What do you really want to do? Representing and Reasoning with Intentional Actions on a Robot*

Rocio Gomez[1], Mohan Sridharan[2] and Heather Riley[3]

***Abstract*— This paper describes an architecture for robots to reason with intentional actions in complex domains. The architecture represents and reasons with tightly-coupled transition diagrams at two different resolutions. Non-monotonic logical reasoning with the 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. We illustrate the capabilities of this architecture in the context of a simulated domain that has a 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. We show that this architecture improves reliability and efficiency in comparison with a more traditional planning architecture that does not include intentional actions.**

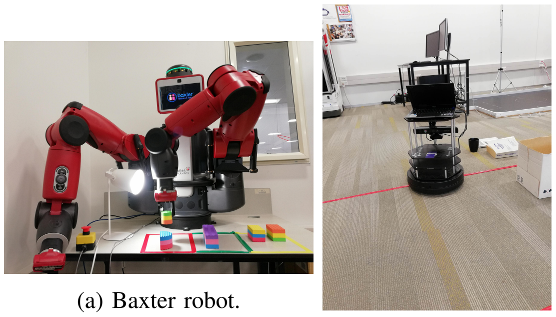
I. Introduction

Consider robots assisting humans in a variety of tasks in dynamic indoor domains, e.g., a robot assisting a human in arranging objects in different configurations on a tabletop in Figure 1a, or a robot delivering objects to particular locations in Figure 1b. These robots often have to reason with different descriptions of uncertainty and incomplete domain knowledge, e.g., commonsense knowledge, especially default knowledge that holds in all but a few exceptional circumstances ("books are usually in the library"), and probabilistic quantification of the uncertainty in sensing and actuation. 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, reasoning with incomplete knowledge can provide incorrect or suboptimal outcomes, but it is difficult to provide robots comprehensive domain knowledge or elaborate supervision.

The architecture described in this paper seeks to address these challenges by building on an agent architecture that uses declarative programming to reason about intended actions to achieve a given goal [1], and on an architecture for robots that reasons with tightly-coupled transition diagrams at different levels of abstraction [2]. 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. Non-monotonic logical reasoning at the coarse-resolution 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 that is defined as a refinement of a coarse-resolution system description. The outcomes of executing the concrete actions probabilistically are added to the coarse-resolution history.

In this paper, the coarse-resolution and fine-resolution action language description is translated to a program in CR-Prolog, an extension of Answer Set Prolog (ASP) [3], for commonsense reasoning; probabilistic execution of each concrete action 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 (Baxter) robot manipulating objects on a tabletop; and (iii) wheeled (Turtlebot) robot moving objects to specific 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 classical planning architecture that does not support reasoning about intentional actions.


(a) Baxter robot.

(b) Turtlebot.

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

II. Related Work

There is much work in the modeling and recognition of intentions, e.g., belief-desire-intention (BDI) architectures for reasoning agents model intention and guide reasoning by eliminating choices inconsistent with current intentions [4]. However, such architectures do not learn from past behavior, adapt to new situations, or include an explicit representation of (or reasoning about) goals.

*The University of Auckland, New Zealand
[1] m.gomez@auckland.ac.nz
[2] m.sridharan@auckland.ac.nz
[3] hril230@aucklanduni.ac.nz

An architecture formalizing intentions based on declarative programming was presented in [5]. 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 [6]. However, this architecture did not support the modeling of agents that desire to achieve specific goals. The more recent *Theory of Intentions* ($\mathcal{TI}$) [1], [7] builds on [5] to model the intentions of goal-driven agents. This theory expanded transition diagrams with physical states and physically executable actions to include mental fluents and actions, 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 [8], and goal-driven agents in dynamic domains have been modeled using such activities [9]. A common requirement of such theories and their use is that all the domain knowledge, including the preconditions and effect 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 challenging. In [8], the authors cluster indistinguishable states [10] but these clusters need to be encoded in advance. Furthermore, these approaches do not consider the uncertainty in sensing and actuation.

