Planning with Task-Oriented Knowledge Acquisition for a Service Robot

Kai Chen,[1*] **Fangkai Yang,**[2†] and **Xiaoping Chen**[3*]
[1,3]School of Computer Science and Technology, University of Science and Technology of China
96 Jinzhai Rd, Hefei, Anhui, 200027, China
[1]chk0105@mail.ustc.edu.cn, [3]xpchen@ustc.edu.cn
[2] Katy Drilling Software Center, Schlumberger Software Technology, Schlumberger Ltd.
23500 Colonial Parkway, Katy, TX, 77493, USA
fyang10@slb.com

Abstract

We propose a framework for a service robot to behave intelligently in domains that contain incomplete information, underspecified goals and dynamic change. Human robot interaction (HRI), sensing actions and physical actions are uniformly formalized in action language $\mathcal{BC}$. An answer set solver is called to generate plans that guide the robot to acquire task-oriented knowledge and execute actions to achieve its goal, including interacting with human to gather information and sensing the environment to help motion planning. By continuously interpreting and grounding useful sensing information, robot is able to use contingent knowledge to adapt to unexpected changes and faults. We evaluate the approach on service robot *KeJia* that serves drink to guests, a testing benchmark for general-purpose service robot proposed by RoboCup@Home competition.

1 Introduction

Research on domestic service robots has received increasing attention in recent years. A domestic service robot scenario combines the research on autonomous robots, human-robot interaction (HRI), computer vision, motion control and automated planning. Typical tasks of a domestic service robot include taking orders and serving drinks, welcoming and guiding guests, or just cleaning up [Wachsmuth *et al.*, 2015]. To measure and compare the performance of such robotics system, starting 2006, RoboCup@Home, a part of RoboCup initiative (www.robocup.org), defines a series of tests and one of the most challenging tests is General-Purpose Service Robot (GPSR). An operator verbally specifies a complex, usually partially defined task to the robot that may request any skill, for instance, "serve Pepsi to Alice". The robot, which only has brief knowledge about the domain, needs to perform it, report any problems, adapt to unexpected changes and find alternative solutions.

*This work is supported by the National Natural Science Foundation of China under grant 61175057.

†Corresponding author.

Automated planning has been widely used for task planning and control in many service robot applications. This approach typically uses a declarative domain representation coupled with a general-purpose planner that produces a sequence of actions for the robot to execute, while fault recovery is usually handled by execution monitor and replanning. They usually assume a fully specified, static and predictable environment, which, however, generally does not hold for GPSR domains. Approaches of using effective HRI and sensing to facilitate planning under incomplete information and uncertainty have also been proposed.

In this paper we are interested in developing a service robot whose task planning integrates task-oriented knowledge acquisition. We believe acquiring task-relevant knowledge proactively in the context of specific planning problem will lead to a more robust service robot facing incomplete domain, uncertainty and faults. We particularly focus on three kinds of knowledge. (i) Domain knowledge. The robot only has limited amount of information about the domain or receives underspecified goal. In order to perform its task to serve Pepsi whose location is unknown, the robot needs to figure out that "location" is a piece of missing information (instead of "color", for example, in this specific plan), and acquires it by asking human. (ii) Control knowledge. When the robot is facing a dining table and about to pick up Pepsi, it will measure the distance of the object to determine if it should move closer, adjust its gripper, or simply fetch. The robot needs to figure out at planning time that the distance of objects may affect its manipulation strategy. (iii) Contingent knowledge. Throughout performing the task, the robot should continuously observe the environment, gather useful information, enrich its knowledge and adapt to the change. This is particularly important because the objects in domestic environment keep changing and the information provided by human can be vague or erroneous. Consequently, the robot has to start from a partial, incomplete and unreliable domain representation to generate plans to gather more information, in order to achieve its goal.

