Using Knowledge Representation and Reasoning Tools in the Design of 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

The paper describes the authors' experience in using knowledge representation and reasoning tools in the design of robots. The focus is on the systematic construction of models of the robot's capabilities and its domain at different resolutions, and on establishing a clear relationship between the models at the different resolutions.

1 Introduction

Our prior paper described an architecture for robots [Zhang *et al.*, 2014], whose reasoning system we view (in this paper) as an interplay between a logician and a statistician. The former has an abstract, coarse-resolution view of the world. The logician perceives an office domain, for instance, as a collection of connected rooms containing various types of objects, and assumes that the robot is capable of successfully moving between two adjacent rooms, and of finding an object located in a room. If the robot's goal is to find a specific book, the logician can design a plan of abstract actions directing the robot to go to the library, where books are normally stored. The first action of this plan, e.g., *move to the lab*, which happens to be the first room on the way to the library, will be passed to a statistician, who has a rather different, fine-resolution view of the world. According to the statistician, the same office domain consists of small grid cells belonging to the different rooms. The statistician believes that the robot is usually capable of moving to a neighboring cell, and of checking a cell for a target object, but these actions are non-deterministic—they only succeed with some probability that the statistician knows a priori or can learn by running some experiments. For the coarse-resolution action of moving to the lab, the statistician thus 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 abstract action passed by the logician has been executed, i.e., that the robot has moved to the lab, this information is reported back to the logician. Otherwise, statistician reports failure after some time, and the logician has to diagnose and replan. Even this simple scenario shows that our robot's architecture should be capable of representing and manipulating both logical and probabilistic knowledge. *In this paper we report our experience in the systematic 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 mathematical models of the domain.

We started with describing the state transition diagram modeling possible trajectories of the robot's domain, in action description language AL_d [Gelfond and Inclezan, 2013]. Using AL_d allowed us to concisely represent the diagram even for domains with complex relationships between fluents, and represent recorded histories of the robot's actions and observations. We *expanded the standard notion of recorded history to include the robot's default knowledge about the initial situation*, which simplifies the diagnostics task. Although there exist action languages that allow causal laws specifying default values of fluents at arbitrary time steps [Lee *et al.*, 2013], such languages are too powerful for our purposes, and occasionally pose difficulties with representing all exceptions to such defaults when the domain is expanded. Reasoning with theories in AL_d is performed by reducing planning, diagnostics, and other robotic tasks to computing answer sets of a program in CR-Prolog, a variant of Answer Set Prolog (ASP) that supports representation and reasoning with defaults and their direct and indirect exceptions [Balduccini and Gelfond, 2003]. This reduction, which considers histories with initial state defaults, is the first novel contribution, but it expands previous work that established close relationship between action languages and logic programming with answer set semantics. Existing efficient solvers help automate the necessary reasoning [Leone *et al.*, 2006].

The second novel contribution is the *precise definition of the world view of the statistician as a refinement of the transition diagram of the logician*. The new diagram can be viewed as the result of increasing the resolution of the robot's ability to see the world. In our example, the new world view includes cells in rooms, which were so far invisible, and relations and actions involving these cells, e.g. location of an object in a cell, and the action of moving from a cell to its neighbor. Construction of such a refinement may require some amount of domain knowledge, but the statistician's diagram is related to the logician's diagram in a precise way that is captured by our novel mathematical definition of refinement. We provide detailed guidelines for the construction of the refinement's action theory, which includes axioms establishing the relationship between fluents of the coarse-resolution diagram and their fine-resolution counterparts, as well as the axioms describing the effects of observations.

After the refinement of the domain is constructed, 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 the action language AL_d by a construct that supports the natural representation of such effects*. Probabilities computed experimentally by the statistician 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" possible action, make observations, and update the belief state until the abstract action provided by the logician is completed with high probability, or all hope of doing so is lost. However, POMDP algorithms need a representation of the diagram that lists all possible combinations of physical states and actions, which can become intractable even for a comparatively small collection of fluents. 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 execute the action "move from room R_1 to room R_2", with the two rooms being next to each other, the fluents and actions related to other rooms are eliminated, dramatically reducing the size of the corresponding diagram. The fourth new contribution is a *precise definition of zooming, which helps automate this process*. Finally, the zoomed part of the randomized fine-resolution diagram 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 action outcomes being added to the coarse-resolution history.

2 Related Work and Example Domain

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 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 explore the systematic construction of robots with the desired knowledge representation and reasoning capabilities. We illustrate these design steps and our contributions using the following example.

