mathscr{H}^n$ be a history of length n, f be a fluent, and $i \in (0,n)$ be a step of $\mathscr{H}^n$. $\mathscr{H}^n$ *entails* a statement $Q = holds(f,i)$ ($\neg holds(f,i)$) if for every model P of $\mathscr{H}^n$, fluent literal f ($\neg f$) belongs to the ith state of P. The entailment is denoted by $\mathscr{H}^n \models Q$.
- A history which has a model is said to be *consistent*.

It can be shown that histories from Example 1 are consistent and that our definition of entailment captures the corresponding intuition.

Reasoning with HL Domain Representation. The HL domain representation is translated into a program $\Pi(\mathscr{D}_H, \mathscr{H})$ in CR-Prolog that incorporates consistency restoring rules in ASP [1,6]. 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 [2]. 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. Algorithms for computing the entailment relation of AL, and for planning and diagnostics, reduce these tasks to computing answer sets of CR-Prolog programs. Π consists of causal laws of $\mathscr{D}_H$, inertia axioms, closed world assumption for 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 default of $\mathscr{I}$ is turned into an ASP rule and a consistency-restoring (CR) rule:

$$holds(p(X),0) \leftarrow c(X), holds(b(X),0),\ not\ \neg holds(p(X),0) \quad \%\ \text{ASP rule} \qquad (6)$$
$$\neg holds(p(X),0) \overset{+}{\leftarrow} c(X), holds(b(X),0) \quad \%\ \text{CR rule}$$

The CR rule states that to restore the program's consistency, one may assume that the default's conclusion is false. See [6] for more details about CR-rules and CR-Prolog.

Proposition 1. *[Models and Answer Sets]*
A path $P = \langle \sigma_0, a_0, \sigma_1, \ldots, \sigma_{n-1}, a_n \rangle$ of τ_H is a model of history $\mathscr{H}^n$ iff there is an answer set S of a program $\Pi(\mathscr{D}_H, \mathscr{H})$ such that:
1. *A fluent $f \in \sigma_i$ iff $holds(f,i) \in S$,*
2. *A fluent literal $\neg f \in \sigma_i$ iff $\neg holds(f,i) \in S$,*
3. *An action $e \in a_i$ iff $occurs(e,i) \in S$.*

The proposition reduces: (a) 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 from $\Pi(\mathscr{D}_H, \mathscr{H})$ by adding the definition of a goal, a constraint stating that the goal must be achieved, and a rule generating possible future actions.

3.2 LL Domain Representation

The LL system description $\mathscr{D}_L$ has a sorted signature Σ_L and axioms that describe a transition diagram τ_L. Σ_L includes sorts from Σ_H and sorts *room* and *cell*, which are subsorts of *place* and whose elements satisfy static relation $part\_of(cell, room)$. A static $neighbor(cell, cell)$ describes the relation between cells. Fluents of Σ_L include those of

Σ_H, a new inertial fluent: $searched(cell, object)$—a cell was searched for an object—and two defined fluents: $found(object, place)$ and $continue\_search(room, object)$. Actions of Σ_L are viewed as HL actions represented at a higher resolution. The causal law:

$$move(Robot, Y) \textbf{ causes } \{loc(Robot, Z) : neighbor(Z, Y)\} \tag{7}$$

where Y, Z are cells, may (for instance) be used to state that moving to a cell in the LL can cause the robot to be in one of the neighboring cells. The LL includes a new action $search(cell, object)$ that enables robots to search for objects in cells; the corresponding causal laws and constraints are:

$$\begin{aligned}
&search(C, Ob) \textbf{ causes } searched(C, Ob) \qquad (8)\\
&found(Ob, C) \textbf{ if } searched(C, Ob),\ loc(Ob, C)\\
&found(Ob, R) \textbf{ if } part\_of(C, R),\ found(Ob, C)\\
&continue\_search(R, Ob) \textbf{ if } \neg found(Ob, R),\ part\_of(C, R),\ \neg searched(C, Ob)
\end{aligned}$$

The LL also has a defined fluent $failure(object, room)$ that holds iff the object under consideration is not found in the room that the robot is searching:

$$failure(Ob, R) \textbf{ if } loc(Robot, R), \neg continue\_search(R, Ob), \neg found(Ob, R) \tag{9}$$

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'.

Unlike the HL, action effects and observations in the LL are only known with some degree of probability. The function $T : S \times A \times S' \to [0, 1]$ defines the state transition probabilities in the LL. Similarly, if Z is the subset of fluents that are observable in the LL, the observation function $O : S \times Z \to [0, 1]$ defines the probability of observing specific elements of Z in specific states. Functions T and O are computed using prior knowledge, or by analyzing the effects of specific actions in specific states (Section 4.1).

