In *Proceedings of the 29th Conference on Artificial Intelligence* (AAAI 2015),
Austin, Texas, USA, January 2015.

CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot

Shiqi Zhang and **Peter Stone**
Department of Computer Science, The University of Texas at Austin
Austin, TX 78712 USA
{szhang,pstone}@cs.utexas.edu

Abstract

In order to be fully robust and responsive to a dynamically changing real-world environment, intelligent robots will need to engage in a variety of simultaneous reasoning modalities. In particular, in this paper we consider their needs to i) reason with commonsense knowledge, ii) model their nondeterministic action outcomes and partial observability, and iii) plan toward maximizing long-term rewards. On one hand, Answer Set Programming (ASP) is good at representing and reasoning with commonsense and default knowledge, but is ill-equipped to plan under probabilistic uncertainty. On the other hand, Partially Observable Markov Decision Processes (POMDPs) are strong at planning under uncertainty toward maximizing long-term rewards, but are not designed to incorporate commonsense knowledge and inference. This paper introduces the CORPP algorithm which combines P-log, a probabilistic extension of ASP, with POMDPs to integrate commonsense reasoning with planning under uncertainty. Our approach is fully implemented and tested on a shopping request identification problem both in simulation and on a real robot. Compared with existing approaches using P-log or POMDPs individually, we observe significant improvements in both efficiency and accuracy.

1 Introduction

Intelligent robots are becoming increasingly useful across a wide range of tasks. In real-world environments, intelligent robots need to be capable of representing and reasoning with logical and probabilistic commonsense knowledge. Additionally, due to the fundamental dynamism of the real world, intelligent robots have to be able to model and reason about quantitative uncertainties from nondeterministic action outcomes and unreliable local observations. While there are existing methods for dealing separately with either reasoning with commonsense knowledge or planning under uncertainty, to the best of our knowledge, there is no existing method that does both.

Answer Set Programming (ASP) is a non-monotonic logic programming language that is good at representing and reasoning with commonsense knowledge (Gelfond and Kahl 2014). ASP in its default form cannot reason with probabilities. A *non-monotonic probabilistic logic* (P-log) extends ASP by allowing both logical and probabilistic arguments in its reasoning (Baral, Gelfond, and Rushton 2009). However, ASP and its extensions are ill-equipped to plan toward maximizing long-term rewards under uncertainty. Partially observable Markov decision processes (POMDPs) generalize Markov decision processes (MDPs) by assuming the partial observability of underlying states (Kaelbling, Littman, and Cassandra 1998). POMDPs can model the nondeterministic state transitions and unreliable observations using probabilities, and plan toward maximizing long-term rewards under such uncertainties. However, POMDPs are not designed to reason about commonsense knowledge. Furthermore, from a practical perspective due to the computational complexity of solving POMDPs, it is necessary to limit the modeled state variables as much as possible.

This paper presents an algorithm called CORPP that stands for combining *COmmonsense Reasoning and Probabilistic Planning*. CORPP combines P-log with POMDPs to, for the first time, integrate reasoning with (logical and probabilistic) commonsense knowledge and planning under probabilistic uncertainty. The key idea is to calculate possible worlds and generate informative priors for POMDP-based planning by reasoning with logical and probabilistic commonsense knowledge. In so doing, the logical reasoning component is able to shield state variables from the POMDP that affect the priors, but that are irrelevant to the optimal policy given the prior. The proposed approach has been implemented and evaluated both in simulation and on a physical robot tasked with identifying shopping requests through spoken dialog. Results show significant improvements on both efficiency and accuracy compared to existing approaches using only P-log or POMDPs.

2 Background

This section briefly reviews the two substrate techniques used in the paper (ASP and POMDPs) and POMDP-based spoken dialog systems. In addition, the syntax of P-log, a probabilistic extension of ASP, is described.

2.1 Answer Set Programming and P-log

An ASP program can be described as a five-tuple $\langle \Theta, \mathcal{O}, \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$ of sets. These sets contain names of the *sorts*, *objects*, *functions*, *predicates*, and *variables* used in the program, respectively. Variables and object constants are *terms*. An *atom* is an expression of the form $\mathtt{p}(\bar{\mathtt{t}}) = \mathtt{true}$ or $\mathtt{a}(\bar{\mathtt{t}}) = \mathtt{y}$, where $\mathtt{p}$ is a predicate, $\mathtt{a}$ is a function, $\mathtt{y}$ is a constant from the range of $\mathtt{a}$ or a variable, and $\bar{\mathtt{t}}$ is a vector of terms. For example, `alice` is an object of sort `person`. We can define a predicate `prof` and use `prof(P)` to identify whether person `P` is a professor, where `P` is a variable.

