In the Workshop on *Planning and Robotics* at ICAPS, Delft, The Netherlands, June 26, 2018.

Representing and Reasoning with Intentional Actions on a Robot

Rocio Gomez, Mohan Sridharan and Heather Riley
Department of Electrical and Computer Engineering
The University of Auckland, New Zealand
m.gomez@auckland.ac.nz, m.sridharan@auckland.ac.nz,hril230@aucklanduni.ac.nz

Abstract

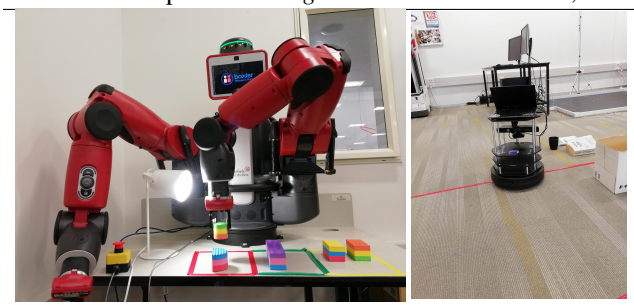
This paper describes a general architecture for robots to represent and reason with intentional actions. The architecture reasons with tightly-coupled transition diagrams of the domain at two different resolutions. Non-monotonic logical reasoning with a coarse-resolution transition diagram is used to compute a plan comprising intentional abstract actions for any given goal. Each such abstract action is implemented as a sequence of concrete actions by reasoning over the relevant part of the fine-resolution transition diagram, with the outcomes of probabilistic execution of the concrete actions being added to the coarse-resolution history. The capabilities of this architecture are illustrated in the context of a simulated robot assisting humans in an office domain, on a physical robot (Baxter) manipulating tabletop objects, and on a wheeled robot (Turtlebot) moving objects to particular places or people in an office. We show that this architecture improves reliability and efficiency in comparison with a planning architecture that does not include intentional actions.

1 Introduction

Consider robots assisting humans in dynamic domains, e.g., a robot helping a human arrange objects in different configurations on a tabletop in Figure 1a, or a robot delivering objects to particular places or people in Figure 1b. These robots often have to reason with different descriptions of uncertainty and incomplete domain knowledge. This information about the domain often includes commonsense knowledge, especially default knowledge that holds in all but a few exceptional circumstances (e.g., "books are usually in the library but cookbooks may be in the kitchen"), and probabilistic quantification of the uncertainty in sensing and actuation (e.g., "I am 90% certain the robotics book is on the table"). The robot also receives a lot more raw sensor data than it can process, and may be equipped with many algorithms to process the data. Furthermore, while it is difficult to provide robots comprehensive domain knowledge or elaborate supervision, reasoning with incomplete or incorrect information can provide incorrect or suboptimal outcomes. This loss in performance is more pronounced in scenarios corresponding to unexpected success or failure, which are common in dynamic domains. For instance, consider a robot trying to move two books from an office to a library. After moving the first book to the library, if the robot observes the second book in the library, or if it observes the second book in the kitchen on the way back to the office, it should stop executing its plan, reason about what may have happened, and compute a new plan if necessary. One way to achieve this behavior is to augment a traditional planning approach with the ability to reason about observations of all domain objects and events during plan execution, but this approach will become computationally intractable in complex domains. Instead, the architecture described in this paper seeks to enable a robot pursuing a particular goal to automatically reason about the underlying *intention* and related observations of its domain during planning and execution. It does so by building on an architecture that uses declarative programming to reason about intended actions to achieve a given goal (Blount, Gelfond, and Balduccini 2015), and on an architecture that reasons with tightly-coupled transition diagrams at different levels of abstraction (Sridharan et al. 2017). We describe the following characteristics of the architecture:

- An action language is used to describe the tightly-coupled transition diagrams of the domain at two different resolutions. At the coarse resolution, non-monotonic logical reasoning with commonsense knowledge, including default knowledge, produces a sequence of intentional abstract actions for any given goal.
- Each intended abstract action is implemented as a sequence of concrete actions by automatically zooming to the relevant part of the fine-resolution system description defined as a refinement of the coarse-resolution system description. The outcomes of executing the concrete actions using probabilistic models or uncertainty are added to the coarse-resolution history.

In this paper, the coarse-resolution and fine-resolution action language descriptions are translated to programs in CR-Prolog, an extension of Answer Set Prolog (ASP) (Gelfond and Kahl 2014), for commonsense reasoning. The execution of each concrete action using probabilistic models of uncertainty in sensing and actuation is achieved using existing algorithms. The architecture thus reasons about intentions and beliefs at different levels of resolution. We demonstrate the general applicability of our architecture in the context of a (i) simulated robot assisting humans in an office domain; (ii) physical robot (Baxter) manipulating objects on a tabletop; and (iii) wheeled robot (Turtlebot) moving objects to desired



(a) Baxter robot. (b) Turtlebot.

Figure 1: (a) Baxter robot manipulating objects on a tabletop; and (b) Turtlebot moving objects to particular locations in a lab.

locations in an office domain. We show that the proposed architecture improves reliability and computational efficiency of planning and execution in dynamic domains in comparison with a planning architecture that does not support reasoning about intentional actions.

2 Related Work

There is much work in the modeling and recognition of intentions. Belief-desire-intention (BDI) architectures for reasoning agents model intention and guide reasoning by eliminating choices inconsistent with current intentions (Bratman 1987; Rao and Georgeff 1995). However, such architectures do not learn from past behavior, adapt to new situations, or include an explicit representation of (or reasoning about) goals. Other work has reasoned with domain knowledge or used models learned from training samples to recognize intentions (Kelley et al. 2008).