Planning with sensing actions and HRI has been addressed separately by different approaches. A service robot system involves complex skills such as speech, vision, navigation and object manipulation. Is there a uniform way to treat HRI,

sensing and the robot's physical actions together in a formal representation and the classical "plan-execute-monitor-replan" loop? In this paper, we propose such a framework and use it to control a service robot to robustly serve people drinks in dynamic and changing domains. Our method features three novelties:

- First, we propose a general modeling methodology of using $\mathcal{BC}$ [Lee *et al.*, 2013], an action language based on answer set programming [Lifschitz, 2008] that formalizes transition graphs, to represent the world state, the robot's belief state and how they are affected by HRI actions, sensing actions and physical actions. Once a goal is given, even if it is underspecified, the plan includes proper HRI and sensing actions to achieve the goal.
- Second, to acquire and interpret interested contingent information and adapt to changing environment in real time, we add a continuous observation mechanism to the execution loop to acquire contingent knowledge. Sensing modules continuously update the robot's belief state based on sensing results. The robot performs reasoning to validate current plan in presence of the new information, and make adjustment if needed. Contingent knowledge also helps subsequent tasks because the robot gets more information about the domain from each task.
- Third, adding HRI and sensing to the plan execution loop introduces the challenge of grounding execution results into "epistemic states". For instance, when Alice said "the Pepsi is in the kitchen" while the robot actually found it unreachable, or simply did not find it, how should we represent this state symbolically? What is the state to represent the belief when an object that the robot detected earlier was later found disappeared? We demonstrate how we handle symbol grounding for various realistic situations with unexpected change.

ASP-based language $\mathcal{BC}$ allows both domain knowledge and effects of actions to be represented altogether in uniform semantics. Different kinds of reasoning and planning tasks can all be supported by calling efficient answer set solvers. The framework is implemented and tested in robot *KeJia* [Chen *et al.*, 2014], the champion of RoboCup@Home competition in 2014 and 2nd in 2015, reaching architectural simplicity, technical soundness and more robustness in GPSR domains.

After reviewing related work, we present an overview of the architecture in Section 3. We explain domain formalization and control loop for plan execution and continuous sensing in Section 4, with details about continuous sensing presented in Section 5 and action controller in Section 6. The experiment results are shown in Section 7 and the paper is concluded in Section 8.

2 Related Work

State machine [Bohren and Cousins, 2010; Srinivasa *et al.*, 2012] is a classical method to implement domestic service robots. It generally requires all steps to accomplish the goal be specified and known beforehand. For more complex domain, the state machine can be highly complex and difficult to scale. For this reason, a general-purpose task planner with a domain formalization becomes a common component for many service robot applications [Chen *et al.*, 2010; Erdem *et al.*, 2012; Hanheide *et al.*, 2015]. In particular, action language $\mathcal{BC}$ has been used in a few applications [Khandelwal *et al.*, 2014; Thomason *et al.*, 2015].

Methods of HRI and planning with sensing actions can address incomplete information and have been developed from separate communities. [Puigbo *et al.*, 2015; Thomason *et al.*, 2015] implemented HRI as a part of natural language processing (NLP) module of the robot, in which the semantic parser decided which information to collect from the user during utterance interpretation. It has been proposed that task-oriented HRI can be achieved by combining HRI with planning directly. Those methods include using planning to interpret utterance and translate it into proper goal representation [Brenner *et al.*, 2007; Brenner, 2007], extending planning algorithm [Brenner and Nebel, 2009], or directly representing HRI actions using action language [Khandelwal *et al.*, 2014; Petrick and Foster, 2013]. Planning with sensing actions has been investigated under different semantics and specific planning algorithms [Son and Baral, 2001; Petrick and Bacchus, 2002; Son *et al.*, 2004; Hoffmann and Brafman, 2005]. Partially inspired by [Khandelwal *et al.*, 2014], we propose a uniform treatment of HRI, sensing and physical actions, and significantly simplify the complexity of implementing a service robot with many different skills. The plan generated from an incomplete domain representation is similar to a contingent plan [Hoffmann and Brafman, 2005], but it is not a complete tree structure of exponential size that captures all possible outcomes of sensing and HRI actions. Instead, it is the shortest of all plans that satisfies the goal for a possible initial state.

Automated planning has been used to implement robot manipulators [Dearden and Burbridge, 2013; Havur *et al.*, 2013; Srivastava *et al.*, 2014]. These work focus on the integration of a symbolic planner with motion control with numeric states, and generally assume complete knowledge of the domain. One exception is [Srivastava *et al.*, 2015], in which plan with loops can be generated to accommodate unknown information at planning time. By comparison, our approach utilizes execution monitor and replanning, a common control loop for a knowledge-based robot architecture.