A *literal* is an atom or its negation, where an atom's negation is of the form $\mathtt{p}(\bar{\mathtt{t}}) = \mathtt{false}$ or $\mathtt{a}(\bar{\mathtt{t}}) \neq \mathtt{y}$. In this paper, we call $\mathtt{p}(\bar{\mathtt{t}})$ and $\mathtt{a}(\bar{\mathtt{t}})$ *attributes*, if there is no variable in $\bar{\mathtt{t}}$. For instance, $\mathtt{prof(alice)} = \mathtt{true}$ is a literal and we can say the value of attribute `prof(alice)` is `true`. For simplicity's sake, we replace $\mathtt{p}(\bar{\mathtt{t}}) = \mathtt{true}$ with $\mathtt{p}(\bar{\mathtt{t}})$ and $\mathtt{p}(\bar{\mathtt{t}}) = \mathtt{false}$ with $\neg\mathtt{p}(\bar{\mathtt{t}})$ in the rest of the paper. An ASP program consists of a set of rules of the form:

$$\mathtt{l_0 \text{ or } \cdots \text{ or } l_k \leftarrow l_{k+1}, \cdots, l_m, \text{ not } l_{m+1}, \cdots, \text{ not } l_n.}$$

where `l`'s are literals. Expressions `l` and `not l` are called *extended literals*. Symbol `not` is a logical connective called *default negation*; `not l` is read as "it is not believed that `l` is true", which does not imply that `l` is believed to be false. For instance, `not prof(alice)` means it is unknown that `alice` is a professor. A rule is separated by the symbol "←". The left side is called the *head* and the right side is called the *body*. A rule is read as "head is true if body is true". A rule with an empty body is referred to as a *fact*.

Using default negation, ASP can represent (prioritized) default knowledge with different levels of exceptions. Default knowledge allows us to draw tentative conclusions by reasoning with incomplete information and commonsense knowledge. The rule below shows a simplified form of defaults that only allows *strong exceptions* that refute the default's conclusion: for object `X` of property `c`, it is believed that `X` has property `p`, if there is no evidence to the contrary.

$$\mathtt{p(X) \leftarrow c(X), \text{ not } \neg p(X).}$$

Traditionally, ASP does not explicitly quantify degrees of uncertainty: a literal is either true, false or unknown. P-log (Baral, Gelfond, and Rushton 2009) is an extension to ASP that allows *random selections* saying that if `B` holds, the value of $\mathtt{a}(\bar{\mathtt{t}})$ is selected randomly from the set $\{\mathtt{X} : \mathtt{q(X)}\} \cap \mathtt{range(a)}$, unless this value is fixed elsewhere:

$$\mathtt{random(a(\bar{t}) : \{X : q(X)\}) \leftarrow B.}$$

where `B` is a collection of extended literals; `q` is a predicate. Finally, *probability atoms* (or *pr-atoms*) are statements of the forms: $\mathtt{pr(a(\bar{t}) = y|B) = v}$, where $\mathtt{v} \in [0,1]$. The pr-atom states if `B` holds, the probability of $\mathtt{a}(\bar{\mathtt{t}}) = \mathtt{y}$ is `v`.

Reasoning with an ASP program generates a set of *possible worlds*: $\{W_0, W_1, \cdots\}$, where each is in the form of an answer set that includes a set of literals. The random selections and probability atoms enable P-log to calculate a probability for each possible world. Therefore, ASP and P-log together enable one to draw inferences regarding possible (and impossible) world states using the strong capabilities of representing and reasoning with (logical and probabilistic) commonsense knowledge. They are also useful for goal-directed planning where the problem is to find sequences of actions that lead from an initial state to possible worlds that entail a goal state. However, to the best of our knowledge, neither ASP nor P-log supports planning under uncertainty toward maximizing long-term rewards.

2.2 Partially observable MDPs

A POMDP can be described as a six-tuple $\langle \mathcal{S}, \mathcal{A}, T, Z, O, R \rangle$, where $\mathcal{S}$ defines all possible states of the world and a state is described by a set of attributes and their values; $\mathcal{A}$ is a set of actions—an action leads state transitions by changing the value(s) of domain attribute(s); $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ represents the probabilistic state transition; Z is a set of observations; $O : \mathcal{S} \times \mathcal{A} \times Z \rightarrow [0,1]$ is the observation function; and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ specifies the rewards.

Unlike MDPs, the current state can only be estimated through observations in POMDPs. A POMDP hence maintains a *belief state* (or simply *belief*), b, in the form of a probability distribution over all possible states. The belief update proceeds as follows:

$$b'(s') = \frac{O(s', a, o) \sum_{s \in \mathcal{S}} T(s, a, s') b(s)}{pr(o|a, b)} \tag{1}$$

where s, a and o represent a state, an action and an observation respectively; and $pr(o|a,b)$ is a normalizer. Solving a POMDP produces a *policy* $\pi : b \mapsto a$ that maps beliefs to actions in such a way that maximizes long-term rewards.