An architecture formalizing intentions based on declarative programming was presented in (Baral and Gelfond 2005). It introduced an action language that can represent intentions based on two principles: (i) *non-procrastination*, i.e., intended actions are executed as soon as possible; and (ii) *persistence*, i.e., unfulfilled intentions persist. This architecture was also used to enable an external observer to recognize the activity of an observed agent, i.e., for determining what has happened and what the agent intends to do (Gabaldon 2009). However, this architecture did not support the modeling of agents that desire to achieve specific goals. The more recent *Theory of Intentions* ($\mathcal{TI}$) (Blount, Gelfond, and Balduccini 2015; 2014) builds on (Baral and Gelfond 2005) to model the intentions of goal-driven agents. This theory expanded transition diagrams that have physical states and physically executable actions to include mental fluents and actions. It associated a sequence of agent actions with the goal it intended to achieve (called an "activity"), and introduced an *intentional agent* that only performs actions that are intended to achieve a desired goal and does so without delay. This theory has been used to create a methodology for understanding of narratives of typical and exceptional restaurant scenarios (Zhang and Inclezan 2017), and goal-driven agents in dynamic domains have been modeled using such activities (Saribatur, Baral, and Eiter 2017). A common requirement of such theories and their use is that all the domain knowledge, including the preconditions and effects of actions and potential goals, be known and encoded in the knowledge base, which is difficult to do in robot domains. Also, the set of states (and actions, observations) to be considered can be large in robot domains, which makes efficient reasoning a challenging task. In (Zhang and Inclezan 2017), the authors cluster indistinguishable states (Saribatur and Eiter 2016) but these clusters need to be encoded in advance. Furthermore, these approaches do not consider the uncertainty in sensing and actuation.

Many logic-based methods have been used in robotics, including those that also support probabilistic reasoning (Hanheide et al. 2017; Zhang, Sridharan, and Wyatt 2015). Methods based on first-order logic do not support non-monotonic logical reasoning or the desired expressiveness for capabilities such as default reasoning, e.g., it is not always meaningful to express degrees of belief by attaching probabilities to logic statements. Non-monotonic logics such as ASP address these limitations and have been used in cognitive robotics (Erdem and Patoglu 2012), but classical ASP formulations do not support the probabilistic models of uncertainty that are used by algorithms for sensing and actuation. Approaches based on logic programming also do not support one or more of the capabilities such as incremental addition of probabilistic information or variables to represent open worlds. Our prior refinement-based architecture reasoned with tightly-coupled transition diagrams at two resolutions, executing each abstract action in a coarse-resolution plan computed using ASP as a sequence of concrete actions computed by probabilistic reasoning over the relevant part of the fine-resolution diagram (Sridharan et al. 2017; Sridharan and Gelfond 2016). This paper explores the combination of these ideas and those drawn from $\mathcal{TI}$; specific differences from prior work are described below.

3 Cognitive Architecture

Figure 2 presents a block diagram of the overall architecture. Similar to prior work (Sridharan et al. 2017), this architecture may be viewed as consisting of three components: controller, logician and executor. In this paper, the controller is responsible for holding the overall beliefs of domain state, and for the transfer of control and information between all components. For any given goal, the logician performs non-monotonic logical reasoning with the coarse-resolution representation of commonsense knowledge to generate an activity, i.e., a sequence of intentional abstract actions. Each abstract action is implemented as a sequence of concrete actions. The executor uses probabilistic models of the uncertainty in sensing and actuation to execute each such action, with the outcomes (and relevant observations) being communicated to the controller and added to the coarse-resolution history of the logician. These components are described below, along with differences from prior work, using variants of the following illustrative domain.

Example 1 *[Robot Assistant (RA) Domain]* Consider a robot assisting humans in moving particular objects to specific locations in an indoor office domain with:

- Sorts such as `place`, `thing`, `robot`, `object`, and `book`, arranged hierarchically, e.g., `object` and `robot` are sub-

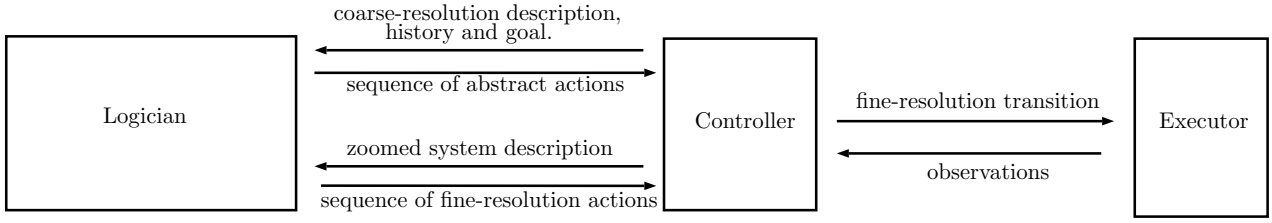


Figure 2: Architecture uses logic representation of the world at two different resolutions; may be viewed as interactions between a controller, logician, and executor.

sorts of thing. Sort names and constants are in lowercase, and variable names are in uppercase.

- Places: $\{\mathtt{office}_1, \mathtt{office}_2, \mathtt{kitchen}, \mathtt{library}\}$ with a door between neighboring places; only door between kitchen and library can be locked—Figure 3.
- Instances of sorts, e.g., $\mathtt{rob}_1$, $\mathtt{book}_1$, $\mathtt{book}_2$.
- Static attributes such as color, size and different parts (e.g., base and handle) associated with objects.
- Other agents that may influence the domain, e.g., move a book or lock a door. These agents are not modeled.

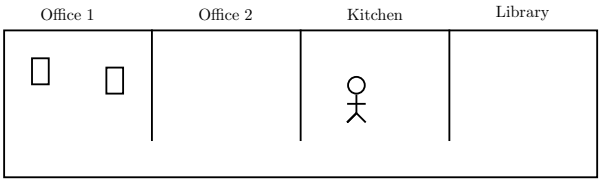


Figure 3: Four rooms considered in Example 1, with a human in the kitchen and two books in $\mathtt{office}_1$. Only the library's door can be locked; all other rooms are open at all times.

3.1 Action Language and Domain Representation

We first describe the action language encoding of domain dynamics, and its translation to CR-Prolog programs for knowledge representation and reasoning.

Action Language Action languages are formal models of parts of natural language used for describing transition diagrams of dynamic systems. We use action language $\mathcal{AL}_d$ (Gelfond and Inclezan 2013) to describe the transition diagrams at different resolutions. $\mathcal{AL}_d$ has a sorted signature with *statics*, *fluents* and *actions*. Statics are domain attributes whose truth values cannot be changed by actions, whereas fluents are domain attributes whose truth values can be changed by actions. Fluents can be *basic* or *defined*. Basic fluents obey the laws of inertia and can be changed by actions. Defined fluents do not obey the laws of inertia and are not changed directly by actions—their values depend on other fluents. Actions are defined as a set of elementary operations. A domain attribute *p* or its negation $\neg p$ is a *literal*. $\mathcal{AL}_d$ allows three types of statements:

a **causes** l_b **if** $p_0, \ldots, p_m$ (Causal law)

l **if** $p_0, \ldots, p_m$ (State constraint)

impossible $a_0, \ldots, a_k$ **if** $p_0, \ldots, p_m$ (Executability condition)

where a is an action, l is a literal, l_b is a basic literal, and $p_0, \ldots, p_m$ are domain literals.