Since states are partially observable in the LL, reasoning uses *belief states*, probability distributions over the set of states. Functions T and O describe a probabilistic transition diagram over belief states. The initial belief state B_0 is revised iteratively using Bayesian updates: $B_{t+1}(s_{t+1}) \propto O(s_{t+1}, o_{t+1}) \sum_s T(s, a_{t+1}, s_{t+1}) \cdot B_t(s)$. The LL system description also includes a reward specification $R : S \times A \times S' \to \Re$ that encodes the relative *utility* of specific actions in specific states. 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 a point-based approximate algorithm to compute this policy [11]. Plan execution uses the policy to repeatedly choose an action in the current belief state, and updates the belief state after executing that action and/or receiving an observation. We call this algorithm "POMDP-1".

Unlike the HL, the LL history only stores observations and actions over one time step. In this paper, the LL domain representation is translated automatically into POMDP models, i.e., data structures for $\mathscr{D}_L$'s components such that existing solvers can be used to obtain policies. One key consequence of the tight coupling between the LL and the HL is that the relevant LL variables for any HL action are identified automatically, significantly improving the efficiency of computing policies.

Algorithm 1. Control loop of the architecture

```
Input: The HL and LL domain representations, and the specific task for robot to perform.
1  LL observations reported to HL history; HL initial state ($s^H_{init}$) communicated to LL.
2  Assign goal state $s^H_{goal}$ based on task.
3  Generate HL plan(s).
4  if multiple HL plans exist then
5      Send plans to the LL, select plan with lowest (expected) action cost and communicate
       to the HL.
6  end
7  if HL plan exists then
8      for $a^H_i \in$ HL plan: $i \in [1,n]$ do
9          Pass $a^H_i$ and relevant fluents to LL.
10         Determine initial belief state over the relevant LL variables.
11         Generate LL action policy.
12         while $a^H_i$ not completed and $a^H_i$ achievable do
13             Execute an action based on LL action policy.
14             Make an observation and update belief state.
15         end
16         LL observations and action outcomes add statements to HL history.
17         if results unexpected then  Perform diagnostics in HL. ;
18         if HL plan invalid then  Replan in the HL (line 3). ;
19     end
20 end
```

3.3 Control Loop

Algorithm 1 describes the architecture's control loop. First, the LL observations obtained by the robot in the current location add statements to the HL history, and the HL initial state is communicated to the LL (line 1). The assigned task determines the HL goal state (line 2) and planning in the HL provides action sequence(s) with deterministic effects (line 3). If there are multiple HL plans, e.g., tentative plans generated for the different possible locations of a desired object, these plans are communicated to the LL; the plan with the least expected execution time is selected and communicated to the HL (lines 4-6). If an HL plan exists, actions are communicated one at a time to the LL along with the relevant fluents (line 9). For an HL action (a^H_i), the relevant LL variables are identified and the initial belief is set (line 10). An LL POMDP policy is computed (line 11) and used to execute actions and update the belief state until a^H_i is achieved or inferred to be unachievable (lines 12-15). The outcome of executing the LL policy, and the observations, add to the HL history (line 16). If the results are unexpected, diagnosis is performed in the HL (line 17); we assume that the robot can identify unexpected outcomes. If the HL plan is invalid, a new plan is generated (line 18); else, the next action in the HL plan is executed.

4 Experimental Setup and Results

This section describes the experimental setup and results of evaluating the architecture.

4.1 Experimental Setup

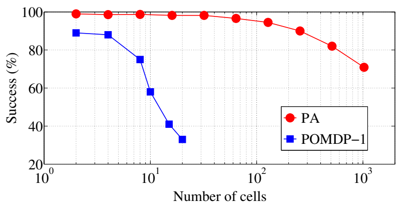
The architecture was evaluated in simulation and on physical robots. The simulator uses models that represent objects using probabilistic functions of features extracted from images, and models that reflect the robot's motion. The robot also acquired data (e.g., computational time of different algorithms) in an initial training phase to define the probabilistic components of the LL domain representation [14].

In each trial, the goal 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 *main_library* to the *office* could be: *move(Robot, main_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 (with the control loop in Algorithm 1), 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 two 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.

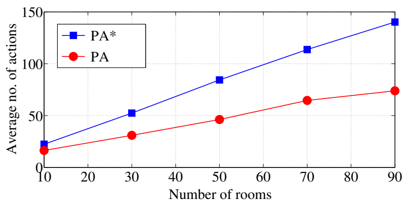
4.2 Experimental Results

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 1(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 1(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 (i.e., domain size) increases, it becomes computationally difficult to generate good POMDP action policies which, in conjunction with incorrect observations (e.g., false positives) significantly impacts the ability to complete the trials. PA focuses the robot's attention on relevant regions (e.g., specific rooms and cells). As the domain size increases, the generation of a large number of 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 domain size (i.e., 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 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 can be automatically selected using the relations in the HL system description. Furthermore, it soon becomes



(a) Success rate.



