Representing and Reasoning with Logical and Probabilistic Knowledge on Robots

Sridharan, Mohan; Gelfond, Michael

Representing and Reasoning with Logical and Probabilistic Knowledge on Robots

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

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

Abstract

This paper describes an architecture that enables robots to represent and reason with logic-based and probabilistic descriptions of uncertainty and domain knowledge. An action language is used to describe tightly-coupled transition diagrams of the domain at two different resolutions. For any given goal, reasoning with commonsense knowledge at coarse-resolution provides a plan of abstract actions. To implement each such abstract action, the relevant part of the fine-resolution transition diagram is identified, and used with a probabilistic representation of the uncertainty in sensing and actuation, to obtain a sequence of concrete actions. The outcomes of executing these concrete actions are used for subsequent coarse-resolution reasoning. We illustrate the architecture's capabilities using examples of a mobile robot finding and moving objects in an indoor domain.

1 Introduction

Robots assisting humans often need to represent and reason with incomplete and uncertain knowledge at different levels of abstraction. Close to the sensorimotor level, information is typically represented probabilistically to quantitatively model the uncertainty in sensing and actuation ("the robotics book is on the shelf with probability 0.9"). Robots also benefit from reasoning with more abstract commonsense knowledge, including knowledge that holds in all but a few exceptional situations ("books are typically in the library") and may not be easy to represent probabilistically. One open problem in robotics and artificial intelligence is to represent and reason with these two types of knowledge. As a step towards addressing this problem, our prior paper described an architecture that coupled non-monotonic logical reasoning and probabilistic reasoning [Zhang *et al.*, 2014]. Here, we view the architecture's reasoning system as an interplay between a logician and a statistician. The former has an abstract, coarse-resolution view of the world, perceiving an office domain, for instance, as a collection of connected rooms containing different types of objects, with the robot able to move between rooms and find objects. If the goal is to find a specific book, the logician provides a plan of abstract actions that may direct the robot to go to the library, where books are normally stored. The first action of this plan, e.g., "*move to the lab*" that is the first room on the way to the library, will be passed to a statistician, who has a different world view. The statistician believes the office domain has grid cells in different rooms, and although the robot is capable of moving to a neighboring cell and checking a cell for a target object, these actions only succeed with some probability that is known or learned over time. To move to the lab, the statistician has the robot move from cell to cell, observe its position, and revise its belief of its position. If the statistician has a sufficiently high confidence that the robot has moved to the lab, this information is reported back to the logician—otherwise, failure is reported after some time, and the logician has to diagnose and replan. Even this simple scenario shows that a robot needs the ability to represent and manipulate both logical and probabilistic knowledge. *This paper reports our experience in the design of such a robot using knowledge representation and reasoning tools tailored towards different reasoning tasks*. In comparison with our prior work [Zhang *et al.*, 2014], we introduce precise (and some new) definitions of basic notions used to build the domain's mathematical models—these contributions are also described in [Sridharan and Gelfond, 2016].

We used action description language AL_d [Gelfond and Inclezan, 2013] to describe the state transition diagram modeling possible trajectories of the domain. AL_d supports a concise representation for complex relationships, and for recorded histories of actions and observations. We expanded the standard notion of recorded history to include default knowledge about the initial state. Other action languages that allow causal laws specifying default values of fluents at arbitrary time steps [Lee *et al.*, 2013] occasionally pose difficulties with representing exceptions to such defaults when the domain is expanded. Reasoning with theories in AL_d is performed by reducing planning, diagnostics and other tasks to computing answer sets of a program in CR-Prolog, a variant of Answer Set Prolog (ASP) that allows us to represent and reason with defaults and their exceptions [Balduccini and Gelfond, 2003]. This *reduction, which considers histories with initial state defaults*, is the first novel contribution—it builds on the close relationship between action languages and logic programming with answer set semantics. Existing solvers help automate the reasoning [Leone *et al.*, 2006].