Knowledge Representation The domain representation consists of system description $\mathcal{D}$, a collection of statements of $\mathcal{AL}_d$, and history $\mathcal{H}$. $\mathcal{D}$ has a sorted signature Σ and axioms that describe the transition diagram τ. Σ defines the basic sorts, domain attributes and actions. Basic sorts of the RA domain and some ground instances were introduced in Example 1. Domain attributes and actions are described in terms of their arguments' sorts. Statics of the RA domain include relations such as $\mathtt{next\_to}(\mathtt{place}, \mathtt{place})$, which describes the relative location of places in the domain; and relations representing object attributes such as color and size, e.g., $\mathtt{obj\_color}(\mathtt{object}, \mathtt{color})$. Fluents include $\mathtt{loc}(\mathtt{thing}, \mathtt{place})$, the location of the robot or domain objects; $\mathtt{in\_hand}(\mathtt{robot}, \mathtt{object})$, which denotes a particular object is in the robot's hand; and $\mathtt{locked}(\mathtt{place})$, which implies a particular place is locked. The locations of other agents, if any, are not changed by the robot's actions; these locations are inferred from observations obtained from other sensors. The domain's actions include $\mathtt{move}(\mathtt{robot}, \mathtt{place})$, $\mathtt{pickup}(\mathtt{robot}, \mathtt{object})$, $\mathtt{putdown}(\mathtt{robot}, \mathtt{object})$, and $\mathtt{unlock}(\mathtt{robot}, \mathtt{place})$; we also consider exogenous actions $\mathtt{exo\_move}(\mathtt{object}, \mathtt{place})$ and $\mathtt{exo\_lock}(\mathtt{place})$ for diagnostic reasoning. Σ also includes the sort step for temporal reasoning, and the relation $\mathtt{holds}(\mathtt{fluent}, \mathtt{step})$ to imply that a particular fluent holds at a particular time step.

Axioms for the RA domain include causal laws, constraints and executability conditions such as:

$\mathtt{move}(\mathtt{rob}_1, P)$ **causes** $\mathtt{loc}(\mathtt{rob}_1, P)$

$\mathtt{pickup}(\mathtt{rob}_1, O)$ **causes** $\mathtt{in\_hand}(\mathtt{rob}_1, O)$

$\neg\mathtt{loc}(Th, L_2)$ **if** $\mathtt{loc}(Th, L_1)$, $L_1 \neq L_2$

$\mathtt{loc}(O, P)$ **if** $\mathtt{loc}(\mathtt{rob}_1, P)$, $\mathtt{in\_hand}(\mathtt{rob}_1, O)$

impossible $\mathtt{pickup}(\mathtt{rob}_1, O)$ **if** $\mathtt{loc}(\mathtt{rob}_1, L_1)$, $\mathtt{loc}(O, L_2)$, $L_1 \neq L_2$

The history $\mathcal{H}$ of a dynamic domain is usually a record of fluents observed to be true or false at a particular time step, i.e., $\mathtt{obs}(\mathtt{fluent}, \mathtt{boolean}, \mathtt{step})$, and the occurrence of an action at a particular time step, i.e., $\mathtt{occurs}(\mathtt{action}, \mathtt{step})$. We expanded this notion to represent defaults describing the values of fluents in the initial state, e.g., "books are usually in the library and if it not there, they are normally in the office" is encoded as:

initial default $\mathtt{loc(X, library)}$ **if** $\mathtt{book(X)}$
initial default $\mathtt{loc(X, office_1)}$ **if** $\mathtt{book(X)}$, $\neg\mathtt{loc(X, library)}$

We can also encode exceptions to these defaults, e.g., "cookbooks are in the kitchen".

Reasoning The domain representation is translated into a program $\Pi(\mathcal{D}, \mathcal{H})$ in CR-Prolog[1], a variant of ASP that incorporates consistency restoring (CR) rules (Balduccini and Gelfond 2003). ASP is based on stable model semantics and is based on concepts such as *default negation* and *epistemic disjunction*, e.g., unlike "$\neg$a" that states *a is believed to be false*, "not a" only implies *a is not believed to be true*. ASP can represent recursive definitions and constructs that are difficult to express in classical logic formalisms, and it supports non-monotonic logical reasoning, i.e., it is able to revise previously held conclusions based on new evidence. An ASP program Π includes the signature and axioms of $\mathcal{D}$, inertia axioms, reality checks, and observations, actions, and defaults from $\mathcal{H}$. Every default also has a CR rule that allows the robot to assume the default's conclusion is false to restore consistency under exceptional circumstances. Algorithms for computing entailment, and for planning and diagnostics, reduce these tasks to computing *answer sets* of CR-Prolog programs. We compute answer sets using the language SPARC (Balai, Gelfond, and Zhang 2013).

3.2 Adapted Theory of Intention

For a given goal, a robot using ASP-based reasoning will compute a plan and execute it until the goal is achieved or an action in the plan has an unexpected outcome; in the latter case, the robot will attempt to explain the outcome (i.e., perform diagnostics) and compute a new plan if necessary. To motivate a need for a different approach in dynamic domains, consider the following scenarios in which the goal is to move $\mathtt{book_1}$ and $\mathtt{book_2}$ to the library; these have been adapted from scenarios considered in prior work (Blount, Gelfond, and Balduccini 2015):

