Towards a Theory of Intentions for Human-Robot Collaboration

Rocio Gomez[1], Mohan Sridharan[2][0000−0001−9922−8969], and Heather Riley[1]

[1] Electrical and Computer Engineering, The University of Auckland, New Zealand
m.gomez@auckland.ac.nz, hril230@aucklanduni.ac.nz
[2] School of Computer Science, University of Birmingham, United Kingdom
m.sridharan@bham.ac.uk

Abstract. The architecture described in this paper encodes a theory of intentions based on the principles of non-procrastination, persistence, and relevance. The architecture reasons with transition diagrams at two different resolutions, with the fine-resolution description defined as a refinement of, and hence tightly-coupled with, a coarse-resolution description. For any given goal, non-monotonic logical reasoning with the coarse-resolution description computes an activity, i.e., a plan, comprising a sequence of abstract actions to be executed to achieve the goal. Each abstract action is implemented as a sequence of concrete actions by automatically zooming to and reasoning with the part of the fine-resolution transition diagram relevant to the coarse-resolution transition and the goal. Each concrete action is executed using probabilistic models of the uncertainty in sensing and actuation, and the corresponding coarse-resolution observations are added to the coarse-resolution history. Experimental results in the context of simulated and physical robots indicate improvements in reliability and efficiency compared with an architecture that does not include the theory of intentions, and an architecture that does not include zooming for fine-resolution reasoning.

1 Introduction

Consider a robot[3] assisting humans in dynamic domains, e.g., a robot helping a human arrange objects in different configurations on a tabletop, or a robot delivering objects to particular places or people—see Figure 1. 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". The robot also receives a lot more sensor data than it can process, and it is equipped with many algorithms that compute and use a probabilistic quantification of the uncertainty in sensing and actuation, e.g., "I am 90% certain the robotics book is on the table". 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.

[3] A journal article based on this work has been accepted for publication in the *Annals of Mathematics and Artificial Intelligence* [11].

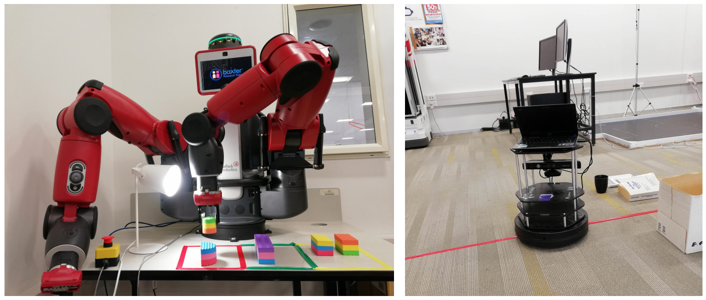
This loss in performance is more pronounced in scenarios corresponding to unexpected success or failure, which are common in dynamic domains. For instance, consider a robot trying to move two books from an office to a library. After moving the first book to the library, if the robot observes the second 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 is 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 [5], and on an architecture that reasons with tightly-coupled transition diagrams at different levels of abstraction [18]. This work has been described in detail in a recently published journal article [11]. Here, we describe the following key 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 and reasoning with 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) [9], 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 two resolutions. We demonstrate the capabilities of our architecture in the context of (i) a simulated robot assisting humans in an office domain; (ii) a physical robot (Baxter) manipulating objects on a tabletop; and (iii) a wheeled robot (Turtlebot) moving objects in an office domain. Experimental results indicate that the proposed architecture improves reliability and computational efficiency of planning and execution in dynamic domains in comparison with an 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 model the intentions of reasoning agents and guide reasoning by eliminating choices inconsistent with current intentions [6,14]. However, such architectures do not learn from past behavior, adapt to new situations, or include an explicit



(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.

representation of (or reasoning about) goals. Other work has reasoned with domain knowledge or used models learned from training samples to recognize intentions [13].

An architecture formalizing intentions based on declarative programming was described in [3]. 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 [8]. However, this architecture did not support the modeling of agents that desire to achieve specific goals. The *Theory of Intentions* ($\mathcal{TI}$) [5,4] builds on [3] to model the intentions of goal-driven agents. $\mathcal{TI}$ expanded transition diagrams that have physical states and physically executable actions to include mental fluents and mental actions. It associated a sequence of agent actions (called an "activity") with the goal it intended to achieve, 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 [20], and goal-driven agents in dynamic domains have been modeled using such activities [15]. 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 recent work [20], the authors attempt to address this problem by clustering indistinguishable states [16] but these clusters need to be encoded in advance. Furthermore, these approaches do not consider the uncertainty in sensing and actuation.