3 Framework Overview

Shown in Figure 1, *domain formalization* in $\mathcal{BC}$ represents type hierarchy, domain objects and causal laws. The causal laws formalize effects of physical actions, HRI actions and sensing actions on fluents that represent world state and the robot's belief state. Initial domain objects are minimal and are enriched when new information is discovered. Human operator verbally specifies a goal which is translated by the semantic parser into the goal representation. The robot uses its sensor to obtain available world state and initialize its belief state. They constitute of the initial *current state* of a planning problem. After that, the planner, which is actually an *answer set solver*, is called to generate a plan that consists of a sequence of actions, and a sequence of states before and after the execution of each action.

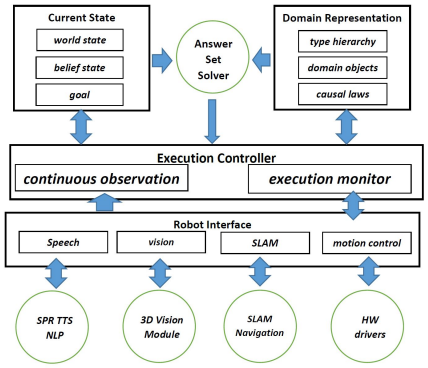

Figure 1: The architecture overview

Actions and states are sent to the *execution controller*. *Execution monitor* and *continuous observation* start simultaneously. Continuous observation constantly receives inputs from *robot interface* that senses domain information, grounds sensing inputs into symbols and updates current state. If update happens, it triggers a background verification process to validate current plan, by calling answer set solver to perform temporal projection reasoning, and sets replan flag if validation fails. Execution monitor sends action commands to controllers in robot interface too, which operates hardware to perform actions and returns grounded HRI, physical, and sensing results. Execution monitor compares the results with the expected state. If discrepancy is detected, it calls for replan.

Action controllers in robot interface directly operate low-level functional modules. Functional modules generally represent domain information in numerical states, and symbolic states are obtained through symbol grounding policies. It is crucial to maintain a correct mapping between low-level numerical states and high-level symbolic states for the robot to behave consistently and correctly.

4 Domain Formalization and Plan Execution

We present a general methodology and examples of representing a domestic environment with rooms, furniture, drink and people in which a service robot serves people drinks, a typical GPSR testing domain in RoboCup competitions. Initially Alice may request service by saying "bring me Pepsi". The robot has no knowledge about the object locations, so it may ask "where is the Pepsi". Alice may tell the robot a specific location "on the dining table", a vague location "in the dining room", an erroneous location "on the cabinet", or simply say "I don't know". The robot utilizes this information to find Pepsi, grasp it, put it in the basket, and bring it to Alice. The robot needs to adapt to changing environment and recover from various software and hardware failures.

4.1 Formalization in $\mathcal{BC}$

Domain knowledge consists of (i) types, relations, objects and (ii) causal laws that formalize the transition system. Causal laws are divided into *static laws*, which describe how the value of a fluent is determined by other fluents, and *dynamic laws*, which describe how the values of the fluents change from one state to another.

Types. We use a rigid (time-independent) fluent to denote type and object membership. For instance, we denote *pepsi1* is a member of type *pepsi* by *pepsi(pepsi1)*. Type hierarchy and relations are formalized using static laws such as $obj(X)$ **if** $pepsi(X)$.

Causal Laws. Non-rigid fluents consists of two groups:

- *world state* is a collection of fluents that represent the properties of the current world. It is further divided into *definite world state* and *possible world state*. Definite world state is affected by physical actions, such as moving affects the location of the robot. Possible world state represents possible value of an unknown property of an object. Possible world fluents are justified by belief fluents when actual HRI and sensing occur.
- *belief state* consists of fluents that represent the robot's belief. For instance, the robot knows a bottle of coke is located in the kitchen, or the robot knows an object is within reachable distance to grasp. Belief state is affected directly by HRI actions and sensing actions. It is also updated by continuous observation.

The combination of fluents can be used to represent various "epistemic states". For instance, {*objloc(pepsi1,kitchen), objloc(pepsi1,diningroom), -believeobjloc(pepsi1,R)*} for all rooms R means it is possible for *pepsi1* to be in kitchen or in dining room, but the robot does not know it. They are important for symbol grounding during continuous observation and execution (Section 5 and Section 6). Due to space limit, below we give examples of formalizing HRI actions and sensing actions, as formalizing physical actions using action languages has been well studied [Khandelwal *et al.*, 2014].