- **Scenario 1 (planning):** Robot $\mathtt{rob_1}$ is in the kitchen holding $\mathtt{book_1}$, and believes $\mathtt{book_2}$ is in the kitchen and that the library is unlocked. The computed plan is: $\mathtt{move(rob_1, library)}$, $\mathtt{put\_down(rob_1, book_1)}$, $\mathtt{move(rob_1, kitchen)}$, $\mathtt{pickup(rob_1, book_2)}$, $\mathtt{move(rob_1, library)}$, $\mathtt{put\_down(rob_1, book_2)}$.
- **Scenario 2 (unexpected success):** Assume that $\mathtt{rob_1}$ in Scenario-1 has moved to the library and put $\mathtt{book_1}$ down, and observes $\mathtt{book_2}$ there. The robot should be able to explain this observation (e.g., $\mathtt{book_2}$ was moved there) and realize the goal has been achieved.
- **Scenario 3 (not expected to achieve goal, diagnose and replan, case 1):** Assume rob1 in Scenario-1 starts moving $\mathtt{book_1}$ to library, but observes $\mathtt{book_2}$ is not in the kitchen. Now, $\mathtt{rob_1}$ should realize the plan will fail, explain the observation and compute a new plan.
- **Scenario 4 (not expected to achieve goal, diagnose and replan, case 2):** Assume rob1 is in the kitchen holding book1, and believes book2 is in $\mathtt{office_2}$ and library is unlocked. The robot plans to first put $\mathtt{book_1}$ in the library before fetching $\mathtt{book_2}$ from $\mathtt{office_2}$. Before $\mathtt{rob_1}$ moves to the library, it suddenly observes $\mathtt{book_2}$ in the kitchen. Now, $\mathtt{rob_1}$ should realize the plan will fail, explain the observation and compute a new plan.
- **Scenario 5 (failure to achieve the goal, diagnose and replan):** Assume $\mathtt{rob_1}$ in Scenario-1 is putting $\mathtt{book_2}$ in the library, after having put $\mathtt{book_1}$ in the library earlier, and observes that $\mathtt{book_1}$ is no longer there. The robot's intention should persist; it should explain the observation(s), replan if necessary, and execute actions until the goal is achieved.

One way to support the desired behavior in such scenarios is to obtain and reason with all possible observations of domain objects and events (e.g., observations of all objects in the sensor's field of view) during plan execution. However, such an approach would be computationally intractable in complex domains. Instead, we build on the principles of non-procrastination and persistence and the ideas from $\mathcal{TI}$. Our architecture enables the robot to compute actions that are intended for any given goal and current beliefs. As the robot attempts to implement each such action, *it obtains all observations relevant to this action and the intended goal*, and adds these observations to the recorded history. We will henceforth use $\mathcal{ATI}$ to refer to this adapted theory of intention that expands both the system description $\mathcal{D}$ and history $\mathcal{H}$ in the original program $\Pi(\mathcal{D}, \mathcal{H})$. First, the signature Σ is expanded to represent an *activity*, a triplet of a *goal*, a *plan* to achieve the goal, and a specific *name*, by introducing relations such as:

$\mathtt{activity(name)}$, $\mathtt{activity\_goal(name, goal)}$
$\mathtt{activity\_length(name, length)}$
$\mathtt{activity\_component(name, number, action)}$

These relations represent each named activity, the goal and length of each activity, and the actions that are the components of the activity. Note that when these relations are ground, they are statics.

Next, the existing fluents of Σ are considered *physical fluents* and are expanded to include *mental fluents* such as:

$\mathtt{active\_activity(activity)}$, $\mathtt{in\_progress\_goal(goal)}$
$\mathtt{next\_action(activity, action)}$,
$\mathtt{in\_progress\_activity(activity)}$,
$\mathtt{active\_goal(goal)}$, $\mathtt{next\_activity\_name(name)}$
$\mathtt{current\_action\_index(activity, index)}$

where the relations in the first three lines are defined fluents, whereas the other relations correspond to basic fluents. These fluents represent the robot's belief about a particular activity, action or goal being active or in progress. None of these fluents' values are changed directly by executing any physical action. The value of *current_action_index* changes if the robot has completed an intended action or if a change

[1] We use the terms "ASP" and "CR-Prolog" interchangeably.

in the domain makes it impossible for an activity to succeed. The values of the other mental fluents are changed directly or indirectly by expanding the existing *physical actions* of Σ to include *mental actions* such as:

$$\texttt{start(name)},\ \texttt{stop(name)},\ \texttt{select(goal)}$$
$$\texttt{abandon(goal)}$$

where the first two mental actions are used by the controller to start or stop a particular activity, and the other two are exogenous actions exercised (e.g., by a human) to select or abandon a goal.

We also define new axioms in $\mathcal{AL}_d$, e.g., to represent the effects of actions, prevent certain outcomes, and generate intentional actions—we do not describe these here due to space constraints. The notion of history is also expanded to include statements such as:

$$\texttt{attempt(action, step)},\ \neg\ \texttt{hpd(action, step)}$$

which denote that a particular action was attempted at a particular time step, and that a particular action did not happen (i.e., was not executed successfully) at a particular time step. The revised system description and history are translated automatically to CR-Prolog program $\Pi(\mathcal{D}', \mathcal{H}')$ that is solved for planning or diagnostics. The complete program for the RA domain is available online (Software 2018).

Key differences between $\mathcal{ATI}$ and prior work on $\mathcal{TI}$ are:

- $\mathcal{TI}$ becomes computationally expensive, especially as the size of the plan or history increases. It also performs diagnostics and planning jointly, which allows it to consider different explanations during planning but increases computational cost in complex domains. $\mathcal{ATI}$, on the other hand, first builds a consistent model of history by considering different explanations, and *uses this model to guide planning*, significantly reducing computational cost in complex domains.
- $\mathcal{TI}$ assumes complete knowledge of the state of other agents (e.g., humans or other robots) that perform exogenous actions; in many robotics domains, this is an unrealistic assumption. $\mathcal{ATI}$ instead makes the more realistic assumption that the robot can only infer exogenous actions by reasoning with the observations that it obtains from sensor input.
- $\mathcal{ATI}$ does not include the notion of sub-goals and sub-activities (and associated relations) from $\mathcal{TI}$, as they were not necessary; these may be introduced later without loss of generality of our overall architecture.

Any architecture with $\mathcal{ATI}$, $\mathcal{TI}$ or any reasoning component based on logic-programming or classical first-order logic, often has two key limitations. First, reasoning does not scale well to the finer resolution required for many tasks to be performed by the robot. For instance, the coarse-resolution representation discussed so far is not sufficient if the robot has to grasp and pickup a particular object from a particular location, and reasoning logically over a sufficiently fine-grained domain representation will be computationally expensive. Second, we have not yet modeled the actual sensor-level observations of the robot or the uncertainty in sensing and actuation. Section 2 further discusses the limitations of other approaches based on logical and/or probabilistic reasoning for robotics domains. Our architecture seeks to address these limitations by combining $\mathcal{ATI}$ with ideas drawn from work on a refinement-based architecture (Sridharan et al. 2017).