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 $object$.
- Four places: $office$, $main\_library$, $aux\_library$, and $kitchen$ of which some of them 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 $object$.

Although this domain may appear simplistic, it illustrates many of the representation, reasoning, perception, and actuation challenges that exist in more complex robotics domains.

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:

$$\begin{aligned}&a \textbf{ causes } f(\bar{x}) = y \textbf{ if } \ body\\&f(\bar{x}) = y \ \textbf{ if } \ body\\&impossible\ a_0, \dots, a_n \ \textbf{ if } \ body\end{aligned}$$

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

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

says that a robot R moving to place Pl will end up in Pl.

The second statement is a state constraint, which says that $f(\bar{x}) = y$ in any state that satisfies *body*. For instance:

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

guarantees that the object grasped by a robot shares the robot's location. The third statement prohibits simultaneous execution of actions $a_0,\dots,a_n$ 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*). Fluent 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 the original AL_d is discussed in [Gelfond and Kahl, 2014]. Unfortunately, it only allows boolean fluents, a restriction that is removed in our variant of the language.

Histories with defaults: In addition to action theory of AL_d, the logician's knowledge contains a *recorded history*, a collection of the robot's observations and actions. In action languages, such a 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. This syntax of histories was rather limited for our purposes, and the robot can benefit substantially from some forms of default knowledge. In Example 1, the robot can, for instance, be told that books are normally kept in the library. To address this limitation, we introduced an additional type of historical record:

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

to assume that if the initial state satisfies *body*, the value of $f(\bar{x})$ in this state is y. For instance, the default statement:

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

says that textbooks are normally kept in the library, whereas the default statement:

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

gives the second most likely location for a textbook. Such defaults may substantially simplify the planning of the logician. For instance, the plan for finding a textbook tb will consist of going directly to the library and looking for the book there. However, defaults are also useful for diagnostics. For instance, if tb is not found in the library, i.e., history includes $obs(loc(tb),library,false,i)$, the robot would realize that this observation defeats the first default, try the second default, and look for the book in the office. To ensure such behavior, we had to redefine the notion of a model of a history, and design a new algorithm for computing such models. To this end, given an AL_d theory $\mathscr{D}_H$ and recorded history $\mathscr{H}$, we construct the program $\Pi(\mathscr{D}_H,\mathscr{H})$ consisting of the standard encoding of $\mathscr{D}_H$ into ASP, and the collection of atoms representing observations and actions of $\mathscr{H}$ together with the initial state defaults. This encoding allows indirect exceptions to defaults (like an 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—these tasks are reduced to computing answer sets of program $\Pi(\mathscr{D}_H,\mathscr{H})$ combined with standard encoding of the robot's goal. This program is passed to an efficient ASP solver—we use SPARC, which expands CR-Prolog and provides explicit constructs to specify objects, relations, and their sorts [Balai *et al.*, 2013]. Atoms of the form $occurs(action, step)$ belonging to the answer set obtained by solving this program, e.g., $occurs(a_1,1),\dots,occurs(a_n,n)$, 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. In a similar manner, suitable atoms in the answer set can be used for diagnostics, e.g., to explain unexpected observations.

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 the transition diagram τ_H. Σ_H defines the names of objects and functions available for use, e.g., the sorts are $place$, $thing$, $robot$, and $object$, with $object$ and $robot$ being subsorts of $thing$, and $textbook$, $printer$ and $kitchenware$ being subsorts of $object$. The statics include a relation $next\_to(place, place)$, which describes if two places are next to each other. The domain's fluents are: $loc : thing \to place$ and $in\_hand : robot \times object \to boolean$. These are basic fluents subject to the laws of inertia. The domain's actions are $move(robot, place)$, $grasp(robot, object)$, and $putdown(robot, object)$. The domain dynamics are defined using axioms that consist of causal laws:

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

state constraints:

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

and executability conditions such as:

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

For any given domain, the part of Σ_H described so far is unlikely to change substantially. However, the last step in the construction of Σ_H, which populates the basic sorts with specific objects, e.g $robot = \{rob_1\}$, $place = \{r_1,\dots,r_n\}$, and $textbook = \{tb_1,\dots tb_n\}$, is likely to undergo frequent revisions. Ground instances of axioms are obtained by replacing variables by ground terms from the corresponding sorts.

In the transition diagram τ_H described by $\mathscr{D}_H$, actions are assumed to be deterministic, and values of fluents are assumed to be observable, which aid in fast, tentative planning and diagnostics for achieving the goals. This domain representation should ideally be tested extensively by including various recorded histories of the domain, which may include histories with prioritized defaults, and using the resulting programs to solve various reasoning tasks.