Fluents	Definite World State: *roboloc*, *ordertaken*, *gripperempty*, *lifted*, *lifterheight* Possible World State: *objloc*, *at*, *distance* Belief State: *believeobjloc*,*believeplat*, *believedistance*,*needsearch*
Actions	Physical Actions: *askorder*,*moveto*,*fetch*, *adjustgotopose*,*adjustpose*,*setlifterheight*, *fetch*, *liftup*, *stepback*,*putinbasket*,*handover* HRI Actions: *askobjloc* Sensing Actions: *detplane*,*checkdistance*

Figure 2: List of non-rigid fluents and actions

The robot asks human the location of an object. To formalize *askobjloc*, a possible world fluent *objloc(O,L)* denotes the object O is possible to be at location L and a belief state fluent *believeobjloc(O,L)* that denotes the "robot believes that" object O is located at location L. When facing the person P, by executing action *askbojloc(O,P)* that asks person P the location of O, the robot can obtain *believeobjloc(O,L)* if *objloc(O,L)* is possible. It is formalized by

$$askobjloc(O, P) \textbf{ causes } believeobjloc(O, L) \textbf{ if } roboloc(P), objloc(O, L).$$

When the actual location is not given, there are many possibilities about the location of an object. Consequently, the

above causal law in practice simulates conditional effect of an action. Formally studying the modal properties of this particular representation pattern using $\mathcal{BC}$ is, however, beyond the scope of this paper and is a part of our future work.

During object manipulation, the robot approaches furniture and detects planes. Known plane heights of different furniture allows the robot to adjust its camera to proper angle for more accurate object recognition. This action is denoted by *detplane*(Pl, F). We use a pair of fluents: *at*(Pl, F), a possible world fluent that denotes plane *Pl* is possibly on furniture F, and *believeplat*(Pl, F), a belief state fluent that denotes the robot believes plane *Pl* is on furniture F:

detplane(Pl, F) **causes** *believeplat*(Pl, F) **if** *at*(Pl, F), *robotloc*(F), *believeobjloc*(O, F).

The second sensing action for manipulation is *checkdistance(O,F)*. It rotates the camera, identifies and localizes the object, then calculates the distance relative to the gripper. The distance can be grounded to *prefech* (the object can be directly grasped), *distadjust* (the robot needs to move closer) and *distgoto* (the robot needs to move to another side of the furniture to grasp). This action is formalized similarly by a pair of possible world fluent and belief fluent:

checkdistance(O, F) **causes** *believedistance*(O, D) **if** *roboloc*(F), *lifterheight*(Pl), *believeat*(Pl, F), *distance*(O, D).

Based on the distance, the robot can make adjustments (actions *adjustpose* to move closer, *adjustgotopose* to move to another side of the furniture), or fetch the object (*fetch*). Besides, we also have static laws that derive fluent values from other fluent values. They may be recursive in their nature:

$$\begin{array}{l} \textit{believeobjloc}(O, R) \textbf{ if } \textit{believeobjloc}(O, F), \textit{in}(F, R). \\ \textit{believeobjloc}(O, P) \textbf{ if } \textit{believeobjloc}(O, \textit{robot}), \textit{roboloc}(P). \end{array} \quad (1)$$

4.2 Plan Generation and Execution Controller

The main control loop for plan generation and execution is presented in Algorithm 1. The initial fluent set S for planning is generated as follows (line 1): (i) fluents that belong to definite world state are initialized based on robot's own sensor inputs; (ii) fluents that belong to belief state are initialized as negated literals, denoting that the robot does not know anything about them. Possible world state is not specified for initial state. According to the semantics of $\mathcal{BC}$, this will lead to all possible worlds to be generated for the initial state that complement the incomplete information. After that, the robot goes into a loop that keeps taking orders and serving people. The robot starts by identifying the person and asking "What can I do for you, Alice?" Human may reply: "Please bring me Pepsi." This sentence is processed by the semantic parser. The semantic parser is built upon a categorical combinatorial grammar parser [Xie *et al.*, 2013] that first translated natural language instructions into ASP rules. In this case, rules

$$\begin{array}{rl} \textit{goal(1)} & \leftarrow \textit{obtained}(O, \textit{alice}, \textit{maxstep}), \textit{pepsi}(O). \\ & \leftarrow \textit{not goal(1)}. \end{array} \quad (2)$$

will be added to goal representation. After a goal is generated, answer set solver (line 7) is called to generate answer sets using the union of domain representation D, goal G and initial fluent set S. In the returned answer set, a sequence of *actions* and *estates* that denote expected states before and after execution of actions are obtained. Following that, continuous observation loop starts (line 8, see Section 5 for details). From line 9, the current sequence of *actions* is sent to execution (line 14). The sequence of actions may successfully reach the goal (line 15), or need replan (line 19). For some unrecoverable failure or change, the goal may be aborted (line 22). The details of action execution and result update (line 14) is explained in Section 6.