3.3 Refinement, Zooming and Execution

Consider a coarse-resolution system description $\mathcal{D}_c$ of transition diagram τ_c that includes $\mathcal{ATI}$. For any given goal, reasoning with $\Pi(\mathcal{D}_c, \mathcal{H}_c)$, as described above, will provide an activity, i.e., a sequence of abstract intentional actions. In our architecture, the execution of the coarse-resolution transition corresponding to each such abstract action is based on a fine-resolution system description $\mathcal{D}_f$ of transition diagram τ_f, which is a *refinement* of, and is tightly coupled to, $\mathcal{D}_c$. We can imagine refinement as taking a closer look at the domain through a magnifying lens, potentially leading to the discovery of structures that were previously abstracted away by the designer (Sridharan et al. 2017). $\mathcal{D}_f$ is constructed automatically as a step in the design methodology using $\mathcal{D}_c$ and some domain-specific information that has to be provided by the designer.

The signature Σ_f of $\mathcal{D}_f$ includes each basic sort of $\mathcal{D}_c$ whose elements have not been *magnified* by the increase in resolution, or both the coarse-resolution copy and its fine-resolution *counterparts* for sorts with magnified elements. For instance, the sorts in the RA domain would include:

$$\texttt{place}^* = \{\texttt{office}_1, \texttt{office}_2, \texttt{kitchen}, \texttt{library}\}$$
$$\texttt{place} = \{c_1, \ldots, c_m\}$$
$$\texttt{cup}^* = \{\texttt{cup}_1\}$$
$$\texttt{cup} = \{\texttt{cup}_1\texttt{\_base}, \texttt{cup}_1\texttt{\_handle}\}$$
$$\texttt{book} = \{\texttt{book}_1, \texttt{book}_2\}$$

where $\{c_1, \ldots, c_m\}$ are the cells that are the components of the original set of places, and any cup has a base and handle as components; a book, on the other hand, is not magnified and has no components. We also include domain-dependent statics relating the magnified objects and their counterparts, e.g., component(cup_base, cup). Next, domain attributes of Σ_f include the coarse-resolution version and fine-resolution counterparts (if any) of each domain attribute of Σ_c. For instance, in the RA domain, Σ_f will include domain attributes such as:

$$\texttt{loc}^*(\texttt{thing}^*, \texttt{place}^*),\ \texttt{next\_to}^*(\texttt{place}^*, \texttt{place}^*)$$
$$\texttt{loc}(\texttt{thing}, \texttt{place}),\ \texttt{next\_to}(\texttt{place}, \texttt{place})$$

These relations describe the location of each thing at two different resolutions, and describe two places or cells that are next to each other. Actions of Σ_f include (a) every action in Σ_c with its magnified parameters replaced by fine-resolution counterparts; and (b) knowledge-producing action test(robot, fluent) that checks the value of a fluent in a given state. Finally, Σ_f includes *knowledge fluents* to describe observations of the environment and the axioms governing them, e.g., basic fluents to describe the direct (sensor-based) observation of the values of the fine-resolution fluents, and defined domain-dependent fluents that determine

when the value of a particular fluent can be tested. The test actions only change the values of knowledge fluents.

The axioms of $\mathcal{D}_f$ include (a) axioms of $\mathcal{D}_c$ with variables ranging over appropriate sorts from Σ_f; (b) axioms for observing the domain through sensor inputs; and (c) axioms relating coarse-resolution domain attributes with their fine-resolution counterparts. For example:

$$\begin{aligned}&\texttt{test}(\texttt{rob}_1, \mathsf{F}) \textbf{ causes } \texttt{dir\_obs}(\texttt{rob}_1, \mathsf{F}) \textbf{ if } \mathsf{F} = \texttt{true}\\ &\textbf{impossible } \texttt{test}(\texttt{rob}_1, \mathsf{F}) \textbf{ if } \neg\texttt{can\_test}(\texttt{rob}_1, \mathsf{F})\\ &\texttt{in\_hand}^*(\texttt{rob}_1, \mathsf{O}) \textbf{ if } \texttt{component}(\texttt{O\_base}, \mathsf{O}),\\ &\qquad\qquad \texttt{in\_hand}(\texttt{rob}_1, \texttt{O\_base})\end{aligned}$$

If certain conditions are met, e.g., each coarse-resolution domain attribute can be defined in terms of the fine-resolution attributes of the corresponding components, there is a path in τ_f for each transition in τ_c—see (Sridharan et al. 2017) for details.

Reasoning at fine resolution using $\mathcal{D}_f$ does not address the uncertainty in sensing and actuation, and becomes computationally intractable for complex domains. We address this problem by drawing on the principle of *zooming* introduced in (Sridharan et al. 2017). Specifically, for each abstract transition T to be implemented (i.e., executed) at fine resolution, we automatically determine the system description $\mathcal{D}_f(\mathsf{T})$ relevant to this transition; we do so by determining the relevant object constants and restricting $\mathcal{D}_f$ to these object constants. To implement T, we then use ASP-based reasoning with $\Pi(\mathcal{D}_f(\mathsf{T}), \mathcal{H}_f)$ to plan a sequence of *concrete* (i.e., fine-resolution) actions. In what follows, we use "refinement and zooming" to refer to the use of both refinement and zooming as described above. Note that fine-resolution reasoning does not (need to) reason with activities or intentional actions.

The actual execution of the plan of concrete action is based on existing implementations of algorithms for common robotics tasks such as motion planning, object recognition and localization; these algorithms use probabilistic models of uncertainty in sensing and actuation. The high-probability outcomes of each action's execution are elevated to statements associated with complete certainty in $\mathcal{H}_f$ and used for subsequent reasoning. The outcomes from fine-resolution execution of each abstract transition, along with relevant observations, are added to $\mathcal{H}_c$ for subsequent reasoning using $\mathcal{ATI}$. The CR-Prolog programs for fine-resolution reasoning and the program for the overall control loop are available online (Software 2018).