4 Statistician's Description

In this section we describe our design of the statistician.

Refinement: First we specify the *deterministic* version of the world of the statistician. It would be given by a system description $\mathscr{D}_L$ defining a transition diagram τ_L which serves as a (deterministic) refinement of the logician's diagram τ_H. Recall that τ_L is the result of increased resolution which magnifies some objects in the signature Σ_H of system description $\mathscr{D}_H$ of τ_H. Newly discovered parts of the magnified objects are referred to as its refined *components*. In our example, for instance, every room is magnified and is viewed as a collection of its component cells.

The refinement will be represented by a system description $\mathscr{D}_L$ with signature Σ_L. 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* $st_L = \{o_1, \dots, o_m\}$ consisting of the components of magnified elements of st_H, e.g., Σ_L replaces sort $place = \{r_1, \dots, r_n\}$ of rooms in Σ_H by:
$$place^* = \{r_1, \dots, r_n\}$$
$$place = \{c_1, \dots, c_m\}$$
where rooms are magnified and viewed as a collection of newly discovered objects, their component cells $c_1, \dots, c_m$.
- Introducing static relation *component* between magnified objects from st_L^* and the corresponding newly-discovered objects from st_L, e.g., relation $component(c,r)$ is defined to be 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). In our example, the new fluents obtained by refinement are:
$$loc^* : thing \to place^*$$
$$loc : thing \to place$$
Other fluents (and their signatures) are unchanged.
- Obtaining actions of Σ_L by replacing the magnified parameters of the original actions from Σ_H by their fine-resolution counterparts. In our example, Σ_L will contain original actions *grasp* and *putdown*, and new action $move(robot, cell)$ of a robot moving to an (adjacent) cell. Σ_L will also include knowledge-producing action $test(robot, fluent, value)$ that activates algorithms on the robot to check the value of an observable fluent of $\mathscr{D}_L$ in a given state, e.g., $test(R, loc(Th), Cell)$. Note that *test* is a general action belonging to every refinement of Σ_H. We also add fluents to describe the result of testing, e.g., $observed(robot, fluent, value)$ is true if the most recent (direct/indirect) observation of $fluent$ returned $value$.

Axioms of the refined system description $\mathscr{D}_L$ include axioms of $\mathscr{D}_H$, and domain-dependent axioms relating coarse-resolution fluents and their fine-resolution counterparts, e.g., in our illustrative domain we have:

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

which includes the new static $component(c,r)$ for every cell c within every room r—$next\_to(c_1, c_2)$ is defined in a similar manner for every pair of adjacent cells accessible from each other. More importantly, we add to $\mathscr{D}_L$, 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:

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

In our example, robot rob_1 testing if location of object o is cell c will make basic knowledge fluent $dir\_obs(rob_1, loc(o), c)$ true if the object is indeed there. Note, that *dir_obs* may have three possible values, *true*, *false* and *undef*—it 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. The designers of the statistician should make sure that τ_L specified by our system description $\mathscr{D}_L$ is indeed a refinement, i.e., it matches the following formal definitions of refinement of a state and a system description.

Definition 1. *[Refinement of a state]*
A state δ of τ_L is said to be a *refinement* of a 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: Our next step is to expand $\mathscr{D}_L$ to capture the non-determinism in action execution and observations on the robot, which is essential for the statistician. To do so, we *extended AL_d by non-deterministic causal laws*:

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

where the first statement says that if 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—second statement says that f may take any value from $\{sort\_name \cap range(f)\}$. Randomized fine-resolution system description $\mathscr{D}_{LR}$ is then 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, in our example, the non-deterministic 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 other cells that are within range—the robot's current cell and neighboring cells are all within range.

To complete our model of the statistician with probabilistic information, we run experiments that sample specific instances of each ground non-deterministic causal laws 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 collected in an initial training phase* are 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, given a specific robot. 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, the robot may compute that 0.85 is the probability with which it can recognize a specific object in a specific cell. Any prior beliefs about these probabilities (e.g., from a human) can be used as the initial belief that is revised by the experimental trials.

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 can be computationally intractable, the statistician first identifies the part $\tau_{LR}(T)$ of diagram τ_{LR} that is necessary for the fine-resolution execution of a^H—we call this operation *zooming*. The size of $\tau_{LR}(T)$ is decreased with respect to τ_{LR} by only considering refinements of σ_1 and σ_2, the states of τ_{LR} relevant to T. Next, fluents and actions not relevant to the execution of a^H are removed based on the following definitions.