Algorithm 1 Main Control Loop

```
1: Generate initial fluent set S
2: loop
3:    if goal G is empty then
4:       G ← ordertaken(P) from a person P
5:    end if
6:    while current task active do
7:       actions, estates ← get_plan(D, G, S)
8:       start background observation loop
9:       if actions ≠ cur_actions then
10:         cur_actions ← actions
11:         action_cursor ← 0
12:         replan_flag ← false
13:      end if
14:      result ← execute_actions(cur_actions, estates)
15:      if G ∈ result then
16:         mark task as finished
17:         break while
18:      else if result is REPLAN then
19:         continue
20:      else
21:         mark task as failed
22:         break while
23:      end if
24:   end while
25: end loop
```

5 Continuous Sensing

Continuous sensing (Algorithm 2) is a mechanism that whenever the sensing modules discover new information, they will ground results and update current state S. It allows the robot to reduce domain uncertainty at the same time of performing actions. As a result, the robot can be a lot more adaptive and robust to a changing domain and unreliable actions.

Continuous observation constantly monitors fluents *gripperempty*, *believeplatloc*, *believeobjloc* and *believedistance*. Vision module uses 3D Kinect cameras to constantly recognize planes on the furniture and add facts to S. For instance, if a new plane is detected from dining table, a new symbol *diningtableplane1* is generated, and *believeplat(diningtableplane1,diningtable1)* is added to S. At the same time, the numerical values denoting the height and location of the plane are also stored. The next time when a plane at similar location is detected, the robot will compare the location with the previous one, and recognize it as *diningtableplane1* to avoid duplicates. Similarly, the vision module recognizes different kinds of drinks and their locations, generates symbols and adds grounded facts using fluent *believeobjloc* to update S. Once the object is identified, its distance

may be calculated. Based on a predefined interval, the calculated distance is classified into *prefetch*, *distgoto*, and *distadjust*. The symbolic distance information is added to S using fluent *believedistance*. However, there are two extra cases to handle. The first case is when an object (e.g. *pepsi1*) is identified on the dining table but not within reach for the robot. According to our semantics of fluents, this result maps to fluents $\{\neg$*believedistance*$(pepsi1, D), \neg$*distance*$(pepsi1, D)\}$ where D be *prefetch*, *adjustdist* and *adjustgoto*. Note that $\{\neg$*believedistance*$(pepsi1, D)\}$ is not sufficient to represent this state, becaue it means the distance of *pepsi* is unknown (this is how we set the initial state, see Section 4.2). The second case is when vision module does not recognize any *pepsi1* when the robot is looking for it. In addition to setting the above fluents, *believeobjloc*$(pepsi1, diningtable1)$ is set to *false* as well. This negative literal indicates the robot abandons the information *believeobjloc(pepsi1,diningtable1)* that is obtained from human or continuous sensing. New plan may involve searching from nearby places or asking person again.

These facts enrich the knowledge about the domain and help the robot to find better plans. It should be noted that object recognition algorithm used by the vision module can more reliably identify objects if the robot is not moving. So if the robot believes an object is at certain location, while continuous observation does not recognize it, it will not update the robot's belief by removing *believeobjloc* fluent for this object, because it may be a false negative.

Fluent values may be derived from other fluents by static laws such as (1). When continuous observation sets *believeobjloc(pepsi1,diningtable1)* to be true, *believeobjloc(pepsi1,diningroom1)* needs to be added to S too. It can be derived by *cautious reasoning* with *believeobjloc(pepsi1,diningtable1)* and static laws of D using answer set solver (Line 5). *Cautious consequence* of an answer set program is the intersection of all answer sets. In some answer set solver such as CLASP [Gebser *et al.*, 2012], it is supported by using a command-line option `-e cautious`.