Logic-based methods have been used widely in robotics, including those that also support probabilistic reasoning [12,21]. 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

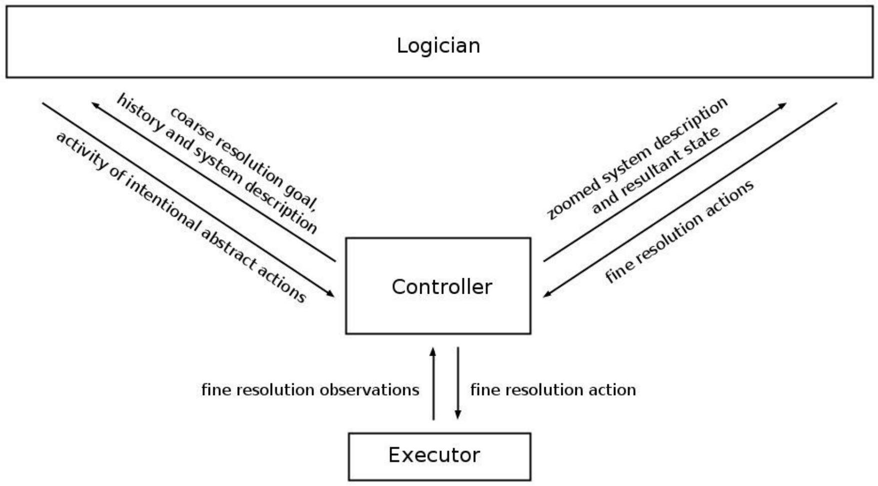


Fig. 2: Architecture combines the complementary strengths of declarative programming and probabilistic reasoning, representing intentions and beliefs as coupled transition diagrams at two resolutions; may be viewed as interactions between a controller, logician, and executor.

some of these limitations, and they have been used in cognitive robotics applications by an international research community [7]. However, classical ASP formulations do not support the probabilistic models of uncertainty that are used by algorithms for sensing and actuation in robotics. 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 reason about open worlds. Towards addressing these limitations, prior work in our group developed a refinement-based architecture that reasoned with tightly-coupled transition diagrams at two resolutions; each abstract action in a coarse-resolution plan computed using ASP was executed as a sequence of concrete actions computed by probabilistic reasoning over the relevant part of the fine-resolution diagram [18]. This paper explores the combination of these ideas with those drawn from $\mathcal{TI}$; specific differences from prior work are described in the relevant sections below.

3 Cognitive Architecture

Figure 2 presents a block diagram of the overall architecture. Similar to prior work [18], this architecture may be viewed as consisting of three components: a controller, a logician, and an executor. In this paper, the controller is responsible for holding the overall beliefs regarding 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 by zooming to and reasoning with a fine-resolution representation defined as a refinement of the coarse-resolution representation. The executor uses probabilistic models of the uncertainty in sensing and actuation to execute each concrete action, with the outcomes being communicated to the controller and added to the coarse-resolution history of the logician. These components of the architecture are described below, along with differences from prior work, using variants of the following illustrative domain.

Example Domain 1 *[Robot Assistant (RA) Domain]* Consider a robot assisting humans in moving particular objects to desired 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 $thing$.
- Places: $\{office_1, office_2, kitchen, library\}$ with a door between neighboring places—see Figure 3; only the door between $kitchen$ and $library$ can be locked.
- Instances of sorts, e.g., rob_1, $book_1$, $book_2$.
- Static attributes such as $color$, $size$ and parts (e.g., $base$ and $handle$) of objects. Other agents that may change the domain are not modeled.

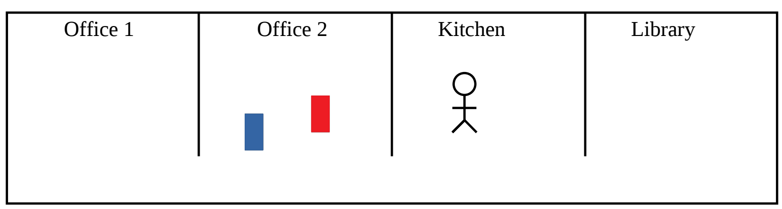


Fig. 3: Four rooms considered in Example 1, with a human in the $kitchen$ and two books in $office_1$. Only the library's door can be locked; all other rooms remain open.

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 $\mathcal{AL}_d$ [10] 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: causal law, state constraint, and executability condition.