The second novel contribution is the *precise definition of the statistician's world view as a refinement of the logician's transition diagram*. The new diagram is viewed as the result of increasing the resolution of the robot's ability to see the world, e.g., it includes newly discovered cells in rooms, and relations and actions involving these cells. Although this refinement may require some domain knowledge, our novel mathematical definition includes axioms that precisely establish the relationship between the coarse-resolution and fine-resolution diagrams, and describe the effects of observations. Next, the fine-resolution description is *randomized*, i.e., modified to consider the non-deterministic effects of fine-resolution actions and observations. The third novel contribution is to *expand AL_d by a construct that supports the representation of such effects*. Probabilities are then associated with the purely logical diagram to obtain a probabilistic diagram. Reasoning in this new diagram also considers *belief states*, i.e., probability distributions over states of the logical diagram. Theoretically, the statistician can now use probabilistic graphical models such as a partially observable Markov decision process (POMDP) to select and execute the "best" action, make observations, and update the belief state until an abstract action provided by the logician is completed with high probability, or all hope of doing so is lost. However, POMDP algorithms consider all possible combinations of physical states and actions, which can become intractable even for a comparatively small domain. To avoid this problem, the robot *zooms* to the part of the fine-resolution diagram that is relevant to the execution of the coarse-resolution action provided by the logician. For instance, to "move from room R_1 to room R_2", domain properties and actions related to other rooms and objects are eliminated, dramatically reducing the size of the corresponding diagram. The fourth contribution is a *precise definition of zooming, which helps automate this process*. Finally, this zoomed part is represented in the format suitable for use with POMDP solvers. The corresponding policy is invoked to execute a sequence of concrete actions that implements the abstract action, with the outcomes added to the coarse-resolution history. We use the following example to demonstrate these contributions and design steps.

Example 1. *[Office Domain]* Consider a robot assigned the goal of moving specific objects to specific places in an office domain. This domain contains:

- The sorts: *place*, *thing*, *robot*, and *object*, with *object* and *robot* being subsorts of *thing*. Sorts *textbook*, *printer* and *kitchenware*, are subsorts of the sort *object*.
- Places: *office*, *main_library*, *aux_library*, and *kitchen*, some of which are directly accessible from each other.
- An instance of the sort *robot*, called rob_1, and a number of instances of subsorts of the sort *object*.

This domain illustrates many of the representation, reasoning, perception, and actuation challenges that exist in more complex robotics domains.

2 Related Work

Logic-based representations and probabilistic graphical models have been used to control sensing, navigation and interaction for robots [Bai *et al.*, 2014; Hawes *et al.*, 2010]. Formulations based on probabilistic representations (by themselves) make it difficult to perform commonsense reasoning, whereas approaches based on logic programming tend to require considerable prior knowledge of the domain and the agent's capabilities, and make it difficult to merge new, unreliable information with an existing knowledge base. Theories of reasoning about actions and change, and the non-monotonic logical reasoning ability of ASP, have been used by an international research community, e.g., for natural language human-robot interaction [Chen *et al.*, 2012], control of unmanned aerial vehicles [Balduccini *et al.*, 2014], and coordination of robot teams [Saribatur *et al.*, 2014]. However, the basic version of ASP does not support probabilistic representation of uncertainty, whereas a lot of information extracted from sensors and actuators is represented probabilistically.

Researchers have designed architectures for robots that combine logic-based and probabilistic algorithms for task and motion planning [Kaelbling and Lozano-Perez, 2013], couple declarative programming and continuous-time planners for path planning in teams [Saribatur *et al.*, 2014], combine a probabilistic extension of ASP with POMDPs for human-robot dialog [Zhang and Stone, 2015], combine logic programming and reinforcement learning to discover domain axioms [Sridharan *et al.*, 2016], or use a three-layered organization for representing knowledge and reasoning with first-order logic and probabilities in open worlds [Hanheide *et al.*, 2015]. Some general formulations that combine logical and probabilistic reasoning include Markov logic network [Richardson and Domingos, 2006], Bayesian logic [Milch *et al.*, 2006], and probabilistic extensions to ASP [Baral *et al.*, 2009; Lee and Wang, 2015]. However, algorithms based on first-order logic do not support non-monotonic logical reasoning and do not provide the desired expressiveness—it is not always possible to associate numbers with logic statements to express degrees of belief. Algorithms based on logic programming do not support one or more of the desired capabilities such as incremental revision of (probabilistic) information, and reasoning with large probabilistic components. As a step towards addressing these limitations, we have developed architectures that couple declarative programming and probabilistic graphical models [Zhang *et al.*, 2014; 2015]. Here, we expand on our prior work to describe the systematic construction of robots with the desired knowledge representation and reasoning capabilities.

3 Logician's Description

We start with the logician's view of the world.