Every time when S is updated, robot verifies if the current plan is still the best one in presence of the new information (Line 6–9). A previous plan may easily become invalid, because, for instance, an object to retrieve disappears unexpectedly or the gripper fails to grasp the object. To validate the current plan, we perform a *temporal projection reasoning* [Lee *et al.*, 2013, Section 7] by projecting the end result of executing the remaining actions of the current plan in presence of the new information. If the goal is satisfied in the projected result and a shorter plan is not found, it means the current plan is still the best. Otherwise, a flag indicator for replan is set, and the current plan is aborted.

6 Execution Monitor and Action Controllers

Execution monitor dispatches action commands to corresponding controllers in robot interface for execution. Controllers operate hardware and, like in continuous observation, ground and return symbolic results (Algorithm 3). The controllers send low-level control commands to corresponding functional modules (Line 6). In case of *fetch*, motion planner generates motion trajectories. For HRI action *askobjloc*, speech module synthesizes voice and speaks to a person. For sensing action *checkdistance*, the controller is the same used in continuous observation. The current action may be aborted (Line 9) due to continuous sensing.

When action execution finishes, its full effect is derived through symbol grounding and cautious reasoning (Line 12). Symbol grounding is controller-specific. Manipulation tasks such as *fetch* is successful when the arm and gripper successfully move along all trajectories and pressure sensor indicates an object is grasped. For HRI action *askobjloc*, it may be grounded to atoms like *believeobjloc(pepsi1,diningroom1)*) for vague information, *believeobjloc(pepsi1,diningtable1)*) for precise information, or *needsearch(pepsi1)* when human answers "I don't know"; then the robot can search every furniture for Pepsi. Symbol grounding for sensing actions is handled the same way as in continuous observation. The execution result is compared with expected state to determine if replan is needed (Line 14) or goal is achieved (Line 16).

Algorithm 2 Observation Loop

```
 1: maintains a full set of observation fluents OBS
 2: loop
 3:     update OBS from observation
 4:     if OBS changed then
 5:         update S by cautious(OBS, D)
 6:         if ¬verify_plan(cur_actions, G, S) or exist a shorter plan
            then
 7:             replan_flag ← true
 8:         end if
 9:     end if
10: end loop
```

Algorithm 3 Execution Monitor

```
Require: cur_actions, estates
 1: for action_cursor = 0 to length(cur_actions) − 1 do
 2:     if replan_flag is true then
 3:         return REPLAN
 4:     end if
 5:     active_action ← cur_actions[action_cursor]
 6:     start executing active_action
 7:     while active_action is running do
 8:         if replan_flag is true then
 9:             abort active_action
10:         end if
11:         if active_action finished then
12:             res ← cautious(ground_result(active_action), G)
13:             update S by res
14:             if conflicts(res, estate) or REPLAN then
15:                 return REPLAN
16:             else if G ∈ res then
17:                 return G
18:             end if
19:         end if
20:     end while
21: end for
```

7 Experiment and Evaluation

The presented framework is implemented in *KeJia*. *KeJia* (Figure 3(a)) is equipped with a wheeled mobile base, a sin-

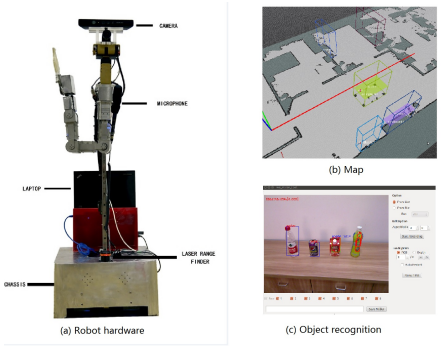

Figure 3: The robot hardware, SLAM generated map and vision

gle 5-Degree-of-Freedom arm, a microphone, 2D laser range finder, Microsoft Kinect and a high-resolution camera. Lower level modules include motion control which drives the mobile base and arm, 2D SLAM and navigation module (Figure 3(b)), 3D vision to recognize and localize furniture planes and pre-trained objects such as water, Pepsi, vitamin drink and apple juice (Figure 3(c)). Our system is implemented as a node in ROS (Robot Operating System) network. It subscribes all necessary information (hardware feedbacks, robot pose, recognized objects, etc.) and publishes control messages (navigate to some location, turn on object recognition function, etc.) which affect the behavior of the robot.