(b) Default knowledge.

Fig. 1. (a) With a limit on the time to compute policies, PA significantly increases accuracy in comparison with POMDP-1 as the number of cells increases; (b) Principled representation of defaults significantly reduces the number of actions (and thus time) for achieving assigned goal

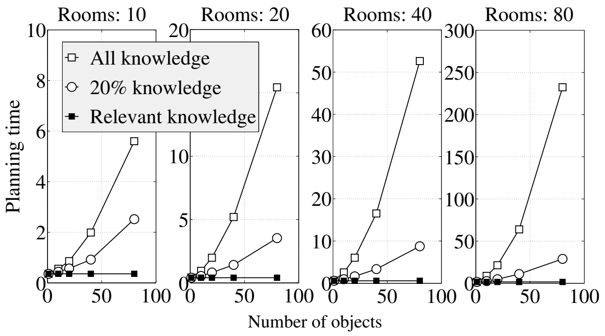


Fig. 2. Planning time as a function of the number of rooms and the number of objects in the domain—*PA* scales to larger number of rooms and objects

computationally intractable to generate a plan with POMDP-1 for domains with many objects and rooms; these results are not shown in Figure 2.

To evaluate H2, we first compared PA with *PA*∗, a version that does not include any default knowledge. Figure 1(b) 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 revise initial defaults and encode exceptions to defaults.

Finally, PA was compared with POMDP-1 on a wheeled robot over 50 trials on two floors. 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. This domain includes additional places; the map is learned and revised by the robot over time. On the third floor, we considered 15 rooms, including offices, labs, common areas and a corridor.

To use POMDP-1 in such large domains, we used a hierarchical decomposition based on our prior work [14]. 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; *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

This paper described 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 the HL definition of recorded history is expanded to allow prioritized defaults. Tentative plans created in the HL using commonsense reasoning are implemented in the LL using probabilistic algorithms, generating observations that add to the HL history. Experimental results indicate that the architecture supports reasoning at the sensorimotor level and the cognitive level with violation of defaults, noisy observations and unreliable actions, and scales well to large, complex domains. The architecture thus provides fundamental capabilities for robots assisting and collaborating with humans in complex real world application domains.

Acknowledgments. The authors thank Evgenii Balai for his help with the ASP software used in the experimental trials. This work was supported in part by the U.S. ONR Science of Autonomy Award N00014-13-1-0766 and the EC-funded Strands project FP7-IST-600623. Opinions and conclusions in this paper are those of the authors.

References

1. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: Logical Formalization of Commonsense Reasoning, AAAI SSS, pp. 9–18 (2003)
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
3. Baral, C., Gelfond, M., Rushton, N.: Probabilistic Reasoning with Answer Sets. Theory and Practice of Logic Programming 9(1), 57–144 (2009)
4. Chen, X., Xie, J., Ji, J., Sui, Z.: Toward Open Knowledge Enabling for Human-Robot Interaction. Human-Robot Interaction 1(2), 100–117 (2012)
5. Erdem, E., Aker, E., Patoglu, V.: Answer Set Programming for Collaborative Housekeeping Robotics: Representation, Reasoning, and Execution. Intelligent Service Robotics 5(4) (2012)
6. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning and the Design of Intelligent Agents. Cambridge University Press (2014)
7. Halpern, J.: Reasoning about Uncertainty. MIT Press (2003)
8. Hanheide, M., Gretton, C., Dearden, R., Hawes, N., Wyatt, J., Pronobis, A., Aydemir, A., Gobelbecker, M., Zender, H.: Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In: International Joint Conference on Artificial Intelligence (2011)

9. Kaelbling, L., Lozano-Perez, T.: Integrated Task and Motion Planning in Belief Space. International Journal of Robotics Research 32(9-10) (2013)
10. Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D.L., Kolobov, A.: BLOG: Probabilistic Models with Unknown Objects. In: Statistical Relational Learning. MIT Press (2006)
11. Ong, S.C., Png, S.W., Hsu, D., Lee, W.S.: Planning under Uncertainty for Robotic Tasks with Mixed Observability. IJRR 29(8), 1053–1068 (2010)
12. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning 62(1) (2006)
13. Rosenthal, S., Veloso, M.: Mobile Robot Planning to Seek Help with Spatially Situated Tasks. In: National Conference on Artificial Intelligence (July 2012)
14. Zhang, S., Sridharan, M., Washington, C.: Active Visual Planning for Mobile Robot Teams using Hierarchical POMDPs. IEEE Transactions on Robotics 29(4) (2013)