Action Language AL_d: The logician's state transition diagram is specified as a system description (theory) in a variant of action language AL_d, which allows statements of the form:

$$a \textbf{ causes } f(\bar{x}) = y \textbf{ if } body$$
$$f(\bar{x}) = y \textbf{ if } body$$
$$impossible\ a_0, \ldots, a_n \textbf{ if } body$$

The first statement describes an action's direct effect—if action a is executed in a state satisfying $body$, the value of fluent f in the resulting state would be y. For instance:

$$move(R, Pl) \textbf{ causes } loc(R) = Pl$$

says that robot R moving to place Pl will end up in Pl. The second statement, a state constraint, says that $f(\bar{x}) = y$ in any state that satisfies statements in $body$. For instance:

$$loc(Ob) = Pl \ \textbf{if} \ loc(R) = Pl, \ in\_hand(R, Ob)$$

requires that the object grasped by a robot share the robot's location. The third statement prohibits simultaneous execution of actions $a_0, \ldots, a_n$ in a state satisfying $body$. The functions in these statements are of two types, those whose values can be changed by actions (*fluents*) and those whose values cannot be changed (*statics*). Fluents can be *basic* or *defined*; the former is subject to inertia laws while the latter is defined in terms of other fluents. Formal semantics of AL_d is discussed at length in [Gelfond and Kahl, 2014]. Unfortunately, it only allows boolean fluents, a restriction that is removed by our variant of the language (details below).

Histories with defaults: The logician's knowledge also contains a *recorded history*, a collection of the robot's observations and actions. This recorded history typically consists of statements of the form $obs(f, y, true, i)$ (or $obs(f, y, false, i)$)—at step i of the robot's trajectory, the value of fluent f is observed to be (or not to be) y; and $hpd(a, i)$—action a was successfully executed (i.e., happened) at step i. The recorded history defines a collection of *models*, trajectories of the system compatible with this record. The robot can benefit substantially from some forms of default knowledge, e.g., it can be told that books are normally kept in the library. To represent such knowledge, we introduced an additional type of historical record:

$$\textbf{initial default} \ f(\bar{x}) = y \ \textbf{if} \ body$$

to assume that in the initial state that satisfies $body$, the value of $f(\bar{x})$ is y. For instance, the following statements:

$$\textbf{initial default} \ loc(X) = main\_library \ \textbf{if} \ textbook(X)$$
$$\textbf{initial default} \ loc(X) = office \ \textbf{if} \ textbook(X), \ loc(X) \neq main\_library.$$

encode the most likely and the second most likely location for a textbook. Such defaults may substantially simplify the logician's planning, e.g., the plan for finding textbook tb will consist of going directly to the library and looking there. Defaults are also useful for diagnostics, e.g., if tb is not found in the library, the robot would realize that this observation defeats the first default, try the second default, and look in the office. To ensure such behavior, we redefined the notion of a model of a history, and designed a new algorithm for computing such models. Given an AL_d theory $\mathscr{D}$ and history $\mathscr{H}$, we construct program $\Pi(\mathscr{D}, \mathscr{H})$ with the encoding of $\mathscr{D}$ in ASP, and the atoms representing observations and actions of $\mathscr{H}$ with initial state defaults. This encoding supports indirect exceptions to defaults (like observation above) and uses CR-Prolog, which justifies our departure from standard ASP.

Planning and diagnostics: The description of the logician's knowledge, as provided above, is sufficient for adequately performing planning and diagnostics, which are reduced to computing answer sets of $\Pi(\mathscr{D}, \mathscr{H})$ with a standard encoding of the robot's goal. We use an efficient ASP solver, SPARC, which expands CR-Prolog and provides explicit constructs to specify objects, relations, and their sorts [Balai *et al.*, 2013]. Atoms of the form $occurs(action, step)$ in the answer set obtained by solving $\Pi(\mathscr{D}, \mathscr{H})$, represent the shortest sequence of abstract actions for achieving the logician's goal. Prior research results in the theory of action languages and ASP ensure that the plan is provably correct. Suitable atoms in the answer set can also be used for diagnostics.

Example 2. *[Logician's view of the world]*
The logician's system description $\mathscr{D}_H$ of the domain in Example 1 consists of sorted signature Σ_H and axioms describing transition diagram τ_H. Σ_H defines the names of objects and functions available for use, e.g., the sorts $place$, $thing$, $robot$, and $object$, with $object$ and $robot$ being subsorts of $thing$. The statics include the relation $next\_to(place, place)$, which describes if two places are next to each other. The domain's fluents are $loc : thing \rightarrow place$ and $in\_hand : robot \times object \rightarrow boolean$—these are basic fluents. The domain's actions are $move(robot, place)$, $grasp(robot, object)$, and $putdown(robot, object)$. The domain dynamics are defined using causal laws:

$$move(R, Pl) \ \textbf{causes} \ loc(R) = Pl$$
$$grasp(R, Ob) \ \textbf{causes} \ in\_hand(R, Ob)$$
$$putdown(R, Ob) \ \textbf{causes} \ \neg in\_hand(R, Ob)$$

state constraint:

$$loc(Ob) = Pl \ \textbf{if} \ loc(R) = Pl, \ in\_hand(R, Ob)$$

and executability conditions such as:

$$\textbf{impossible} \ move(R, Pl_2) \ \textbf{if} \ loc(R) = Pl_1, \neg next\_to(Pl_1, Pl_2)$$
$$\textbf{impossible} \ grasp(R, Ob) \ \textbf{if} \ loc(R) \neq loc(Ob)$$
$$\textbf{impossible} \ putdown(R, Ob) \ \textbf{if} \ \neg in\_hand(R, Ob)$$

For any given domain, the part of Σ_H described so far is unlikely to change substantially. However, the last step in the constructions of Σ_H, which populates the basic sorts with specific objects, e.g $robot = \{rob_1\}$ and $place = \{r_1, \ldots, r_n\}$, is likely to undergo frequent revisions. Ground instances of axioms are obtained by replacing variables by ground terms from the corresponding sorts.

In τ_H, actions are assumed to be deterministic, and values of fluents are assumed to be observable, which supports fast, tentative planning and diagnostics for achieving the goals. This domain representation is ideally tested extensively by including various recorded histories and using the resulting programs to solve various reasoning tasks.

4 Statistician's Description

Next, we describe our design of the statistician.

Refinement: We begin with the *deterministic* version of the statistician's world, given by system description $\mathscr{D}_L$ defining a transition diagram τ_L that is a (deterministic) *refinement* of the logician's diagram τ_H—τ_L is the result of magnifying some objects in signature Σ_H of system description $\mathscr{D}_H$ of τ_H. Newly discovered parts of the magnified objects are called the refined *components*, e.g., every room is magnified and viewed as a collection of its component cells. We say that a signature Σ_L *refines* signature Σ_H if it is obtained by:

- Replacing every basic sort st_H of Σ_H, whose elements were magnified, by its coarse-resolution version $st_L^* = st_H$, and fine-resolution *counterpart* $st_L = \{o_1, \ldots, o_m\}$ consisting of the components of magnified elements of st_H, e.g., replace sort $place = \{r_1, \ldots, r_n\}$ of rooms in Σ_H by:

$$place^* = \{r_1, \ldots, r_n\}$$
$$place = \{c_1, \ldots, c_m\}$$

  where the rooms are magnified to discover component cells $c_1, \ldots, c_m$.
- Introducing static relation *component* between magnified objects from st_L^* and the newly-discovered objects from st_L, e.g., $component(c, r)$ is true iff cell c is in room r.
- Replacing sort $fluent$ of Σ_H by its coarse-resolution copy $fluent^*$ (a defined fluent) and its fine-resolution counterpart $fluent$ (a basic fluent). For instance:

$$loc^* : thing \rightarrow place^*$$
$$loc : thing \rightarrow place$$

  Other fluents (and their signatures) are unchanged.
- Obtaining actions of Σ_L by replacing the magnified parameters of actions from Σ_H by their fine-resolution counterparts. For instance, Σ_L includes original actions *grasp* and *putdown*, and new action $move(robot, cell)$. Σ_L also has knowledge-producing action $test(robot, fluent, value)$ that activates algorithms to check the value of an observable fluent of $\mathscr{D}_L$ in a given state, e.g., $test(R, loc(Th), Cell)$. We also add fluents to describe the result of testing, e.g., $observed(robot, fluent, value)$ is true if the most recent (direct or indirect) observation of $fluent$ returned *value*.

Axioms of $\mathscr{D}_L$ include axioms of $\mathscr{D}_H$, and domain-dependent axioms relating coarse-resolution fluents and their fine-resolution counterparts, e.g., in our example domain:

$$loc^*(Th) = Rm \textbf{ if } component(Cl, Rm),\ loc(Th) = Cl$$

where we define static $component(c, r)$ for every cell c within room r—also, static $next\_to(c_1, c_2)$ holds for adjacent cells not separated by obstacles. More importantly, $\mathscr{D}_L$ has basic knowledge fluents that model the direct and indirect knowledge effects of sensing, and introduce axioms relating these fluents and the *test* actions. For instance:

$$test(R, F, Y) \textbf{ causes } dir\_obs(R, F, Y) = true \textbf{ if } F = Y.$$

In our example, robot rob_1 testing cell c for object o will make basic knowledge fluent $dir\_obs(rob_1, loc(o), c)$ true if the object is indeed there—the fluent has three possible values, *true*, *false* and *undef*, and is initially set to the third value. Another fluent, $indir\_obs(rob_1, loc(o), R)$ holds if $loc(o)$ is observed to be true in some component cell of room R. The fluent's value is *observed* if it is observed directly or indirectly. It is the responsibility of the designers to make sure that the transition diagram τ_L specified by our system description $\mathscr{D}_L$ matches the following formal definitions of refinement.