Key differences between the current representation and use of fine-resolution information, and prior work on the refinement-based architecture (Sridharan et al. 2017) are:

- Prior work used a partially observable Markov decision process (POMDP) to reason probabilistically over the zoomed fine-resolution system description $\mathcal{D}_f(\mathsf{T})$ for any coarse-resolution transition T; this can be computationally expensive, especially when domain changes prevent reuse of POMDP policies (Sridharan et al. 2017). In this paper, CR-Prolog is used to compute a plan of concrete actions from $\mathcal{D}_f(\mathsf{T})$; each concrete action is executed using algorithms that incorporate probabilistic models of uncertainty, significantly reducing the computational costs of fine-resolution planning and execution.
- Prior work did not (a) reason about intentional actions; (b) maintain any fine-resolution history; or (c) obtain and exploit all the information from fine-resolution observations. The architecture described in this paper keeps track of the relevant fine-resolution observations and adds appropriate statements to the coarse-resolution history to use all relevant information. It also explicitly builds a consistent model of history at the finer resolution.

4 Experimental Setup and Results

This section reports the results of experimentally evaluating the capabilities of our architecture in different scenarios. We evaluated the following hypotheses:

- **H1:** using $\mathcal{ATI}$ improves the computational efficiency in comparison with not using it, especially in scenarios with unexpected success.
- **H2:** using $\mathcal{ATI}$ improves the accuracy in comparison with not using it, especially in scenarios with unexpected goal-relevant observations.
- **H3:** the architecture that combines $\mathcal{ATI}$ with refinement and zooming supports reliable and efficient operation in complex robot domains.

We evaluated these hypotheses experimentally: (a) in a simulated domain based on Example 1; (b) on a Baxter robot manipulating objects on a tabletop; and (c) on a Turtlebot finding and moving objects in an indoor domain. We also provide some execution traces as illustrative examples of the working of the architecture. In each trial, the robot's goal was to find and move one or more objects to particular locations. As a baseline for comparison, we used an ASP-based reasoner that does not include $\mathcal{ATI}$—we refer to this as the "traditional planning" ($\mathcal{TP}$) approach in which only the outcome of the action currently being executed is monitored. Note that this baseline still uses refinement and zoom, and probabilistic models of the uncertainty in sensing and actuation. To evaluate the hypotheses, we used one or more of the following performance measures: (i) total planning and execution time; (ii) number of plans computed; (iii) planning time; (iv) execution time; (v) number of actions executed; and (vi) accuracy.

4.1 Experimental Results (Simulation)

We first evaluated hypotheses H1 and H2 extensively in a simulated world that mimics Example 1, with four places and different objects. Please also note the following:

- To fully explore the effects of $\mathcal{ATI}$, the simulation-based trials did not include refinement, i.e., the robot only reasons with the coarse-resolution domain representation. We also temporarily abstracted away uncertainty in perception and actuation.
- We conducted paired trials and compared the results obtained with $\mathcal{TP}$ and $\mathcal{ATI}$ for the same initial conditions and for the same dynamic domain changes (when appropriate), e.g., a book is moved unknown to the robot and the robot obtains an unexpected observation.

Scenarios	Average Ratios					Accuracy	
	Total Time	Number Plans	Planning Time	Exec. Time	Exec.Steps	$\mathcal{TP}$	$\mathcal{ATI}$
1	0.81	1.00	0.45	1.00	1.00	100%	100%
2	3.06	2.63	1.08	5.10	3.61	100%	100%
3	0.81	0.92	0.34	1.07	1.12	72%	100%
4	1.00	1.09	0.40	1.32	1.26	73%	100%
5	0.18	0.35	0.09	0.21	0.28	0%	100%
All	1.00	1.08	0.41	1.39	1.30	74%	100%
3 - no failures	1.00	1.11	0.42	1.32	1.39	100%	100%
4 - no failures	1.22	1.31	0.49	1.61	1.53	100%	100%
All - no failures	1.23	1.30	0.5	1.72	1.60	100%	100%

Table 1: Experimental results comparing $\mathcal{ATI}$ with $\mathcal{TP}$ in different scenarios. Values of all performance measures (except accuracy) for $\mathcal{TP}$ are expressed as a fraction of the values of the same measures for $\mathcal{ATI}$. We notice that $\mathcal{ATI}$ improves accuracy and computational efficiency, especially in dynamic domains.

- To measure execution time, we assumed a fixed execution time for each concrete action, e.g., 15 units for moving from a room to the neighboring room, 5 units to pick up an object or put it down; and 5 units to open a door. Ground truth is provided by a component that reasons with complete domain knowledge.

Table 1 summarizes the results of $\approx$ 800 paired trials in each scenario described in Section 3.2; all claims made below were tested for statistical significance. The initial conditions, e.g., starting location of the robot and objects' locations, and the goal were set randomly in each paired trial; the simulation ensures that the goal is reachable from the chosen initial conditions. Also, in suitable scenarios, a randomly-chosen, valid (unexpected) domain change is introduced in each paired trial. Given the differences between paired trials, it does not make sense to average the measured time or plan length across different trials. In each paired trial, the value of each performance measure (except accuracy) obtained with $\mathcal{TP}$ is thus expressed as a fraction of the value of the same performance measure obtained with $\mathcal{ATI}$; each value reported in Table 1 is the average of these computed ratios. We highlight some key results below.

Scenario-1 represents a standard planning task with no unexpected domain changes. Both $\mathcal{TP}$ and $\mathcal{ATI}$ provide the same accuracy (100%) and compute essentially the same plans, but reasoning with intentions to compute plans takes longer. This explains the reported average values of 0.45 and 0.81 for planning and total time in Table 1.

In Scenario-2 (unexpected success), both $\mathcal{TP}$ and $\mathcal{ATI}$ achieve 100% accuracy. Here, $\mathcal{ATI}$ stops reasoning and execution once it realizes the desired goal has been achieved unexpectedly. However, $\mathcal{TP}$ does not realize this because it does not consider observations not directly related to the action being executed; it keeps trying to find the objects of interest in different places. This explains why $\mathcal{TP}$ has a higher planning time and execution time, and also computes many more plans and executes more plan steps than $\mathcal{ATI}$.

Scenarios 3–5 correspond to different kinds of unexpected failures. In all trials corresponding to these scenarios, $\mathcal{ATI}$ leads to successful achievement of the goal, but there are a significant number of instances in which $\mathcal{TP}$ is unable to recover from the unexpected observations and achieve the goal. For instance, if the goal is to move two books to the library, and one of the books is moved to an unexpected location when it is no longer part of an action in the robot's plan, the robot may not reason about this unexpected occurrence and will thus fail to achieve the goal. This phenomenon is especially pronounced in Scenario-5 that represents an extreme case in which the robot using $\mathcal{TP}$ is never able to achieve the desired goal because it never realizes that it has failed to achieve the goal. Notice that in the trials corresponding to all three scenarios, $\mathcal{ATI}$ takes more time than $\mathcal{TP}$ to plan and execute the plans for any given goal, but this increase in time is more than justified given the high accuracy and the desired behavior that the robot is able to achieve in these scenarios using $\mathcal{ATI}$.