Thus, POMDPs enable principled decision making under uncertainty, but are ill-equipped to scale to large numbers of domain variables or reason with commonsense knowledge. Intuitively, we use ASP and P-log to represent the commonsense knowledge that includes all domain attributes, and use a POMDP to model a subset of attributes that are needed for computing the action policy for a specific task. Therefore, given a task, there can be many of the attributes that contribute to calculating the priors for the POMDP, but are irrelevant to the optimal policy of the POMDP given the prior. The next section will describe how the reasoning components shield such attributes from the POMDPs and more generally a principled approach that enables scalable commonsense reasoning and planning under uncertainty by integrating P-log with POMDPs.

2.3 POMDP-based Spoken Dialog Systems

A spoken dialog system (SDS) enables an agent to interact with a human using speech, and typically has three key components: spoken language understanding (SLU), dialog management, and natural language generation (NLG). SLU takes speech from humans and provides semantic representations to a dialog manager; the dialog manager uses the semantic representations to update its internal state s and uses a policy π to determine the next action; and NLG converts the action back into speech. Despite significant improvements in speech recognition over the past decades, it is still a challenge to reliably understand spoken language, especially in robotic domains. POMDPs have been used in dialog management to account for the uncertainties from SLU by maintaining a *distribution* (as a POMDP belief state) over all possible user meanings. Solving a POMDP problem generates a policy that maps the current belief state to an optimal action (an utterance by the system). A recent paper reviews existing POMDP-based spoken dialog systems (Young et al. 2013), and Section 5 compares such systems with our approach.

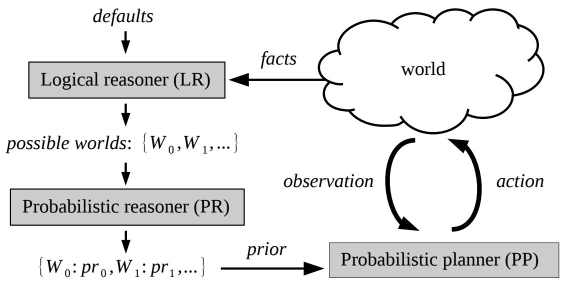

Figure 1: Overview of algorithm CORPP for combining commonsense reasoning with probabilistic planning

3 The CORPP Algorithm

Both the possible worlds and POMDP states are described using the same set of domain attributes. We say an attribute e is partially observable, if e's value can only be (unreliably) observed using sensors. The values of attributes that are not partially observable can be specified by facts, defaults, or reasoning with values of other attributes. The value of an attribute can be *unknown*. For instance, attribute *current time* can be specified by facts. Similarly, identities of people as facts can be available but not always. *Current location* (of a robot) is partially observable, because self-localization relies on sensors; and the value of attribute within *if it is within working hours now* can be inferred from *current time*.

We propose algorithm CORPP for reasoning with commonsense and planning under uncertainty, as shown in Figure 1. The *logical reasoner* (LR) includes a set of logical rules in ASP and takes defaults and facts (Section 3.1) as input. The facts are collected by querying internal memory and databases. It is possible that facts and defaults try to assign values to the same attributes, in which case, default values will be automatically overwritten by facts. The output of LR is a set of possible worlds $\{W_0, W_1, \cdots\}$. Each possible world, as an answer set, includes a set of literals that specify the values of attributes—possibly unknown.

The *probabilistic reasoner* (PR) includes a set of random selection rules and probabilistic information assignments (Section 3.2) in P-log and takes the set of possible worlds as input. Reasoning with PR associates each possible world with a probability $\{W_0 : pr_0,\ W_1 : pr_1, \cdots\}$.

Unlike LR and PR, the *probabilistic planner* (PP), in the form of a POMDP, is specified by the goal of the task and the sensing and actuating capabilities of the agent (Section 3.3). The prior in Figure 1 is in the form of a distribution and denoted by α. The ith entry in the prior, α_i, is calculated by summing up the probabilities of possible worlds that are consistent with the corresponding POMDP state s_i. In practice, α_i is calculated by sending a P-log query of this form:

$$?\{\mathtt{s_i}\}|\mathtt{obs}(\mathtt{l_0}),\cdots,\mathtt{obs}(\mathtt{l_m}),\mathtt{do}(\mathtt{l_{m+1}}),\cdots,\mathtt{do}(\mathtt{l_n}).$$

where `l`'s are facts. If a fact `l` specifies the value of a random attribute, we use `obs(l)`. Otherwise we use `do(l)`. Technically, `do(l)` adds `l` into a program before calculating the possible worlds, while `obs(l)` is used to remove the calculated possible worlds that do not include literal `l`.

Algorithm 1 The CORPP algorithm

Require: a task τ and a set of defaults $\mathcal{D}$
Require: a policy π produced by POMDP solvers
collect facts μ in the world, and add μ and $\mathcal{D}$ into LR
reason with LR and calculate possible worlds: $\mathcal{W}$
add the possible worlds into PR
for state $s_i \in \mathcal{S}$ **do**
  create a query ϕ_i using s_i and add ϕ_i into PR
  reason with PR, produce α_i, and remove ϕ_i from PR
end for
initialize belief state in PP: $b = \alpha$
repeat
  make an observation z; and update belief b using Equation 1
  select an action a using policy π
until s **is** *term*

The prior is used for initializing POMDP beliefs in PP. Afterwards, the robot interacts with the world by continually selecting an action, executing the action, and making observations in the world. A task is finished after falling into a terminating state. CORPP is summarized in Algorithm 1. We next use an illustrative problem to present more details.