Definition 1. *[Refinement of a state]*
A state δ of τ_L is said to be a *refinement* of state σ of τ_H if:

- For every magnified fluent f from the signature of Σ_H:

$$f(x) = y \in \sigma \textbf{ iff } f^*(x) = y \in \delta$$

- For every other fluent of Σ_H:

$$f(x) = y \in \sigma \textbf{ iff } f(x) = y \in \delta$$

Definition 2. *[Refinement of a system description]*
Let $\mathscr{D}_L$ and $\mathscr{D}_H$ be system descriptions with transition diagrams τ_L and τ_H respectively. $\mathscr{D}_L$ is a *refinement* of $\mathscr{D}_H$ if:

- States of τ_L are the refinements of states of τ_H.
- For every transition $\langle \sigma_1, a^H, \sigma_2 \rangle$ of τ_H, every fluent f in a set F of simultaneously observable fluents, and every refinement δ_1 of σ_1, there is a path P in τ_L from δ_1 to a refinement δ_2 of σ_2 such that:
  - Every action of P is executed by the robot which executes a^H.
  - Every state of P is a refinement of σ_1 or σ_2, i.e., no unrelated fluents are changed.
  - $observed(R, f, Y) = true \in \delta_2$ if $(f = Y) \in \delta_2$ and $observed(R, f, Y) = false \in \delta_2$ if $(f = Y) \notin \delta_2$.

Randomization: Next, to expand $\mathscr{D}_L$ to capture the non-determinism in action execution and observations on the robot, we *extended AL_d by non-deterministic causal laws*:

$$a \textbf{ causes } f(\bar{x}) \ :\ \{Y : p(Y)\} \textbf{ if } body$$
$$a \textbf{ causes } f(\bar{x}) \ :\ sort\_name \textbf{ if } body$$

The first statement says that if action a is executed in a state satisfying *body*, f may take on any value from the set $\{Y : p(Y)\} \cap range(f)$ in the resulting state—the second statement says that f may take any value from $\{sort\_name \cap range(f)\}$. The *randomized* fine-resolution description $\mathscr{D}_{LR}$ is obtained by replacing each action's deterministic causal laws in $\mathscr{D}_L$ by non-deterministic ones, declaring the affected fluent as a random fluent. For instance, the new causal law for *move* is:

$$move(R, C_2) \textbf{ causes } loc(R) = \{C : range(loc(R), C)\}$$

where defined fluent *range* is given by:

$$range(loc(R), C) \textbf{ if } loc(R) = C$$
$$range(loc(R), C) \textbf{ if } loc(R) = C_1,\ next\_to(C, C_1)$$

where a robot moving to a cell can end up in any cell within range, e.g., the current cell and its immediate neighbors.

To complete the statistician's model with probabilistic information, we run experiments that sample specific instances of each ground non-deterministic causal law in $\mathscr{D}_{LR}$, have the robot execute the corresponding action multiple times, and collect statistics (e.g., counts) of the number of times each outcome of the corresponding fluent is obtained. These *statistics are collected in an initial training phase*, and used to compute causal probabilities of action outcomes and the probability of observations being correct. Local symmetry assumptions are used to simplify this collection of statistics, e.g., movement from a cell to one of its neighbors is assumed to be the same for any cell—the designer provides the required domain-specific information. In our example, if rob_1 in cell c_1 may reach $\{c_1, c_2, c_3\}$ when executing $move(rob_1, c_2)$, the probabilities of these outcomes may be 0.1, 0.8, and 0.1 respectively. Similarly, 0.85 may be the probability with which a specific object can be recognized in a specific cell. Any prior beliefs about these probabilities

(e.g., from a human) are used as initial beliefs that are revised by the experiments.