Many logic-based approaches have been used in robotics, including those that also support probabilistic reasoning [11], [12]. Algorithms based on first-order logic do not support non-monotonic logical reasoning. ASP addresses this limitation and has been used in cognitive robotics applications [13], [14]. The classical ASP formulation does not support probabilistic models of uncertainty, whereas many algorithms for sensing and actuation use such models. 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 [2], [15]. This paper explores the combination of ideas from $\mathcal{TI}$ and the refinement-based architecture for robotics applications; these ideas and differences from prior work are described below.

III. Cognitive Architecture

Figure 2 presents a block diagram of the overall architecture. Similar to prior work [2], this architecture may be viewed as consisting of three components: controller, logician and executor. 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; these actions are executed probabilistically by the executor, 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 subsorts of $thing$. Sort names and constants are in lower-case, and variable names are in uppercase.
- Places: $\{office_1, office_2, kitchen, library\}$ with a door between neighboring places; only door between $kitchen$ and $library$ can be locked—Figure 3.
- Instances of sorts, e.g., rob_1, $book_1$, $book_2$.
- 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.

A. Action Language and Domain Representation

Action languages are formal models of parts of natural language used for describing the transition diagram of dynamic systems. We use action language $\mathcal{AL}_d$ [16], which has a sorted signature with *actions*, *statics*, i.e., domain attributes whose values cannot be changed by actions, and *fluents*, i.e., domain attributes whose values can be changed by actions. Fluents can be *inertial* (and obey laws of inertia) or *defined*. A domain attribute *p* or its negation is a domain *literal*. $\mathcal{AL}_d$ allows three types of statements: *causal law*, *state constraint*, and *executability condition*; some examples below.

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 `next_to(place, place)`. Fluents include $loc(thing, place)$, the location of robot or objects—other agents' locations are not modeled; $in\_hand(robot, object)$, which denotes a particular object is in the robot's hand; and $locked(place)$, which implies a particular place is locked. Object attributes such as $color$ and $size$ are assumed to be static. Robot's actions include $move(robot, place)$, $pickup(robot, object)$, $putdown(robot, object)$, and $unlock(robot, place)$; we also consider exogenous actions $exo\_move(object, place)$ and $exo\_lock(place)$ for diagnostic reasoning. Relation $holds(fluent, step)$ implies a

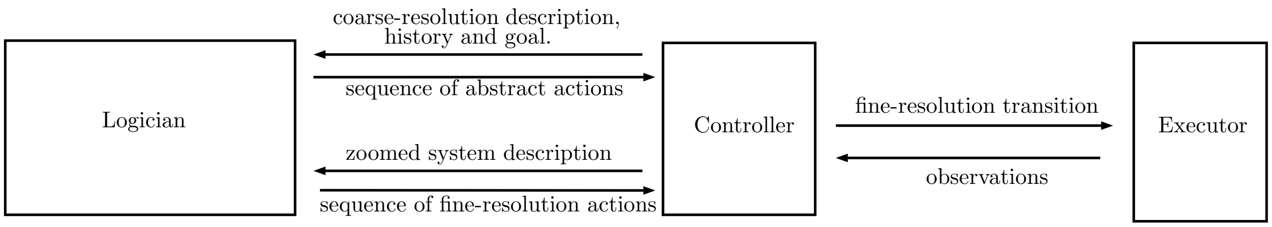

coarse-resolution description, history and goal.
sequence of abstract actions
zoomed system description
sequence of fine-resolution actions
Logician
Controller
fine-resolution transition
observations
Executor

Fig. 2: Architecture uses logic representation of the world at two levels of resolution

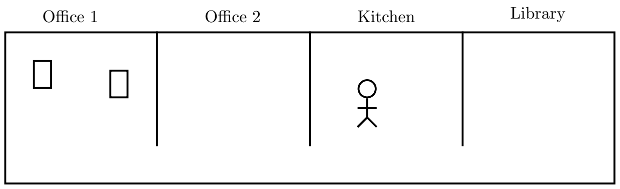


Fig. 3: Representation of four rooms in Example 3, with a unknown human in the kitchen and two books in office_1. Only door to library can be locked.

particular fluent is true at a timestep.