Illustrative Problem: *Shopping Request Identification* In a campus environment, the shopping robot can buy an item for a person and deliver to a room, so a shopping request is in the form of $\langle$*item, room, person*$\rangle$. A person can be either a *professor* or a *student*. Registered students are authorized to use the robot and professors are not unless they paid. The robot can get access to a database to query about registration and payment information, but the database may be incomplete. The robot can initiate spoken dialog to gather information for understanding shopping requests and take a delivery action when it becomes confident in the estimation. This task is challenging for the robot because of its imperfect speech recognition ability. The goal is to identify shopping requests, e.g., $\langle$*coffee, office*1*, alice*$\rangle$, efficiently and robustly.

3.1 Logical Reasoning with ASP

Sorts and Objects: LR includes a set of sorts $\Theta : \{\mathtt{time}, \mathtt{item}, \mathtt{room}, \mathtt{person}\}$ and a set of objects $\mathcal{O}$:

$$\mathtt{time} = \{\mathtt{morning}, \mathtt{noon}, \mathtt{afternoon}\}.$$
$$\mathtt{item} = \{\mathtt{sandwich}, \mathtt{coffee}\}.$$
$$\mathtt{room} = \{\mathtt{office1}, \mathtt{office2}, \mathtt{lab}, \mathtt{conference}\}.$$
$$\mathtt{person} = \{\mathtt{alice}, \mathtt{bob}, \mathtt{carol}, \mathtt{dan}, \mathtt{erin}\}.$$

Variables: We define the set of variables $\mathcal{V} : \{\mathtt{T}, \mathtt{I}, \mathtt{R}, \mathtt{P}\}$, using a construct `#domain`, which can be interpreted by popular ASP solvers.

```
#domain time(T).    #domain item(I).
#domain room(R).    #domain person(P).
```

Predicates: The set of predicates, $\mathcal{P}$, includes:

```
place(P,R). prof(P). student(P). registered(P).
authorized(P). paid(P). task(I,R,P).
```

where `place(P,R)` represents person `P`'s working room is `R`, `authorized(P)` states P is authorized to place orders, and a ground of `task(I,R,P)` specifies a shopping request.

The following two logical reasoning rules state that professors who have paid and students who have registered are authorized to place orders.

$$\texttt{authorized(P)} \leftarrow \texttt{paid(P)},\ \texttt{prof(P)}.$$
$$\texttt{authorized(P)} \leftarrow \texttt{registered(P)},\ \texttt{student(P)}.$$

Since the database can be incomplete about the registration and payment information, we need default knowledge to reason about unspecified variables. For instance, if it is unknown that a professor has paid, we believe the professor has not; if it is unknown that a student has registered, we believe the student has not.

$$\neg\texttt{paid(P)} \leftarrow \texttt{not paid(P)},\ \texttt{prof(P)}.$$
$$\neg\texttt{registered(P)} \leftarrow \texttt{not registered(P)},\ \texttt{student(P)}.$$

ASP is strong in default reasoning in that it allows prioritized defaults and exceptions at different levels (Gelfond and Kahl 2014). LR has the Closed World Assumption (CWA) for some predicates, e.g., the below rule guarantees that the value of attribute `authorized(P)` must be either `true` or `false` (cannot be unknown):

$$\neg\texttt{authorized(P)} \leftarrow \texttt{not authorized(P)}.$$

To identify a shopping request, the robot always starts with collecting all available facts, e.g.,

```
prof(alice). prof(bob). prof(carol). student(dan).
student(erin). place(alice, office1).
place(bob, office2). place(erin, lab).
```

If the robot also observes facts of `paid(alice)`, `paid(bob)` and `registered(dan)`, reasoning with the above defaults and rules will imply that `alice`, `bob` and `dan` are authorized to place orders. Thus, LR can generate a set of possible worlds by reasoning with the rules, facts and defaults.

3.2 Probabilistic Reasoning with P-log

PR includes a set of random selection rules describing possible values of random attributes:

random(`curr_time`). `curr_time : time`.
random(`req_item(P)`). `req_item : person` $\rightarrow$ `item`.
random(`req_room(P)`). `req_room : person` $\rightarrow$ `room`.
random(`req_person`). `req_person : person`.