Coarse-Resolution Knowledge Representation: The coarse-resolution domain representation consists of system description $\mathcal{D}_c$, a collection of statements of $\mathcal{AL}_d$, and history $\mathcal{H}_c$. System description $\mathcal{D}_c$ has a sorted signature Σ_c and axioms that describe the transition diagram τ_c. Σ_c defines the basic sorts, domain attributes and actions. Example 1 introduced some basic sorts and ground instances of the RA domain. Σ_c also includes the sort $step$ for temporal reasoning. Domain attributes (i.e., statics and fluents) and actions are described in terms of their arguments' sorts. In the RA domain, statics include relations such as $next\_to(place, place)$, which describes the relative location of places in the domain; and relations representing object attributes such as $color$ and $size$, e.g., $obj\_color(object, color)$. Fluents include $loc(thing, place)$, the location of the robot or domain objects; $in\_hand(robot, object)$, which denotes a particular object is in the robot's hand; and $locked(place)$, which implies a particular place is locked. 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 $move(robot, place)$, $pickup(robot, object)$, $putdown(robot, object)$, and $unlock(robot, place)$; we also consider exogenous actions $exo\_move(object, place)$ and $exo\_lock(place)$, which are used for diagnostic reasoning. Σ_c also includes the relation $holds(fluent, step)$ to imply that a particular fluent holds true at a particular time step. Axioms for the RA domain include causal laws, state constraints and executability conditions such as:

$$move(rob_1, P) \ \textbf{causes} \ loc(rob_1, P)$$
$$loc(O, P) \ \textbf{if} \ loc(rob_1, P), \ in\_hand(rob_1, O)$$
$$\textbf{impossible} \ pickup(rob_1, O) \ \textbf{if} \ loc(rob_1, L_1), \ loc(O, L_2), \ L_1 \neq L_2$$

The history $\mathcal{H}_c$ of the domain contains the usual record of fluents observed to be true or false at a particular time step, i.e., $obs(fluent, boolean, step)$, and the execution of an action at a particular time step, i.e., $occurs(action, step)$. In [18] this notion was expanded 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". We can also encode exceptions to these defaults, e.g., "cookbooks are in the kitchen". This representation, which does not quantitatively model beliefs in these defaults, supports elegant reasoning with generic defaults and their specific exceptions.

Reasoning: The coarse-resolution domain representation is translated into a program $\Pi(\mathcal{D}_c, \mathcal{H}_c)$ in CR-Prolog[4], a variant of ASP that incorporates consistency restoring (CR) rules [2]. ASP is based on stable model semantics and supports 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 $\mathcal{D}_c$, inertia axioms, reality checks, and observations, actions, and defaults from $\mathcal{H}_c$. 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. Each *answer*

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

set of an ASP program represents the set of beliefs of an agent associated with the program. Algorithms for computing entailment, and for planning and diagnostics, reduce these tasks to computing answer sets of CR-Prolog programs. We compute answer sets of CR-Prolog programs using the system called SPARC [1].

3.2 Adapted Theory of Intention

For any given goal, a robot using ASP-based reasoning will compute a plan and execute it until the goal is achieved or a planned action has an unexpected outcome; in the latter case, the robot will try to explain the outcome (i.e., diagnostics) and compute a new plan if necessary. To motivate the 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 scenarios have been adapted from scenarios in [5]:

- **Scenario 1 (planning):** Robot rob_1 is in the kitchen holding $book_1$, and believes $book_2$ is in the kitchen and the library is unlocked. The plan is: $move(rob_1, library)$, $put\_down(rob_1, book_1)$, $move(rob_1, kitchen)$, $pickup(rob_1, book_2)$, followed by $move(rob_1, library)$ and $put\_down(rob_1, book_2)$.
- **Scenario 2 (unexpected success):** Assume that rob_1 in Scenario-1 has moved to the $library$ and put $book_1$ down, and observes $book_2$. The robot should 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 $kitchen$. The robot should realize the plan will fail to achieve the overall goal, explain the unexpected 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 plan is to put $book_1$ in the $library$ before fetching $book_2$ from $office_2$. Before rob_1 moves to $library$, it observes $book_2$ in the $kitchen$. The robot should realize the plan will fail 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 unexpected observation, replan if necessary, and execute actions until the goal is achieved.

One way to support the desired behavior in such scenarios is to 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 $\mathcal{D}_c$ and history $\mathcal{H}_c$ in the original program $\Pi(\mathcal{D}_c, \mathcal{H}_c)$.

First, the signature Σ_c 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:

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

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

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

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

where the first four relations are defined fluents, and other relations are 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 in the domain makes it impossible for an activity to succeed. The values of other mental fluents are changed by expanding the set of existing *physical actions* of Σ to include *mental actions* such as $start(name)$, $stop(name)$, $select(goal)$, and $abandon(goal)$, 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 that are used (e.g., by human) to select or abandon a particular goal.