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

$move(rob_1, P)$ **causes** $loc(rob_1, P)$

$pickup(rob_1, O)$ **causes** $in\_hand(rob_1, O)$

$loc(O, P)$ **if** $loc(rob_1, P),\ in\_hand(rob_1, O)$

impossible $pickup(rob_1, O)$ **if** $loc(rob_1, L_1), loc(O, L_2)$

The history $\mathcal{H}$ of a dynamic domain is usually a record of fluents observed to be true or false at a timestep, and the occurrence of an action at a time step. We expand this notion to allow the representation of defaults describing the values of fluents in their initial states and exceptions (if any).

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 [17]. ASP is based on stable model semantics, and supports *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. Π includes the signature and axioms of $\mathcal{D}$, inertia axioms, reality checks, and observations, actions, and defaults from $\mathcal{H}$. Every default has a CR rule that allows the robot to assume the default's conclusion is false to restore consistency. 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 [18].

B. 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 has an unexpected outcome; in the latter case, the robot will try to explain the outcome and compute a new plan. To motivate a need for a different approach in dynamic domains, consider the following scenarios in which the goal is to move $book_1$ and $book_2$ to the library; these have been adapted from scenarios considered in prior work [1]:

- **Scenario 1 (planning):** Robot rob_1 is in the kitchen holding $book_1$, and believes $book_2$ is in the kitchen and library is unlocked. The computed plan is: move(rob_1, library), put_down(rob_1, $book_1$), move(rob_1, kitchen), pickup(rob_1, $book_2$), move(rob_1, library), put_down(rob_1, $book_2$).
- **Scenario 2 (unexpected success):** Assume that rob_1 in Scenario-1 has moved to the library and put $book_1$ down, and observes $book_2$ there. The robot should be able to explain this observation (e.g., $book_2$ 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 $book_1$ to library, but observes $book_2$ is not in the kitchen. Now, 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 $office_2$ and library is unlocked. The robot plans to first put $book_1$ in the library before fetching $book_2$ from $office_2$. Before rob_1 moves to the library, it suddenly observes $book_2$ in the kitchen. Now, 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 rob_1 in Scenario-1 is putting $book_2$ in the library, after having put $book_1$ in the library earlier, and observes that $book_1$ 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.

To support the desired behavior in such scenarios, 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 given any goal and its 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 history. We denote this adapted theory of intention component as $\mathcal{ATI}$. This component first extends Σ to represent an *activity*, a triplet of a *goal*, a *plan* to achieve the goal, and a *name*, by

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

introducing relations such as:

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

for named activities, and the actions, length and goal corresponding to an activity—when ground, these relations are statics. The existing fluents of Σ are considered *physical fluents* and are expanded to include *mental fluents*, e.g.:

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

where relations in the top two lines are defined fluents, whereas the others are inertial fluents. These fluents represent the robot's belief about a specific action, activity, or goal being active or in progress. None of these are changed directly by a physical action; the *current_action_index* can change if the robot has completed an intended action or if a change in the domain makes it impossible for an activity to succeed. Next, existing actions in Σ are considered *physical actions*, and expanded to include *mental actions* such as:

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

where the first two are used by the controller to start or stop an activity, and the other two are exogenous actions exercised (say, 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: `attempt(action, step)`, which denotes an action was attempted at a time step, and $\neg$ `hpd(action, step)`, which denotes an action did not happen at a 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 program for the RA domain is available online [19].