For instance, the second rule above states that if the delivery is for person `P`, the value of `req_item` is randomly selected from the range of `item`, unless fixed elsewhere. The following two *pr-atoms* state the probability of delivering for person `P` to `P`'s working place (0.8) and the probability of delivering `coffee` in the morning (0.8).

$$\texttt{pr(req\_room(P)} = \texttt{R} \mid \texttt{place(P,R))} = 0.8.$$
$$\texttt{pr(req\_item(P)} = \texttt{coffee} \mid \texttt{curr\_time} = \texttt{morning)} = 0.8.$$

Such random selection rules and pr-atoms allow us to represent and reason with commonsense with probabilities. Finally, a shopping request is specified as follows:

$$\texttt{task(I,R,P)} \leftarrow \texttt{req\_item(P)} = \texttt{I},\ \texttt{req\_room(P)} = \texttt{R},\ \texttt{req\_person} = \texttt{P},\ \texttt{authorized(P)}.$$

PR takes queries from PP and returns the joint probability. For instance, if it is known that Bob, as a professor, has paid and the current time is morning, a query for calculating the probability of $\langle$`sandwich`, `office1`, `alice`$\rangle$ is of the form:

$$?\{\texttt{task(sandwich,office1,alice)}\} \mid \texttt{do(paid(bob))},\ \texttt{obs(curr\_time} = \texttt{morning)}.$$

The fact of `bob` having paid increases the uncertainty in estimating the value of `req_person` by bringing additional possible worlds that include `req_person` = `bob`.

3.3 Probabilistic planning with POMDPs

A POMDP needs to model all partially observable attributes relevant to the task at hand. In the shopping request identification problem, an underlying state is composed of an item, a room and a person. The robot can ask polar questions such as "*Is this delivery for Alice?*", and wh-questions such as "*Who is this delivery for?*". The robot expects observations of "yes" or "no" after polar questions and an element from the sets of items, rooms, or persons after wh-questions. Once the robot becomes confident in the request estimation, it can take a delivery action that deterministically leads to a terminating state. Each delivery action specifies a shopping task.

- $\mathcal{S} : \mathcal{S}_i \times \mathcal{S}_r \times \mathcal{S}_p \cup$ *term* is the state set. It includes a Cartesian product of the set of items $\mathcal{S}_i$, the set of rooms $\mathcal{S}_r$, and the set of persons $\mathcal{S}_p$, and a terminal state *term*.
- $\mathcal{A} : \mathcal{A}_w \cup \mathcal{A}_p \cup \mathcal{A}_d$ is the action set. $\mathcal{A}_w = \{a_w^i, a_w^r, a_w^p\}$ includes actions of asking wh-questions. $\mathcal{A}_p = \mathcal{A}_p^i \cup \mathcal{A}_p^r \cup \mathcal{A}_p^p$ includes actions of asking polar questions, where $\mathcal{A}_p^i$, $\mathcal{A}_p^r$ and $\mathcal{A}_p^p$ are the sets of actions of asking about items, rooms and persons respectively. $\mathcal{A}_d$ includes the set of delivery actions. For $a \in \mathcal{A}_d$, we use $s \odot a$ to represent that the delivery of a matches the underlying state s (i.e., a correct delivery) and use $s \oslash a$ otherwise.
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ is the state transition function. Action $a \in \mathcal{A}_w \cup \mathcal{A}_p$ does not change the state and action $a \in \mathcal{A}_d$ results in the terminal state *term* deterministically.
- $Z : Z_i \cup Z_r \cup Z_p \cup \{z^+, z^-\}$ is the set of observations, where Z_i, Z_r and Z_p include observations of action *item*, *room* and *person* respectively. z^+ and z^- are the positive and negative observations after polar questions.
- $O : \mathcal{S} \times \mathcal{A} \times Z \rightarrow [0,1]$ is the observation function. The probabilities of O are empirically hand-coded, e.g., z^+ and z^- are more reliable than other observations. Learning the probabilities is beyond the scope of this paper.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. In our case:

$$R(s,a) = \begin{cases} -r_p, & \textbf{if } s \in \mathcal{S},\ a \in \mathcal{A}_p \\ -r_w, & \textbf{if } s \in \mathcal{S},\ a \in \mathcal{A}_w \\ -r_d^-, & \textbf{if } s \in \mathcal{S},\ a \in \mathcal{A}_d,\ s \oslash a \\ r_d^+, & \textbf{if } s \in \mathcal{S},\ a \in \mathcal{A}_d,\ s \odot a \end{cases} \quad (2)$$

  where we use r_w and r_p to specify the costs of asking wh- and polar questions. r_d^- is a big cost for an incorrect delivery and r_d^+ is a big reward for a correct one. Unless otherwise specified, $r_w = 1$, $r_p = 2$, $r_d^- = 100$, and $r_d^+ = 50$.

Consider an example where $\mathcal{S}_i = \{coffee,sandwich\}$, $\mathcal{S}_r = \{lab\}$, and $\mathcal{S}_p = \{alice,bob\}$. The state set will be specified as: $\mathcal{S} = \{coffee\_lab\_alice, \ldots, term\}$ with totally five states, where each state corresponds to a possible world specified by a set of literals (a `task` in our case), and *term* corresponds to the possible world with no `task`. The corresponding action set $\mathcal{A}$ will have 12 actions with $|\mathcal{A}_w| = 3$, $|\mathcal{A}_p| = 5$, and $|\mathcal{A}_d| = 4$. Observation set Z will be of size $|Z| = 7$ including z^+ and z^- for polar questions.