The row labeled "All" in Table 1 shows the average of the results obtained in the different scenarios. The following three rows in Table 1 summarize results after removing from consideration all trials in which $\mathcal{TP}$ fails to achieve the assigned goal. We then notice that $\mathcal{ATI}$ is at least as fast as $\mathcal{TP}$ and is often faster, i.e., takes less time (overall) to plan and execute actions to achieve the desired goal. In summary, $\mathcal{TP}$ results in faster planning but results in lower accuracy and higher execution time than $\mathcal{ATI}$ in dynamic domains, especially in the presence of unexpected successes and failures that are common in dynamic domains. All these results provide evidence in support of hypotheses H1 and H2.

4.2 Execution traces

The following execution traces illustrate the differences in the decisions made by a robot using $\mathcal{ATI}$ in comparison with a robot using $\mathcal{TP}$. These traces correspond to scenarios in which the robot has to respond to the observed effects of an exogenous action.

Execution Example 1 *[Example of Scenario-2]*
Assume that robot $\mathtt{rob}_1$ is in the kitchen initially, holding $\mathtt{book}_1$ in its hand, and believes that $\mathtt{book}_2$ is in $\mathtt{office}_2$ and the library is unlocked.

- The goal is to have $\mathtt{book}_1$ and $\mathtt{book}_2$ in the library. The computed plan is the same for $\mathcal{ATI}$ and $\mathcal{TP}$, and

consists of actions:

$move(rob_1, library)$, $put\_down(rob_1, book_1)$,
$move(rob_1, kitchen)$, $move(rob_1, office_2)$,
$pickup(rob_1, book_2)$, $move(rob_1, kitchen)$
$move(rob_1, library)$, $putdown(rob_1, book_2)$

- Assume that as the robot is putting $book_1$ down in the $library$, someone has moved $book_2$ to the $library$.
- With $\mathcal{ATI}$, the robot observes $book_2$ in the $library$, reasons and explains the observation as the result of an exogenous action, realizes the goal has been achieved and stops further planning and execution.
- With $\mathcal{TP}$, the robot does not observe or does not use the information encoded in the observation of $book_2$. It will thus waste time executing subsequent steps of the plan until it is unable to find or pickup $book_2$ in the $library$. It will then replan (potentially including prior observation of $book_2$) and eventually achieve the desired goal. It may also compute and pursue plans assuming $book_2$ is in different places, and take more time to achieve the goal.

Execution Example 2 *[Example of Scenario-5]*
Assume that robot rob_1 is in the $kitchen$ initially, holding $book_1$ in its hand, and believes that $book_2$ is in $kitchen$ and the $library$ is unlocked.

- The goal is to have $book_1$ and $book_2$ in the $library$. The computed plan is the same for $\mathcal{ATI}$ and $\mathcal{TP}$, and consists of the actions:

$move(rob_1, library)$, $put\_down(rob_1, book_1)$,
$move(rob_1, kitchen)$, $pickup(rob_1, book_2)$,
$move(rob_1, library)$, $putdown(rob_1, book_2)$

- Assume the robot is in the act of putting $book_2$ in the $library$, after having already put down $book_1$ in the $library$ earlier. However, someone has moved $book_1$ to the $kitchen$ while the robot was moving $book_2$.
- With $\mathcal{ATI}$, the robot observes $book_1$ in not in the $library$, realizes the goal has not been achieved, and continues to replan until it finds $book_1$ and moves it to the $library$.
- With $\mathcal{TP}$, the robot puts $book_2$ in the $library$ and stops execution because it believes it has achieved the desired goal. In other words, it does not even realize that the goal has not been achieved.

4.3 Robot Experiments

We also ran experimental trials with the combined architecture, i.e., $\mathcal{ATI}$ with refinement and zoom, on two different robot platforms. These trials represented instances of the different scenarios (in Section 3.2) in domains that are variants of the domain in Example 1.

First, consider the experiments with the Baxter robot manipulating objects on a tabletop as shown in Figure 1a. Some other details of the domain include:

- The goal is to move particular objects between different "zones" (instead of places) or particular cell locations on a tabletop.
- After refinement, each zone is magnified to obtain grid cells. Also, each object is magnified into parts such as $base$ and $handle$ after refinement.
- Objects are characterized by $color$ and $size$.
- The robot cannot move its body but it can use its arm to move objects between cells or zones.

Next, consider the experiments with the Turtlebot robot operating in an indoor domain as shown in Figure 1b. Some other details of the domain include:

- The goal is to find and move particular objects between places in an indoor domain.
- The robot does not have a manipulator arm; it solicits help from a human to pickup the desired object when it has reached the desired source location and found the object, and to put the object down when it has reached the desired target location.
- Objects are characterized by $color$ and $type$.
- After refinement, each place or zone was magnified to obtain grid cells. Also, each object is magnified into parts such as $base$ and $handle$ after refinement.

Although the two domains differ significantly, e.g., in the domain attributes, actions and complexity, no change is required in the architecture or the underlying methodology. Other than providing the domain-specific information, no human supervision is necessary; most of the other steps are automated. In ≈ 50 experimental trials in each domain, the robot using the combined architecture is able to successfully achieve the assigned goal. The performance is similar to that observed in the simulation trials. For instance, if we do not include $\mathcal{ATI}$, the robot has lower accuracy or takes more time to achieve the goal in the presence of unexpected success or failure; in other scenarios, the performance with $\mathcal{ATI}$ and $\mathcal{TP}$ is comparable. Also, if we do not include zooming, the robot takes a significantly longer to plan and execute concrete, i.e., fine-resolution actions. In fact, as the domain becomes more complex, i.e., there are many objects and achieving the desired goal requires plans with multiple steps, there are instances when the planning starts becoming computationally intractable. All these results provide evidence in support of hypothesis H3.