Zooming: The statistician uses $\mathscr{D}_{LR}$ and the computed probabilities for fine-resolution execution of the transition $T = \langle \sigma_1, a^H, \sigma_2 \rangle \in \tau_H$. Since reasoning probabilistically about all of $\mathscr{D}_{LR}$ may result in incorrect behavior and be computationally intractable, the statistician identifies the part $\tau_{LR}(T)$ of τ_{LR} that is necessary for the fine-resolution execution of a^H—we call this operation *zooming*. The following definitions are used to identify refinements of σ_1 and σ_2, the states of τ_{LR} relevant to T, and remove irrelevant fluents and actions.

Definition 3. *[Direct relevance]*
An element y of a basic sort st_H of $\mathscr{D}_H$ is *directly relevant* to transition T of τ_H if:

- Element y occurs in a^H; or
- For some f, $f(\bar{x}) = y$ belongs to σ_1 or σ_2 but not both.

Consider the transition corresponding to robot rob_1 moving from the $kitchen$ to the $office$, i.e., $a^H = move(rob_1, office)$. For this transition, element rob_1 of sort *robot*, and elements $office$ and $kitchen$ of sort *place*, are relevant.

Definition 4. *[Zoom]*
To construct $\mathscr{D}_{LR}(T)$, we need to determine the signature and the axioms describing the transition diagram $\tau_{LR}(T)$. The signature of $\mathscr{D}_{LR}(T)$ is constructed as follows:

1. If st_H is a sort of $\mathscr{D}_H$ with at least one element directly relevant to T:
   - If sort st_H is not magnified, it is its own *zoomed counterpart* st_L^z.
   - If sort st_H is magnified, st_L^z is the set of components of elements of st_H directly relevant to T.

   The zoomed counterparts form a hierarchy of basic sorts of $\mathscr{D}_{LR}(T)$ (with subclass relation inherited from $\mathscr{D}_{LR}$). In our example, the sorts of $\mathscr{D}_{LR}(T)$ are $robot_L^z = \{rob_1\}$ and $place_L^z = \{c_i : c_i \in kitchen \cup office\}$.

2. Functions of $\mathscr{D}_{LR}(T)$ are those of $\mathscr{D}_{LR}$ restricted to the identified sorts. Functions in our example include $loc(rob_1)$ taking values from $place_L^z$, $next\_to(place_L^z, place_L^z)$, and restricted functions related to testing these functions' values.

3. Actions of $\mathscr{D}_{LR}(T)$ are restricted to the identified sorts. In our example, relevant actions are of the form $move(rob_1, c_i)$ and $test(rob_1, loc(rob_1), c_i)$, where $c_i \in place_L^z$.

Axioms of $\mathscr{D}_{LR}(T)$ are those of $\mathscr{D}_{LR}$ restricted to the signature of $\mathscr{D}_{LR}(T)$. In our example, this interpretation removes the causal laws for *grasp* and *put_down*, and removes the state constraint related to fluent *in_hand*. Also, in the axioms corresponding to action *move*, C can only take values from any cell in the $kitchen$ or the $office$.

Example 3. *[Example of zoom]*
If rob_1 has to execute $a^H = grasp(rob_1, tb_1)$ to pickup textbook tb_1 in the $office$, zooming constructs the signature:

- Relevant sorts of $\mathscr{D}_H$ are *robot*, *object*, and *place*, and $st_L^z = \{robot_L^z, object_L^z, place_L^z\}$ are the zoomed counterparts in $\mathscr{D}_{LR}(T)$, with $robot_L^z = \{rob_1\}$, $object_L^z = \{tb_1\}$ and $place_L^z = \{c_i : c_i \in office\}$.
- Functions of $\mathscr{D}_{LR}(T)$ include (a) $loc(robot_L^z)$ and $loc(object_L^z)$, basic fluents that takes values from $place_L^z$; (b) static $next\_to(place_L^z, place_L^z)$; (c) defined fluent $range(loc(robot_L^z), place_L^z)$; and (d) knowledge fluent dir_obs restricted to st_L^z, e.g., $dir\_obs(robot_L^z, loc(object_L^z), place_L^z)$.
- Actions of $\mathscr{D}_{LR}(T)$ include (a) $move(robot_L^z, place_L^z)$; (b) $grasp(robot_L^z, object_L^z)$; (c) $putdown(robot_L^z, object_L^z)$; and (d) knowledge-producing actions to test the location of rob_1 and tb_1.