Such robot is constantly affected by uncertainties: perception errors (e.g. error in localization and recognized object positions, false positives and low recognition rates), HRI uncertainties (vague/erroneous information from human), changing environment (recognized object removed by human), and hardware malfunction (e.g. arm stuck during grasping). Since initially drinks and their locations are unknown to the robot and there are so many changing factors, it is extremely important for the robot to constantly gather domain information using HRI and sensing, adapt to change and recover from failure. In experiment we use CLASP as our answer set solver, and serving one person typically need the robot to execute 9–12 actions. Our demo video (https://youtu.be/zoKNFozlFPk) shows 7 consecutive tasks in one trial how robot responds to so many challenges.

Scenario 1. Alice requested water. The robot started by asking Alice the location. Alice offered vague information: in the dining room. The robot started searching in dining room by visiting the end table first. On the end table, there were Pepsi and water. However, due to unreliable vision, the robot recognized them to be water, and brought one to Alice.

Scenario 2. Bob requested Pepsi. Since the robot didn't recognize Pepsi in the previous task, it asked Bob the location and obtained a correct answer: "end table". At this very moment, Alice came to move the coke from end table to the dining table, making Bob's information erroneous. The robot, after failed to identify the Pepsi on the end table, abandoned the information from Bob, and found Pepsi on the side table. However, due to manipulation error, the Pepsi slipped out from the basket.

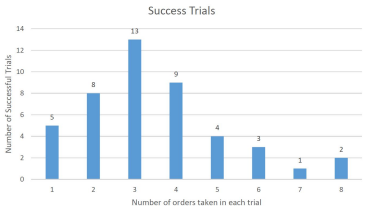

Figure 4: Results of Experiment

Scenario 3. Alice requested Vitamin. The robot got the answer "dining table" from Alice and moved to the dining table, but unfortunately, Vitamin was too far away for the robot to recognize. However, the robot recognized closer objects: water, Pepsi and apple juice. Consequently, the robot abandoned human information and searched by itself. Finally, Vitamin was found on the side board and was delivered to Alice.

Scenario 4. Bob requested water. The robot directly moved to the end table because in Scenario 1, it mistakenly identified Pepsi as water there. This time, vision correctly identified that water was not there, and the robot found an alternative by moving to the dining table and grasped water identified in Scenario 3.

Scenario 5. Alice requested apple juice. The robot directly moved to dining table and grasped the apple juice identified in Scenario 3. Vitamin was also recognized this time.

Scenario 6. Bob requested Vitamin. The robot moved to dining table for Vitamin directly. After measuring distance, it adjusted its position by moving to another position to grasp. The first attempt failed and was captured by continuous sensing immediately. A new plan was generated and executed, which was successful.

Scenario 7. Alice requested Pepsi. The robot directly moved to Pepsi and grasped it. Unfortunately, the gripper hit the can and stuck in its trajectory, leading to manipulation failure.

Despite so many uncertainties and failures, the robot managed to handle most of them pretty robustly and achieved 5 of 7 tasks, by adaptive planning and reasoning. Vision errors tend to occur more frequently in the early stage, but they are all corrected later. Furthermore, after the 3rd run, the robot no longer asked human questions because it has already acquired sufficient information about drinks and their locations.

Following the scoring policy of RoboCup@Home competition, we evaluate our framework by conducting a total of over 20 hours running the robot and finished about 157 orders in 45 trials. The trial stops due to unrecoverable fault. As is shown in Figure 4, most often the robot can successfully serve 2–4 persons consecutively, with the average being 3 persons. Despite that, there are two trials that the robot managed to take 8 orders. Given the challenge of GPSR domain, the experiment demonstrates robustness and the ability to tackle uncertainty, failure and incomplete information.

8 Conclusion

In this paper we proposed a framework of building general-purpose service robot, by handling HRI, sensing and physical actions in a uniform representation and execution loop

with continuous observation. Our experiment demonstrates robustness of the service robot in domestic environment. We conclude that, proper use of symbolic planning combined with task-oriented knowledge acquisition can be helpful to handle unpredictable domain changes and perception errors, two challenges in all GPSR domains. In the future, we will address more complex domains with more reliable control.

References

[Bohren and Cousins, 2010] Jonathan Bohren and Steve Cousins. The smach high-level executive [ros news]. *IEEE Robotics & Automation Magazine*, 4(17):18–20, 2010.

[Brenner and Nebel, 2009] Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331, 2009.