In addition to the signature Σ_c, history $\mathcal{H}_c$ is also expanded to include relations such as $attempt(action, step)$ and $\neg\ hpd(action, step)$, which denote that a particular action was attempted at a particular time step, and that a particular action was not executed successfully at a particular time step. Figuring out when an action was actually executed (or not executed) requires reasoning with observations of whether an action had the intended outcome(s).

We also introduce new axioms in $\mathcal{D}_c$, e.g., to represent the effects of the physical and mental actions on the physical and mental fluents, e.g., starting (stopping) an activity makes it active (inactive), and executing an action in an activity keeps the activity active. The new axioms also include state constraints, e.g., to describe when a particular activity or goal is active, and executability conditions, e.g., it is not possible for the robot to simultaneously execute two mental actions. In addition, axioms are introduced to generate intentional actions, build a consistent model of the domain history, and to perform diagnostics.

The revised system description $\mathcal{D}'_c$ and history $\mathcal{H}'_c$ are translated automatically to CR-Prolog program $\Pi(\mathcal{D}'_c, \mathcal{H}'_c)$ that is solved for planning or diagnostics. The complete program for the RA domain is available online [17]. 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 assumption is rather unrealistic. $\mathcal{ATI}$ instead makes the more realistic assumption that the robot can only infer exogenous actions by reasoning with the observations that it obtains from sensors.
- $\mathcal{ATI}$ does not include the notion of sub-goals and sub-activities (and associated relations) from $\mathcal{TI}$, as they were not necessary. Also, the sub-activities and sub-goals will need to be encoded in advance, and reasoning with these relations will also increase computational complexity in many situations. The inclusion of sub-activities and sub-goals will be explored in future work.

Any architecture with $\mathcal{ATI}$, $\mathcal{TI}$, or a different reasoning component based on logic-programming or classical first-order logic, 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 [18].

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 $\Pi(\mathcal{D}_c, \mathcal{H}_c)$ 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 [18]. $\mathcal{D}_f$ is constructed automatically as a step in the design methodology using $\mathcal{D}'_c$ and some domain-specific information provided by the designer.

First, 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, sorts in the RA domain include cells that are components of the original set of places, and any cup has a $base$ and $handle$ as components; any $book$, 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 include domain attributes, e.g.: $loc^*(thing^*, place^*)$, $next\_to^*(place^*, place^*)$, $loc(thing, place)$, and $next\_to(place, place)$, where relations with and without the "*" represent the coarse-resolution counterparts and fine-resolution counterparts respectively. The specific relations listed above 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) coarse-resolution and fine-resolution counterparts of all state constraints of $\mathcal{D}_c$, and fine-resolution counterparts of all other axioms of $\mathcal{D}_c$, with variables ranging over appropriate sorts from Σ_f; (b) general and domain-specific axioms for observing the domain through sensor inputs; and (c) axioms relating coarse-resolution domain attributes with their fine-resolution counterparts. 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 [18] for formal definitions and proofs.

Reasoning with $\mathcal{D}_f$ 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 [18]. Specifically, for each abstract transition T to be implemented 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) 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, grasping 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 of the architecture are available online [17].

Key differences between the current representation and use of fine-resolution information, and the prior work on the refinement-based architecture [18] 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, espe-

cially when domain changes prevent reuse of POMDP policies [18]. 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 algorithms that incorporate probabilistic models of uncertainty, significantly reducing the computational costs of fine-resolution planning and execution. The disadvantage is that the uncertainty associated with each algorithm is not considered explicitly during planning at the fine-resolution.
– Prior work did not (a) reason about intentional actions; (b) maintain any fine-resolution history; or (c) extract 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 the 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 report results of evaluating 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. Also, we do not use $\mathcal{TI}$ as the baseline because it includes components that make it much more computationally expensive than $\mathcal{ATI}$—see Section 3.2 for more details. 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.

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 $\mathcal{ATI}$. $\mathcal{ATI}$ improves accuracy and computational efficiency, especially in dynamic domains.

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

Table 1 summarizes the results of ≈ 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 plan, but computing plans comprising intentional actions takes longer. This explains the reported average values of 0.45 and 0.81 for planning time and total time (for $\mathcal{TP}$) 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, computes more plans, and executes more plan steps.

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 many 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 thus not 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 assigned 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 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 often faster, i.e., takes less time (overall) to plan and execute actions. 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. For extensive results in more complex domains, including a comparison with an architecture that does not use zooming at the fine-resolution, please see [11].