Key differences between $\mathcal{ATI}$ and $\mathcal{TI}$ 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.
- Unlike $\mathcal{TI}$, we make the realistic assumption that the robot does not have access to the state of the agents (e.g., humans) that may perform exogenous actions; any exogenous actions can only be inferred.
- $\mathcal{ATI}$ 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 using $\mathcal{TI}$, and others based on logic-programming or classical first-order logic, often have two key limitations. First, the coarse-resolution representation discussed so far (e..g, places and objects) is not sufficient for most tasks to be performed by the robot, e.g., to move to a particular location or grasp a particular object, and reasoning logically over a more fine-grained domain representation will be computationally intractable for complex domains. Second, we have not yet modeled the actual sensor-level observations of the robot or the uncertainty in sensing and actuation. Section II discusses the limitations of other approaches based on logical and/or probabilistic reasoning. Our architecture seeks to to address these limitations by combining $\mathcal{ATI}$ with ideas drawn from work on a refinement-based architecture [2].

C. Refinement, Zooming and Execution

Consider a coarse-resolution system description $\mathcal{D}_c$ of transition diagram τ_c that includes the $\mathcal{ATI}$. 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. The execution of the coarse-resolution transition corresponding to each such action is based on a fine-resolution system description $\mathcal{D}_f$ of transition diagram τ_f, which is a *refinement* of $\mathcal{D}_c$ and 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 [2]. $\mathcal{D}_f$ 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. In the RA domain, this would include the original set of places, places^*, and $\mathit{place} = \{c_1, \ldots, c_m\}$, i.e., cells in those places; cups may, in a similar manner, have a `base` and `handle`. 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) if each domain attribute of Σ_c. For instance, Σ_f will include both $\mathit{next\_to(place, place)}$ and $\mathit{next\_to}^*(\mathit{place}^*, \mathit{place}^*)$—the former describes two cells that are next to each other while the latter describes two rooms 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., inertial 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:

$$\begin{aligned}
&test(rob_1, F) \textbf{ causes } dir\_obs(rob_1, F) \textbf{ if } F = true\\
&\textbf{impossible } test(rob_1, F) \textbf{ if } \neg can\_test(rob_1, F)\\
&in\_hand^*(rob_1, O) \textbf{ if } component(O\_base, O),\\
&\qquad\qquad\qquad\qquad in\_hand(rob_1, O\_base)
\end{aligned}$$

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 [2] 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 [2]. Specifically, for each abstract transition T to be implemented (i.e., executed) at fine resolution, we automatically determine the system description $\mathcal{D}_f(T)$ 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 $\Pi(\mathcal{D}_f(T), \mathcal{H}_f)$ 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) use $\mathcal{ATI}$.

The actual execution of the plan of concrete action is based on existing implementations of probabilistic algorithms for common robotics tasks such as motion planning, object recognition and localization. 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 [19].

Key differences between the current representation and use of fine-resolution information, and prior work [2] are:

- Prior work used a partially observable Markov decision process (POMDP) to reason probabilistically over the zoomed fine-resolution system description $\mathcal{D}_f(T)$ for any coarse-resolution transition T; this can be computationally expensive, especially when domain changes prevent reuse of POMDP policies [2]. In this paper, CR-Prolog is used to compute a plan of concrete actions from $\mathcal{D}_f(T)$; each concrete action is executed using probabilistic algorithms that use the corresponding 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 fine-resolution.

IV. 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 this component, especially in scenarios with unexpected goal-relevant observations.
- **H3:** architecture that combines $\mathcal{ATI}$ with refinement and zooming supports reliable and efficient operation in complex robot domains.

To evaluate these hypotheses, we ran experimental trials: (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 to specific locations in an indoor domain. 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. 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.

A. 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 $\mathcal{ATI}$, we did not include refinement in these simulated trials, 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.
- 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

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 I: 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.

a door. Ground truth is provided by a component that reasons with complete domain knowledge.

Table I summarizes the results of ≈ 800 paired trials in each scenario described in Section III-B; 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 I 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 (100%) accuracy 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 I.