[Brenner *et al.*, 2007] Michael Brenner, Nick Hawes, John D Kelleher, and Jeremy L Wyatt. Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. In *IJCAI*, pages 2072–2077, 2007.

[Brenner, 2007] Michael Brenner. Situation-aware interpretation, planning and execution of user commands by autonomous robots. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 540–545. IEEE, 2007.

[Chen *et al.*, 2010] Xiaoping Chen, Jianmin Ji, Jiehui Jiang, Guoqiang Jin, Feng Wang, and Jiongkun Xie. Developing High-Level Cognitive Functions For Service Robots. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010.

[Chen *et al.*, 2014] Kai Chen, Dongcai Lu, Yingfeng Chen, Keke Tang, Ningyang Wang, and Xiaoping Chen. The intelligent techniques in robot kejia–the champion of robocup@ home 2014. In *RoboCup 2014: Robot World Cup XVIII*, pages 130–141. Springer, 2014.

[Dearden and Burbridge, 2013] Richard Dearden and Chris Burbridge. An approach for efficient planning of robotic manipulation tasks. In *ICAPS*. Citeseer, 2013.

[Erdem *et al.*, 2012] Esra Erdem, Erdi Aker, and Volkan Patoglu. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *Intelligent Service Robotics*, 5(4):275–291, 2012.

[Gebser *et al.*, 2012] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, 2012.

[Hanheide *et al.*, 2015] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 2015.

[Havur *et al.*, 2013] Giray Havur, Kadir Haspalamutgil, Can Palaz, Esra Erdem, and Volkan Patoglu. A case study on the tower of hanoi challenge: Representation, reasoning and execution. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4552–4559. IEEE, 2013.

[Hoffmann and Brafman, 2005] Jörg Hoffmann and Ronen Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, volume 2005, 2005.

[Khandelwal *et al.*, 2014] Piyush Khandelwal, Fangkai Yang, Matteo Leonetti, Vladimir Lifschitz, and Peter Stone. Planning in Action Language $\mathcal{BC}$ while Learning Action Costs for Mobile Robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[Lee *et al.*, 2013] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action Language $\mathcal{BC}$: A Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Lifschitz, 2008] Vladimir Lifschitz. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. MIT Press, 2008.

[Petrick and Bacchus, 2002] Ronald PA Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *AIPS*, pages 212–222, 2002.

[Petrick and Foster, 2013] Ronald PA Petrick and Mary Ellen Foster. Planning for social interaction in a robot bartender domain. In *ICAPS*, 2013.

[Puigbo *et al.*, 2015] Jordi-Ysard Puigbo, Albert Pumarola, Cecilio Angulo, and Ricardo Tellez. Using a cognitive architecture for general purpose service robot control. *Connection Science*, 27(2), 2015.

[Son and Baral, 2001] Tran Cao Son and Chitta Baral. Formalizing sensing actionsa transition function based approach. *Artificial Intelligence*, 125(1):19–91, 2001.

[Son *et al.*, 2004] Tran Cao Son, Phan Huy Tu, and Chitta Baral. Planning with sensing actions and incomplete information using logic programming. In *LPNMR*, pages 261–274. Springer, 2004.

[Srinivasa *et al.*, 2012] Siddhartha S Srinivasa, Dmitry Berenson, Maya Cakmak, Alvaro Collet, Mehmet R Dogar, Anca D Dragan, Ross Knepper, Tim Niemueller, Kyle Strabala, Mike Vande Weghe, et al. Herb 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of the IEEE*, 100(8):2410–2428, 2012.

[Srivastava *et al.*, 2014] Sanjeev Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stephen Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 639–646. IEEE, 2014.

[Srivastava *et al.*, 2015] Siddharth Srivastava, Shlomo Zilberstein, Abhishek Gupta, Pieter Abbeel, and Stuart Russell. Tractability of planning with loops. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Thomason *et al.*, 2015] Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the Twenty-Fourth international joint conference on Artificial Intelligence (IJCAI)*, 2015.

[Wachsmuth *et al.*, 2015] Sven Wachsmuth, Dirk Holz, Maja Rudinac, and Javier Ruiz-del Solar. Robocup@home – benchmarking domestic service robots. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Xie *et al.*, 2013] Jiongkun Xie, Xiaoping Chen, and Zhiqiang Sui. Translating action knowledge into high-level semantic representations for cognitive robots. *Nonmonotonic Reasoning, Action and Change*, page 53, 2013.