4.2 Execution Trace

The following execution trace illustrates the differences in the decisions made by a robot using $\mathcal{ATI}$ in comparison with a robot using $\mathcal{TP}$. This trace corresponds 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 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 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.

4.3 Robot Experiments

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

First, consider the experiments with the Baxter robot manipulating objects on a tabletop. 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. 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 [19]. 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 and thus stops planning and execution. With just $\mathcal{TP}$, the robot continued with its initial plan and realized that there was a problem (unexpected position of the cup) only when it went back to $office_2$ and did not find the cup.

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 [18]. 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.

References

1. Balai, E., Gelfond, M., Zhang, Y.: Towards Answer Set Programming with Sorts. In: International Conference on Logic Programming and Nonmonotonic Reasoning. Corunna, Spain (September 15-19, 2013)
2. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning. pp. 9–18 (2003)
3. Baral, C., Gelfond, M.: Reasoning about intended actions. In: Proceedings of the National Conference on Artificial Intelligence. vol. 20, p. 689 (2005)
4. Blount, J., Gelfond, M., Balduccini, M.: Towards a theory of intentional agents. In: Knowledge Representation and Reasoning in Robotics. AAAI Spring Symp. Series. pp. 10–17 (2014)
5. Blount, J., Gelfond, M., Balduccini, M.: A theory of intentions for intelligent agents. In: International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 134–142. Springer (2015)
6. Bratman, M.: Intention, Plans, and Practical Reason. Center for the Study of Language and Information (1987)

7. Erdem, E., Patoglu, V.: Applications of ASP in Robotics. Kunstliche Intelligenz **32**(2-3), 143–149 (2018)
8. Gabaldon, A.: Activity Recognition with Intended Actions. In: International Joint Conference on Artificial Intelligence (IJCAI). Pasadena, USA (July 11-17 2009)
9. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach. Cambridge University Press (2014), https://books.google.co.nz/books?id=99XSAgAAQBAJ
10. Gelfond, M., Inclezan, D.: Some Properties of System Descriptions of AL_d. Journal of Applied Non-Classical Logics, Special Issue on Equilibrium Logic and Answer Set Programming **23**(1-2), 105–120 (2013)
11. Gomez, R., Sridharan, M., Riley, H.: What do you really want to do? Towards a Theory of Intentions for Human-Robot Collaboration. Annals of Mathematics and Artificial Intelligence, special issue on commonsense reasoning (2020)
12. 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., Wyatt, J.: Robot Task Planning and Explanation in Open and Uncertain Worlds. Artificial Intelligence **247**, 119–150 (June 2017)
13. Kelley, R., Tavakkoli, A., King, C., Nicolescu, M., Nicolescu, M., Bebis, G.: Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots. In: International Conference on Human-Robot Interaction (HRI). Amsterdam, Netherlands (March 12-15, 2008)
14. Rao, A.S., Georgeff, M.P.: BDI Agents: From Theory to Practice. In: First International Conference on Multiagent Systems. pp. 312–319. San Francisco, CA (1995)
15. Saribatur, Z.G., Baral, C., Eiter, T.: Reactive maintenance policies over equalized states in dynamic environments. In: Oliveira, E., Gama, J., Vale, Z., Lopes Cardoso, H. (eds.) Progress in Artificial Intelligence. pp. 709–723. Springer International Publishing, Cham (2017)
16. Saribatur, Z.G., Eiter, T.: Reactive policies with planning for action languages. In: Michael, L., Kakas, A. (eds.) Logics in Artificial Intelligence. pp. 463–480. Springer International Publishing (2016)
17. Software and results corresponding to the evaluation of our architecture (2019), https://github.com/hril230/theoryofintentions/tree/master/code
18. Sridharan, M., Gelfond, M., Zhang, S., Wyatt, J.: REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. Journal of Artificial Intelligence Research **65**, 87–180 (May 2019)
19. Videos demonstrating the use of our architecture on robot platforms (2019), https://drive.google.com/open?id=1m-jVV25vFvi35Ai9N7RYFFOPNaZIUdpZ
20. Zhang, Q., Inclezan, D.: An application of asp theories of intentions to understanding restaurant scenarios. International Workshop on Practical Aspects of Answer Set Programming (2017)
21. Zhang, S., Khandelwal, P., Stone, P.: Dynamically Constructed (PO)MDPs for Adaptive Robot Planning. In: AAAI Conference on Artificial Intelligence (AAAI). San Francisco, USA (February 2017)