Given a POMDP, we calculate a policy using state-of-the-art POMDP solvers, e.g., APPL (Kurniawati, Hsu, and Lee 2008). The policy maps a POMDP belief to an action toward maximizing the long-term rewards. Specifically, the policy enables the robot to take a delivery action only if it is confident enough about the shopping request that the cost of asking additional questions is not worth the expected increase in confidence. The policy also decides *what*, for *whom* and *where* to deliver. There are attributes that contribute to calculating the POMDP priors but are irrelevant to the optimal policy given the prior. The reasoning components shield such attributes, e.g., `curr_time`, from the POMDPs.

CORPP enables the dialog manager (for identifying shopping requests) to combine commonsense reasoning with probabilistic planning. For instance, reasoning with the commonsense rule of "people usually buy coffee in the morning" and the fact of current time being morning, our robot prefers "Would you want to buy coffee?" to a wh-question such as "What item do you want?" in initiating a conversation. At the same time, the POMDP-based planner ensures the robustness to speech recognition errors.

4 Experiments

The CORPP algorithm has been implemented both in simulation and on real robots. The experiments in simulation focus on comparisons based on results of large numbers of simulated trials. Experimentation on real robots was more limited and results were more informal. Experiments were designed to evaluate the following hypotheses: (I) shopping requests can be efficiently and accurately identified using the POMDP-based probabilistic planner (PP); (II) combining PP with the logical reasoner (LR) improves the performance; and (III) CORPP performs the best in both accuracy and efficiency by combining LR, PR, and PP.

Defined Information Gathering Policies: We first define three straightforward policies that gather information in a pre-defined way. They serve as comparison points representing easy-to-define hand-coded policies. **Defined-1** allows the robot to take actions from $\mathcal{A}_w$; **Defined-2** allows actions from $\mathcal{A}_p$; and **Defined-3** allows actions from $\mathcal{A}_w \cup \mathcal{A}_p$. We further define a *round* as taking all allowed actions, once for each. In the end of a trial, the robot finds the shopping request (corresponding to state s) that it is most certain about and then takes action $a \in \mathcal{A}_d$ to maximize the probability of $s \odot a$ (defined in Section 3.3).

Probabilistic Knowledge at Different Levels: The robot does not necessarily have full and/or accurate probabilistic commonsense knowledge. We distinguish the probabilistic knowledge provided to the robot based on its availability and accuracy. **All**: the robot can get access to the knowledge described in Section 3.1 and 3.2 in a complete and accurate way. **Limited**: the accessibility to the knowledge is the same as "All" except that current time is hidden from the robot. **Inaccurate**: the accessibility to the knowledge is the same as "All" except that the value of current time is always wrong.

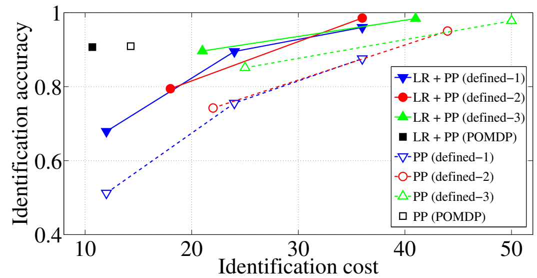

Figure 2: POMDP-based probabilistic planner (PP) performs better than the defined baseline policies in efficiency and accuracy (Hypothesis-I); and combining PP with logical reasoner (LR) further improves the performance (Hypothesis-II).

4.1 Simulation Experiments

All three hypotheses were evaluated extensively in simulation, where action costs are defined in Section 3.3. Each data point in the figures in this section is the average of at least 10,000 simulated trials.

To evaluate Hypothesis-I (POMDP-based PP), we compared the POMDP-based probabilistic planner against the three defined information gathering policies—all start with uniform α meaning that *all* worlds are equally probable. The defined policies can gather information by taking multiple rounds of actions. The results are shown as the *hollow markers* in Figure 2. The POMDP-based PP enables the shopping requests to be correctly identified in more than 90% of the trials with costs of about 14.3 units on average (black hollow square) with the imperfect sensing ability. In contrast, the defined policies need more cost (e.g., about 44 units for Defined-2) to achieve comparable accuracy (red hollow circle). Therefore, POMDP-based planning enables efficient and accurate information gathering and behavior in identifying shopping requests.