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, but this increase in time is justified by the high accuracy and the desired behavior in these scenarios.

The row labeled "All" summarizes the results across the different scenarios. The following three rows summarize results after removing 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 robot domains. All these results provide evidence in support of hypotheses H1 and H2.

B. *Execution traces*

The following execution traces exemplify the differences in decision making between the robot using intentional agent and a traditional planning agent when an exogenous action happens in the domain.

Execution Example 1: [Example of Scenario-2] Assume that robot rob_1 is in the `kitchen` initially, holding $book_1$ in its hand, and believes that $book_2$ is in $office_2$ 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 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 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 $\mathit{kitchen}$ and the $\mathit{library}$ is unlocked.

- The goal is to have book_1 and book_2 in the $\mathit{library}$. 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.

C. 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 III-B) 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 specific (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 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/region 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 can be 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 several 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 computes plan is for the robot to 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 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 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).

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

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.

V. Discussion and Future Work

In this paper we presented a generic architecture that reasons with transition diagrams at two 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 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. This architecture improves the accuracy and computational efficiency of decision making in different complex domains. In the future, we will explore and formally establish the relationship between the different transition diagrams in this architecture, along the lines of the analysis provided in [2]. 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 [1] and his contributions to the refinement-based architecture [2] we build on in this paper. The authors also thank Evgenii Balai for providing support with the SPARC software.

References

[1] J. Blount, M. Gelfond, and M. Balduccini, "A theory of intentions for intelligent agents," in *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, 2015, pp. 134–142.

[2] M. Sridharan, M. Gelfond, S. Zhang, and J. L. Wyatt, "A refinement-based architecture for knowledge representation and reasoning in robotics," *CoRR*, vol. abs/1508.03891, 2017. [Online]. Available: http://arxiv.org/abs/1508.03891

[3] M. Gelfond and Y. Kahl, *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, 2014. [Online]. Available: https://books.google.co.nz/books?id=99XSAgAAQBAJ

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

[5] C. Baral and M. Gelfond, "Reasoning about intended actions," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, 2005, p. 689.

[6] A. Gabaldon, "Activity recognition with intended actions." in *IJCAI*, 2009, pp. 1696–1701.

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

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

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

[10] Z. G. Saribatur and T. Eiter, "Reactive policies with planning for action languages," in *Logics in Artificial Intelligence*, L. Michael and A. Kakas, Eds. Springer International Publishing, 2016, pp. 463–480.

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

[12] S. Zhang, M. Sridharan, and J. Wyatt, "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 699–713, 2015.

[13] E. Erdem and V. Patoglu, "Applications of action languages in cognitive robotics," in *Correct Reasoning*. Springer, 2012, pp. 229–246.

[14] M. Balduccini, W. C. Regli, and D. N. Nguyen, "Towards an ASP-Based Architecture for Autonomous UAVs in Dynamic Environments (Extended Abstract)," in *International Conference on Logic Programming (ICLP)*, Vienna, Austria, July 19-22, 2014.

[15] M. Sridharan and M. Gelfond, "Using knowledge representation and reasoning tools in the design of robots." in *IJCAI*, 2016.

[16] M. Gelfond and D. Inclezan, "Some Properties of System Descriptions of $\mathcal{AL}_d$," *Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming*, vol. 23, no. 1-2, pp. 105–120, 2013.

[17] M. Balduccini and M. Gelfond, "Logic Programs with Consistency-Restoring Rules," in *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, 2003, pp. 9–18.

[18] E. Balai, M. Gelfond, and Y. Zhang, "Towards Answer Set Programming with Sorts," in *International Conference on Logic Programming and Nonmonotonic Reasoning*, Corunna, Spain, September 15-19, 2013.

[19] "Software and Results for Architecture combining Theory of Intentions and Refinement," 2018, retrieved March 2018 from https://github.com/hril230/theoryofintentions/tree/master/simmulation.