Definition 3. *[Direct relevance]*
An element y of a basic sort st_H of $\mathscr{D}_H$ is *directly relevant* to a 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 are $loc(rob_1)$ taking values from $place_L^z$, $next\_to(place_L^z, place_L^z)$, $range(loc(rob_1), place_L^z)$, and properly restricted functions related to testing these functions' values.
3. Actions of $\mathscr{D}_{LR}(T)$ are restricted to the sorts identified above. In our example, the relevant actions are of the form $move(rob_1, c_i)$ and $test(rob_1, loc(rob_1), c_i)$, where c_i are individual elements of $place_L^z$.

The axioms of $\mathscr{D}_{LR}(T)$ are those of $\mathscr{D}_{LR}$ that are 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* in $\mathscr{D}_{LR}$. Furthermore, in the causal law and executability condition corresponding to the action *move*, C can only take values from $place_L^z$, i.e., any cell in the *kitchen* or the *office*.

Example 3. *[Example of zoom]*
Assume that robot rob_1 has to execute $a^H = grasp(rob_1, tb_1)$ to pickup textbook tb_1—rob_1 is, and tb_1 is assumed to be, in the *office*. Zooming constructs the following 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) the knowledge fluent *dir_obs* restricted to the zoomed sorts, e.g., $dir\_obs(robot_L^z, loc(robot_L^z), place_L^z)$ and $dir\_obs(robot_L^z, loc(object_L^z), place_L^z)$.
- 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, e.g., $test(robot_L^z, loc(object_L^z), place_L^z)$.

The axioms of $\mathscr{D}_{LR}(T)$ are those of $\mathscr{D}_{LR}$ restricted to the signature of $\mathscr{D}_{LR}(T)$. For instance, these axioms include:

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

where $range(loc(rob_1), C)$ may hold for values in $\{c_i, c_j, c_k\}$ within the range of the robot's current location (c_i), and are elements of $place^z_L$. 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 and c_j are values in $place^z_L$, $in\_hand(rob_1, tb_1)$, direct observations of these atoms, and statics such as $next\_to(c_i, c_j)$. Specific actions include $move(rob_1, c_i)$, $grasp(rob_1, tb_1)$, $putdown(rob_1, tb_1)$ and *test* actions.

POMDP construction: The statistician uses system description $\mathscr{D}_{LR}(T)$, and the learned probabilities, to construct a POMDP for the probabilistic implementation of a^H in state σ_1 of τ_H. It may be possible to use other (more computationally efficient) probabilistic models for implementing specific actions. Also, it may be possible to use specific (computationally efficient) heuristic or probabilistic algorithms for specific tasks such as path planning. However, POMDPs provide (a) principled and quantifiable trade-off between accuracy and computational efficiency in the presence of uncertainty; and (b) near-optimal solution if the POMDP is modeled accurately. Also, our architecture only constructs a POMDP for a small (relevant) part of the domain, significantly reducing the computational complexity of solving the POMDP. Furthermore, many POMDPs for any given domain can be precomputed, solved and used as needed.

A POMDP is described by a tuple $\langle S^L, A^L, Z^L, T^L, O^L, R^L \rangle$ for a specific goal state. Elements of this tuple correspond to the set of states, set of actions, set of values of observable fluents, the state transition function, the observation function, and the reward specification, with the last three elements being based on the statistics acquired during randomization—for details about POMDPs and their use in AI and robotics, see [Littman, 1996; 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 in so far. The POMDP tuple is used to compute a *policy* $\pi : b_t \mapsto a^L_{t+1}$, which maps belief states to actions, using an algorithm that maximizes the reward over a planning horizon. The policy is then used to choose an action in the current belief state, revising the belief state through Bayesian updates after executing the action a_{t+1} and receiving an observation o_{t+1}. The belief update continues until policy execution is terminated. In our case, a terminal action is executed when it has a higher (expected cumulative) utility than continuing to execute non-terminal actions. This action choice happens when the belief in a specific state is high (e.g., ≥ 0.8), or none of the states have a high probability associated with them after invoking the policy several times. The latter case is interpreted as the failure to execute the coarse-resolution action.