To evaluate Hypothesis-II (LR and PP), the POMDP-based PP and the three defined policies are next combined with LR that incorporates the logical reasoning rules, defaults, and facts (Section 3.1)—results are shown as the *solid markers* in Figure 2. Without PR, we can only assume all logically possible worlds to be equally probable in calculating the prior α. We can see the combination of LR and the POMDP-based PP performs better than the combination of LR and the three defined planning policies—see the *solid markers*. Furthermore, comparing to the corresponding hollow markers, we can see adding LR improves the performance of both PP and the defined policies. Specifically, LR enables the POMDP-based PP to reduce the average cost to about 10.5 units without hurting the accuracy. LR reduces

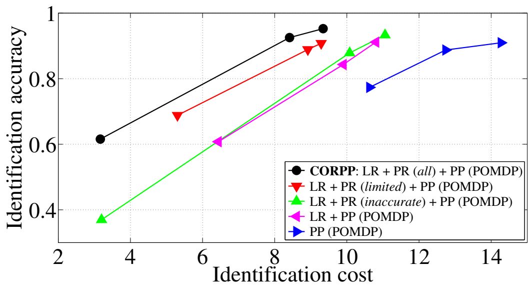

Figure 3: CORPP performs better than the other approaches in both efficiency and accuracy (Hypothesis-III).

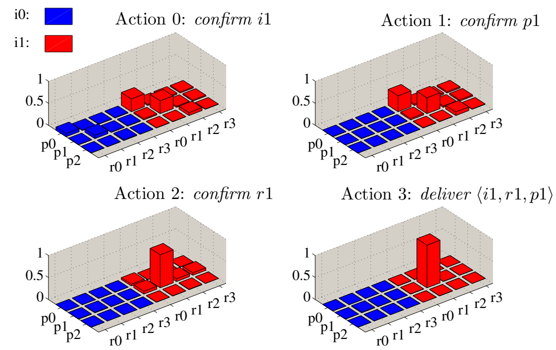

Figure 5: Belief change in an illustrative trial.

the number of possible worlds for PP (from 40 to 24 in our case), which enables POMDP solvers to calculate more accurate action policies in reasonable time (an hour in our case) and reduces the uncertainty in state estimation.

To evaluate Hypothesis-III (LR, PR and PP), we provide the probabilistic commonsense knowledge (Section 3.2) to the robot at different completeness and accuracy levels—learning the probabilities is beyond the scope of this paper. Experimental results are shown in Figure 3. Each set of experiments has three data points because we assigned different penalties to incorrect identifications in PP (r_d^- equals 10, 60 and 100). Generally, a larger penalty requires the robot to ask more questions before taking a delivery action. POMDP-based PP without commonsense reasoning (blue rightward triangle) produced the worst results. Combining LR with PP (magenta leftward triangle) improves the performance by reducing the number of possible worlds. Adding *inaccurate* probabilistic commonsense in PR (green upward triangle) hurts the accuracy significantly when the penalty of incorrect identifications is small. Reasoning with *limited* probabilistic commonsense in PR requires much less cost and results in higher (or at least similar) accuracy on average, compared to planning without PR. Finally, the proposed algorithm, CORPP, produced the best performance in both efficiency and accuracy. We also find that the POMDP-based PP enables the robot to recover from inaccurate knowledge by actively gathering more information—compare the right ends of the "*limited*" and "*inaccurate*" curves.

For completeness, we evaluated the performance of pure reasoning (LR and PR): information gathering actions are not allowed; the robot uses all knowledge to determine the most likely shopping request (in case of a tie, it randomly chooses one from the most likely ones). The reasoning takes essentially no time and the average accuracy is only 0.193.

4.2 Robot Experiments

We have also implemented the proposed approach on a physical robot shown in Figure 4. The robot is built on top of a Segway Robotic Mobility Platform. It uses a Hokuyo URG-04LX laser rangefinder and a Kinect RGB-D camera for navigation and sensing, and Sphinx-4 (Walker et al. 2004) for speech recognition. The software modules run in a Robot Operating System (ROS) (Quigley et al. 2009). After the proposed approach determines the parameters of the shopping request, it is passed to a hierarchical task planner for creating a sequence of primitive actions that can be directly executed by the robot (Zhang et al. 2014b).

Figure 4: Robot platform (SegBot) used in experiments.

We present the belief change in an illustrative trial in Figure 5, where i, r and p are item, room and person respectively. $i0$ is sandwich and $i1$ is coffee. The robot first read its internal memory and collected a set of facts such as the current time was "morning", $p0$'s office is $r0$ and $p1$'s office is $r1$. Reasoning with commonsense produced a prior shown in the top-left of Figure 5, where the most possible two requests were $\langle i1, r0, p0 \rangle$ and $\langle i1, r1, p1 \rangle$. The robot took the first action to confirm the item to be coffee. After observing a "*yes*", the robot further confirmed $p1$ and $r1$. Finally, it became confident in the estimation and successfully identified the shopping request. Therefore, reasoning with domain knowledge produced an informative prior, based on which the robot could directly focus on the most likely attribute values, and ask corresponding questions. In contrast, when starting from a uniform prior, the robot would have needed at least six actions before the delivery action. A demo video is available by this link: `http://youtu.be/2UJG4-ejVww`

5 Related Work

Logical and probabilistic reasoning: Researchers have developed algorithms and frameworks that combine logical and probabilistic reasoning, e.g., probabilistic first-order logic (Halpern 2003), Markov logic network (Richardson and Domingos 2006), and Bayesian logic (Milch et al. 2005). However, algorithms based on first-order logic for probabilistic reasoning have difficulties in representing or reasoning with commonsense knowledge. P-log (Baral, Gelfond, and Rushton 2009) can do logical and probabilistic reasoning with commonsense but has difficulties to plan toward maximizing long-term rewards.