Videos of the trials on the Baxter robot and Turtlebot corresponding to different scenarios can be viewed online[2]. For instance, in one trial involving the Turtlebot, the goal is to have both a cup and a bottle in the $library$, and these objects and the robot are initially in $office_2$. The computed plan has the robot pick up the bottle, move to the $kitchen$, move to the $library$, put the bottle down, move back to the $kitchen$ and then to $office_2$, pick up the cup, move to the $library$ through the $kitchen$, and put the cup down. When the Turtlebot is moving to the $library$ holding the bottle, someone moves the cup to the $library$. With $\mathcal{ATI}$, the robot uses the observation of the cup, once it has put the bottle in the $library$, to infer the goal has been achieved

[2] https://drive.google.com/drive/u/1/folders/1cWXVib82K7qVSIP5i_cT7HEBfE5cGB4G

and thus stops planning and execution. With just $\mathcal{TP}$, the robot continued with its initial plan and realized there was a problem only when it went back to $\texttt{office}_2$ and did not find the cup.

Similarly, in one trial with the Baxter, the goal is to have blue and green blocks in zone Y (right side of the screen) and these blocks are initially in zone R (left side of the screen). The computed plan has the Baxter move its arm to zone R, pick up a block, move to zone G (center) then to zone Y to put the block down, and repeat this process until it has moved all blocks. When the Baxter has moved one block and is moving back to pick up the second block from zone R, an exogenous action puts the first block in zone G (center). With $\mathcal{ATI}$, as the Baxter is moving over zone G on the way to zone R, it observes the block (it had previously put in zone Y), performs diagnostics and realizes his current activity will not achieve the goal. It then re-plans and manages to move all blocks to zone Y. With $\mathcal{TP}$, the robot is not able to use the observation of the first block in zone G, continues with the initial plan and never realizes that the goal has not been achieved.

5 Discussion and Future Work

In this paper we presented a general architecture that reasons with intentions and beliefs using transition diagrams at two different resolutions. Non-monotonic logical reasoning with a coarse-resolution domain representation containing commonsense knowledge is used to provide a plan of abstract intentional actions for any given goal. Each such abstract intentional action is implemented as a sequence of concrete actions by reasoning with the relevant part of a fine-resolution representation that is a refinement of the coarse-resolution representation. Also, the architecture allows the robot to automatically and elegantly consider the observations that are relevant to any given goal and the underlying intention. Experimental results in simulation and on different robot platforms indicate that this architecture improves the accuracy and computational efficiency of decision making in comparison with an architecture that does not reason with intentional actions and/or does not include refinement and zooming.

This architecture opens up directions for future research. First, we will explore and formally establish the relationship between the different transition diagrams in this architecture, along the lines of the analysis provided in (Sridharan et al. 2017). This will enable us to prove correctness and provide other guarantees about the robot's performance. We will also instantiate the architecture in different domains and to further demonstrate the applicability of the architecture. The long-term goal will be enable robots to represent and reason reliably and efficiently with different descriptions of knowledge and uncertainty.

Acknowledgments

The authors thank Michael Gelfond for discussions related to the $\mathcal{TI}$ architecture (Blount, Gelfond, and Balduccini 2015) and his contributions to the refinement-based architecture (Sridharan et al. 2017) we build on in this paper. The authors also thank Evgenii Balai for providing support with the SPARC software.

References

Balai, E.; Gelfond, M.; and Zhang, Y. 2013. Towards Answer Set Programming with Sorts. In *International Conference on Logic Programming and Nonmonotonic Reasoning*.

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

Baral, C., and Gelfond, M. 2005. Reasoning about intended actions. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, 689.

Blount, J.; Gelfond, M.; and Balduccini, M. 2014. Towards a theory of intentional agents. In *Knowledge Representation and Reasoning in Robotics. AAAI Spring Symp. Series*, 10–17.

Blount, J.; Gelfond, M.; and Balduccini, M. 2015. A theory of intentions for intelligent agents. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 134–142. Springer.

Bratman, M. 1987. *Intention, Plans, and Practical Reason*. Center for the Study of Language and Information.

Erdem, E., and Patoglu, V. 2012. Applications of action languages in cognitive robotics. In *Correct Reasoning*. Springer. 229–246.

Gabaldon, A. 2009. Activity Recognition with Intended Actions. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gelfond, M., and Inclezan, D. 2013. Some Properties of System Descriptions of $\mathrm{AL_d}$. *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming* 23(1-2):105–120.

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

Hanheide, M.; Gobelbecker, M.; Horn, G.; Pronobis, A.; Sjoo, K.; Jensfelt, P.; Gretton, C.; Dearden, R.; Janicek, M.; Zender, H.; Kruijff, G.-J.; Hawes, N.; and Wyatt, J. 2017. Robot Task Planning and Explanation in Open and Uncertain Worlds. *Artificial Intelligence* 247:119–150.

Kelley, R.; Tavakkoli, A.; King, C.; Nicolescu, M.; Nicolescu, M.; and Bebis, G. 2008. Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots. In *International Conference on Human-Robot Interaction (HRI)*.

Rao, A. S., and Georgeff, M. P. 1995. BDI Agents: From Theory to Practice. In *First International Conference on Multiagent Systems*, 312–319.

Saribatur, Z. G., and Eiter, T. 2016. Reactive policies with planning for action languages. In Michael, L., and Kakas, A., eds., *Logics in Artificial Intelligence*, 463–480. Springer International Publishing.

Saribatur, Z. G.; Baral, C.; and Eiter, T. 2017. Reactive maintenance policies over equalized states in dynamic environments. In Oliveira, E.; Gama, J.; Vale, Z.; and Lopes Cardoso, H., eds., *Progress in Artificial Intelligence*, 709–723. Cham: Springer International Publishing.

2018. Software and Results for Architecture combining Theory of Intentions and Refinement. Retrieved March 2018 from `https://github.com/hril230/theoryofintentions/tree/master/simmulation`.

Sridharan, M., and Gelfond, M. 2016. Using knowledge representation and reasoning tools in the design of robots. In *IJCAI*.

Sridharan, M.; Gelfond, M.; Zhang, S.; and Wyatt, J. L. 2017. A refinement-based architecture for knowledge representation and reasoning in robotics. *CoRR* abs/1508.03891.

Zhang, Q., and Inclezan, D. 2017. An application of asp theories of intentions to understanding restaurant scenarios. *International Workshop on Practical Aspects of Answer Set Programming*.

Zhang, S.; Sridharan, M.; and Wyatt, J. 2015. Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds. *IEEE Transactions on Robotics* 31(3):699–713.