Axioms of $\mathscr{D}_{LR}(T)$ include:

$move(rob_1, c_j)$ **causes** $loc(rob_1) = \{C : range(loc(rob_1), C)\}$
$grasp(rob_1, tb_1)$ **causes** $in\_hand(rob_1, tb_1) = \{true, false\}$
$test(rob_1, loc(tb_1), c_j)$ **causes** $dir\_obs(rob_1, loc(tb_1), c_j) = \{true, false\}$ **if** $loc(tb_1) = c_j$
impossible $move(rob_1, c_j)$ **if** $loc(rob_1) = c_i$, $\neg next\_to(c_j, c_i)$

where $range(loc(rob_1), C)$ may hold for values that are elements of $place_L^z$ and within the range of the robot's current location. The states of $\tau_{LR}(T)$ thus include atoms of the form $loc(rob_1) = c_i$ and $loc(tb_1) = c_j$, where $c_i, c_j \in place_L^z$, $in\_hand(rob_1, tb_1)$, direct observations of these atoms, and statics. Actions include $move(rob_1, c_i)$, $grasp(rob_1, tb_1)$, $putdown(rob_1, tb_1)$ and *test* actions.

POMP construction: The statistician uses the system description $\mathscr{D}_{LR}(T)$, and the learned probabilities, to construct a partially observable Markov decision process (POMDP) for the probabilistic implementation of a^H. It may be possible to use other computationally efficient probabilistic models for specific actions. However, POMDPs provide (a) quantifiable trade-off between accuracy and efficiency in the presence of uncertainty; and (b) near-optimal solution if the POMDP is modeled correctly. Furthermore, our architecture only constructs a POMDP for the relevant part of the domain, and many of the POMDPs may be precomputed.

A POMDP is described by a tuple $\langle S^L, A^L, Z^L, T^L, O^L, R^L \rangle$ for a goal state [Littman, 1996]. Elements of this tuple are the set of states, set of actions, set of values of observable fluents, the state transition function, the observation function, and the reward specification. The last three elements are based on the statistics acquired during randomization—please see [Zhang *et al.*, 2015]. The POMDP formulation considers states to be *partially observable*, and reasons with probability distributions over the states, called *belief states*. Functions T^L and O^L describe a probabilistic transition diagram over belief states. The POMDP formulation also implicitly includes a history of observations and actions—the current belief state is assumed to be the result of all information obtained so far. The POMDP tuple is used to compute a *policy* that maps belief states to actions, using an algorithm that maximizes the reward over a planning horizon. The policy is used to choose an action in the current belief state, revising the belief state through Bayesian updates after executing the action and receiving an observation. The belief update continues until a terminal action is executed because it has a higher (expected) utility than continuing to execute non-terminal actions. The corresponding observations and action outcomes revise $\mathscr{H}$,

to be used for subsequent reasoning by the logician. If τ_H is constructed correctly, and the POMDP correctly captures the domain dynamics, following an optimal policy produced by an exact POMDP solver is most likely (among all possible policies) to achieve the goal state [Littman, 1996]. Since we use an approximate solver for computational efficiency, we obtain a bound on the regret (i.e., loss in value) due to the computed policy [Ong *et al.*, 2010].

Example 4. *[POMDP construction]*
Consider the implementation of $a^H = move(rob_1, kitchen)$, with rob_1 in the $office$ and one cell in each room. Assume that (a) move from a cell to a neighboring cell succeeds with probability 0.85—otherwise, the robot remains where it is; (b) non-terminal actions have unit cost; and (c) terminating the POMDP policy after reaching cell 1 (*kitchen*) receives a large positive reward (100), whereas termination in cell 0 ($office$) receives -100. In the description below, states are possible robot locations, and *absb* is a terminal state. Actions correspond to the robot moving to specific cells, testing its cell location, or terminating the policy. The robot observes its cell location or receives no observation. Knowledge-producing actions do not cause a state transition, and actions that change the state do not provide observations. For each state s_i and action a_j, the corresponding ASP program $\Pi(\mathscr{D}_{LR}(T), s_i, a_j)$ is solved to (a) identify inconsistencies and eliminate impossible states, e.g., the robot and an object cannot be in different locations when the robot is holding the object; and (b) identify and eliminate impossible transitions in the construction of T^L.

```
discount: 0.99
states: robot-0 robot-1 absb
actions: move-0 move-1 test-0 test--1 done
observations: rob-found rob-not-found none

% Transition function format
% T : action : S x S' -> [0, 1]
T: move-0
1       0       0
0.85    0.15    0
0       0       1

T: move-1
0.15    0.85    0
0       1       0
0       0       1

T: test-robot-0
identity

T: test-robot-1
identity

T: done
uniform

% Observation function format
% O : action : s_i : z_i -> [0, 1] (or)
%             : S x Z -> [0, 1]
O: move-0 : * : none 1
O: move-1 : * : none 1

O: test-robot-0
0.95    0.05    0
0.05    0.95    0
0       0       1

O: test-robot-1
0.05    0.95    0
0.95    0.05    0
0       0       1

O: done : * : none 1

% Reward function format
% R : action : s_i : s_i' : real value
R: * : * : * : -1
R: done   : robot-0 : * : -100
R: done   : robot-1 : * : 100
```