Planning with POMDPs: POMDPs have been applied to a variety of probabilistic planning tasks (Young et al. 2013; Zhang, Sridharan, and Washington 2013). However, existing POMDP-based planning work does not readily support representation of or reasoning with rich commonsense knowledge. Furthermore, from a practical perspective, the state variables modeled by POMDPs have to be limited to allow real-time operation. This makes it challenging to use POMDPs in large, complex state-action spaces, even if hierarchical decomposition and approximate algorithms have been applied (Zhang, Sridharan, and Washington 2013; Kurniawati, Hsu, and Lee 2008).

The illustrative example problem in this paper is a Spoken dialog system (SDS). POMDP-based SDSs have been shown to be more robust and efficient than traditional work using deterministic flowchart-based action policies (Roy, Pineau, and Thrun 2000). A recent paper reviews existing techniques and applications of POMDP-based SDSs (Young et al. 2013). Similar to other POMDP-based applications, such SDSs are ill-equipped to represent and reason with commonsense knowledge.

Combining ASP with POMDPs: Existing work investigated generating priors by inference with domain knowledge using ASP for POMDP-based planning (Zhang, Sridharan, and Bao 2012). However, this work did not have a probabilistic reasoner to reason with probabilistic commonsense knowledge. Furthermore, the logical reasoner in that work did not calculate possible worlds for POMDPs. An action language has been combined with POMDP-based planning toward reasoning with qualitative and quantitative domain descriptions (Zhang et al. 2014a). The use of the action language limits the capability of reasoning with commonsense knowledge in that work.

6 Conclusions and Future Work

This paper presents the CORPP algorithm that, for the first time, integrates reasoning with (logical and probabilistic) commonsense knowledge and planning under probabilistic uncertainty. Answer Set Programming, a non-monotonic logic programming language, is used to reason with *logical* commonsense knowledge. P-log, a probabilistic extension of ASP, further enables reasoning with *probabilistic* commonsense knowledge. POMDPs are used to plan under uncertainty toward maximizing long-term rewards. The complementary features of ASP and POMDPs ensure efficient, robust information gathering and behavior in robotics. Experimental results on a shopping request identification problem show significant improvements on both efficiency and accuracy, compared with existing approaches using P-log or POMDPs individually.

One direction of future investigation is to apply CORPP to problems of larger scales by programming with P-log using SCOU grammar (more strict but not fully declarative) (Zhu 2012) and using N-best and/or factorized POMDPs (Young et al. 2013). Another possible direction is to apply CORPP to a team of intelligent robots with a shared knowledge base.

7 Acknowledgments

This research has taken place in the Learning Agents Research Group (LARG) at the AI Laboratory, University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), Office of Naval Research (21C184-01), and Yujin Robot. LARG research is also supported in part through the Freshman Research Initiative (FRI), College of Natural Sciences, University of Texas at Austin.

References

Baral, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming* 9(1):57–144.

Gelfond, M., and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach.* Cambridge University Press.

Halpern, J. Y. 2003. Reasoning about uncertainty.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101:99–134.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*.

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2005. BLOG: probabilistic models with unknown objects. In *Proceedings of the 19th international joint conference on Artificial intelligence*.

Quigley, M.; Conley, K.; Gerkey, B. P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: An Open-Source Robot Operating System. In *Open Source Software Workshop*.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Roy, N.; Pineau, J.; and Thrun, S. 2000. Spoken dialogue management using probabilistic reasoning. In *the Annual Meeting on Association for Computational Linguistics*.

Walker, W.; Lamere, P.; Kwok, P.; Raj, B.; Singh, R.; Gouvea, E.; Wolf, P.; and Woelfel, J. 2004. Sphinx-4: A flexible open source framework for speech recognition.

Young, S.; Gasic, M.; Thomson, B.; and Williams, J. D. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Zhang, S.; Sridharan, M.; Gelfond, M.; and Wyatt, J. 2014a. Towards an architecture for knowledge representation and reasoning in robotics. In *Social Robotics*. Springer. 400–410.

Zhang, S.; Yang, F.; Khandelwal, P.; and Stone, P. 2014b. Mobile robot planning using action language $\mathcal{BC}$ with hierarchical domain abstractions. In *The 7th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)*.

Zhang, S.; Sridharan, M.; and Bao, F. S. 2012. ASP+POMDP: Integrating Non-monotonic Logic Programming and Probabilistic Planning on Robots. In *International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*.

Zhang, S.; Sridharan, M.; and Washington, C. 2013. Active visual planning for mobile robot teams using hierarchical pomdps. In *IEEE Transactions on Robotics*.

Zhu, W. 2012. *PLOG: Its Algorithms and Applications*. Ph.D. Dissertation, Texas Tech University, USA.