The observations and action outcomes obtained by executing the POMDP policy correspond to observations and fluents in $\mathscr{D}_{LR}(T)$. This information, in turn, revises $\mathscr{H}$, to be used for subsequent reasoning by the logician. If τ_H is constructed correctly, and the statistics collected during the initial training phase correctly model the domain dynamics, following an optimal policy produced by an exact POMDP solver is most likely (among all possible policies) to take the robot to the desired goal state [Littman, 1996; Sondik, 1971]. 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]. Due to space constraints, we provide (below) a simplistic example of constructing a POMDP.

Example 4. *[POMDP construction]*
Consider rob_1 in the $office$ and the implementation of $a^H = move(rob_1, kitchen)$, with one cell in each room. Due to zooming, the robot only reasons about its location. Assume that a move from a cell to a neighboring cell succeeds with probability 0.85—otherwise, the robot remains where it is. Assume that all non-terminal actions have unit cost. Terminating the POMDP policy after reaching cell 1 (in *kitchen*) receives a large positive reward (100), whereas termination while in cell 0 (in $office$) receives a large negative reward (-100). Elements of the POMDP are described (below) in the format used by the POMDP solver [Ong *et al.*, 2010].

The states correspond to possible robot locations, and *absb* is a terminal state. The actions correspond to the robot moving to specific cells, testing its cell location, or terminating policy execution (transitioning to *absb*). 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. Our architecture constructs such data structures for complex domains by zooming to the relevant part of the system description (as described earlier). Also, 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 possible state transitions, eliminating impossible transitions in the construction of the transition function.

```
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

Algorithm 1 describes the reasoning loop. For any given goal, the logician reasons with the system description $\mathscr{D}_H$ and a recorded history $\mathscr{H}$ with initial state defaults, accounting for any discrepancies between observations and predictions to compute a plan of abstract actions. For an abstract action a^H, the statistician zooms to the relevant part of the randomized refinement of $\mathscr{D}_H$. The corresponding system description $\mathscr{D}_{LR}(T)$ and probabilities are used to construct and solve a POMDP. The POMDP policy is invoked repeatedly (until termination) to execute a sequence of concrete actions. The corresponding outcomes are reported to the logician to be used for subsequent reasoning. Correctness of this control loop is ensured by (1) applying the algorithm that reduces coarse-resolution planning and diagnostics to computing answer sets of the corresponding program; (2) using the refinement and zoom operations as described above; and (3) using POMDPs to probabilistically execute an action sequence for each abstract action in the logician's plan.

Experimental trials, both in simulation and on physical robots assisting humans in finding and moving objects an office domain, indicate that our architecture provides significantly higher efficiency and accuracy in comparison with using just POMDPs with hierarchical decompositions, similar to our prior work [Sridharan *et al.*, 2015]. These results are not described here because the focus is on describing the steps in the design process, and due to space limitations.

Algorithm 1: Control loop

```
Input: coarse-resolution system description $\mathscr{D}_H$ and
       history $\mathscr{H}$; randomized fine-resolution system
       description $\mathscr{D}_{LR}$; coarse-resolution goal.
Output: robot is in a state satisfying the goal; reports
        failure if this is impossible.
1  while goal is not achieved do
2      Logician uses $\tau_H$ and $\mathscr{H}$ to find a plan of abstract
       actions, $a_1^H, \dots, a_n^H$ to achieve the goal.
3      if no plan exists then
4          return failure
5      end
6      i := 1, continue1 := true
7      while continue1 do
8          Check pre-requisites of $a_i^H$.
9          if pre-requisites not satisfied then
10             continue1 := false
11         else
12             Statistician zooms to the relevant part of $\tau_{LR}$
               for executing $a_i^H$ and constructs a POMDP.
13             Statistician solves POMDP to compute an
               action policy to implement $a_i^H$.
14             continue2 := true
15             while continue2 do
16                 Statistician invokes POMDP policy to
                   select and execute an action, obtain
                   observation, and update belief state.
17                 if terminal action invoked then
18                     Statistician communicates success or
                       failure of $a_i^H$ to the logician, to be
                       recorded in $\mathscr{H}$.
19                     continue2 = false
20                     i := i+1
21                     continue1 := $(i < n+1)$
22                 end
23             end
24         end
25     end
26 end
```

6 Conclusions

This paper described the systematic design of robots capable of representing and reasoning 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, with a fine-resolution diagram being a refinement of a coarse-resolution diagram. 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, constructing and solving a a POMDP, and invoking a policy until termination. The corresponding outcomes revise the coarse-resolution history for subsequent reasoning. The design steps are illustrated using examples, which indicate that the architecture supports reasoning at the sensorimotor level and the cognitive level with violation of defaults, and unreliable observations and actions.

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.

[Sondik, 1971] Edward J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

[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.