5 Reasoning System

For any given goal, the reasoning system first has the logician use $\mathscr{D}_H$ and $\mathscr{H}$ (with initial state defaults) to explain inconsistencies and compute a plan of abstract actions. For an abstract action, the statistician zooms to $\mathscr{D}_{LR}(T)$, the relevant part of the randomized fine-resolution refinement of $\mathscr{D}_H$, using $\mathscr{D}_{LR}(T)$ and probabilities (of action outcomes) to construct a POMDP. The policy obtained by solving the POMDP is invoked to execute a sequence of concrete actions, with the corresponding outcomes being added to $\mathscr{H}$ and used by the logician for subsequent reasoning. Similar to prior work [Zhang *et al.*, 2014], experimental results in simulation and on a physical robot finding and moving objects between rooms—not reported here due to space limitations, see [Sridharan *et al.*, 2015]—indicate that the architecture scales well, is more accurate and computationally efficient than a probabilistic approach, and supports reasoning with violation of defaults, and with unreliable observations and actions.

6 Conclusions

This paper described the design of robots that can represent and reason with logic-based and probabilistic descriptions of domain knowledge and uncertainty. Our architecture uses tightly-coupled transition diagrams of the domain at two levels of granularity. For any given goal, non-monotonic logical reasoning at the coarse-resolution plans a sequence of abstract actions. Each abstract action is implemented probabilistically as a sequence of concrete actions by zooming to the relevant part of the fine-resolution description. The individual design steps are illustrated using examples.

Acknowledgements

The authors thank Jeremy Wyatt and Shiqi Zhang for discussions related to the architecture described in this paper. The first author was supported in part by the US Office of Naval Research Science of Autonomy award N00014-13-1-0766.

References

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

[Balai *et al.*, 2013] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. Towards Answer Set Programming with Sorts. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, Corunna, Spain, 2013.

[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.

[Balduccini *et al.*, 2014] Marcello Balduccini, William C. Regli, and Duc N. Nguyen. An ASP-Based Architecture for Autonomous UAVs in Dynamic Environments: Progress Report. In *International Workshop on Non-Monotonic Reasoning (NMR)*, Vienna, Austria, July 17-19, 2014.

[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.

[Chen *et al.*, 2012] Xiaoping Chen, Jiongkun Xie, Jianmin Ji, and Zhiqiang Sui. Toward Open Knowledge Enabling for Human-Robot Interaction. *Human-Robot Interaction*, 1(2):100–117, 2012.

[Gelfond and Inclezan, 2013] Michael Gelfond and Daniela Inclezan. Some Properties of System Descriptions of AL_d. *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming*, 23(1-2):105–120, 2013.

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

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

[Hawes *et al.*, 2010] Nick Hawes, Jeremy Wyatt, Mohan Sridharan, Henrik Jacobsson, Richard Dearden, Aaron Sloman, and Geert-Jan Kruijff. Architecture and Representations. In *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs*, pages 51–93. Springer Berlin Heidelberg, April 2010.

[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), 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*, Stanford, USA, March 2015.

[Lee *et al.*, 2013] Joohyun Lee, Vladimir Lifschitz, and Fangkai Yang. Action Language BC: Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Beijing, China, August 3-9, 2013.

[Leone *et al.*, 2006] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[Littman, 1996] Michael Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, Department of Computer Science, Providence, USA, March 1996.

[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. *IJRR*, 29(8):1053–1068, July 2010.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine learning*, 62(1), 2006.

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

[Sridharan and Gelfond, 2016] Mohan Sridharan and Michael Gelfond. Using Knowledge Representation and Reasoning Tools in the Design of Robots. In *IJCAI Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS)*, New York, USA, July 10, 2016.

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

[Sridharan *et al.*, 2016] Mohan Sridharan, Prashanth Devarakonda, and Rashmica Gupta. Discovering Domain Axioms Using Relational Reinforcement Learning and Declarative Programming. In *ICAPS Workshop on Planning and Robotics (PlanRob)*, London, UK, June 13-14, 2016.

[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 (AAAI)*, pages 1394–1400, Austin, USA, 2015.

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

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