Tell me when and why to do it!

Run-time planner model updates via natural language instruction

Rehj Cantrell
Cognitive Science Program
Indiana University
Bloomington, IN
rcantrel@indiana.edu

Kartik Talamadupula
Dept. of Computer Science
Arizona State University
Tempe, AZ
krt@asu.edu

Paul Schermerhorn
Cognitive Science Program
Indiana University
Bloomington, IN
pscherme@indiana.edu

J. Benton
Dept. of Computer Science
Arizona State University
Tempe, AZ
j.benton@asu.edu

Subbarao Kambhampati
Dept. of Computer Science
Arizona State University
Tempe, AZ
rao@asu.edu

Matthias Scheutz
HRI Laboratory
Tufts University
Boston, MA
mscheutz@cs.tufts.edu

ABSTRACT

Robots are currently being used in and developed for critical HRI applications such as search and rescue. In these scenarios, humans operating under changeable and high-stress conditions must communicate effectively with autonomous agents, necessitating that such agents be able to respond quickly and effectively to rapidly-changing conditions and expectations. We demonstrate a robot planner that is able to utilize new information, specifically information originating in spoken input produced by human operators.

We show that the robot is able to learn the pre- and postconditions of previously-unknown action sequences from *natural* language constructions, and immediately update (1) its knowledge of the current state of the environment, and (2) its underlying world model, in order to produce new and updated plans that are consistent with this new information. While we demonstrate in detail the robot's successful operation with a specific example, we also discuss the dialogue module's inherent scalability, and investigate how well the robot is able to respond to natural language commands from untrained users.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Concept Learning*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language Parsing and Understanding*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, and generation*

General Terms

HRI Communication

1. INTRODUCTION

Robots are currently being used in and developed for critical HRI applications (e.g., search and rescue) in which they must smoothly adapt to new information, and communicate quickly and effectively with human operators despite serious challenges imposed by the task and environment. These scenarios tend to induce high stress levels in human operators, while rapidly-changing conditions and expectations combine to increase operators' cognitive loads. If robots are be truly helpful in such scenarios, they must be able to communicate naturally and react flexibly to dynamic conditions by, for example, updating previous goals and accepting new goals according to changing information about the world. In this paper, we describe a robot architecture capable of understanding new action sequences presented via natural language and integrating the new knowledge to allow the robot to reason about the new abilities (e.g., in problem-solving).

The architecture described here features a planner that utilizes new information, such as state updates from sensory processing, at run time (e.g., to generate a plan to enter and explore a room when informed of a newly-detected open doorway). We outline extensions to the planner that allow us to update its list of available actions (e.g., to include a new capability to enter rooms by pushing closed doors). We then demonstrate that the robot is able to learn the preconditions for and the postconditions of previously-unknown action sequences from a natural language description (e.g., "if you are at a closed door and you push it one meter, you will be in the room"), and immediately apply this knowledge to generate and update plans.

In previous work [3, 2] we described a natural language human-robot interaction system that uses a trainable parser to operate on spoken natural language input, producing incrementally and in parallel (1) a semantic representation, and (2) identification of and information about new word forms. In this paper, we expand on this in two key ways. First, the robot's learning capabilities are expanded to include learning pre- and postconditions of an action. For example, given a robot's limited conceptual knowledge, it may not be immediately clear that, when one is in the hallway and goes through a door, one is then in a room. Our method allows the robot to accept information given ver-

bally, which a planner can then combine with the robot's existing knowledge in order to produce logical insights. Second, our method of referent grounding is expanded to allow the robot to infer the existence of entities that have not been explicitly mentioned. For example, when the robot knows that doors are connected to rooms, mention of a door implies that there is also a room.

The structure of the paper is as follows. First we summarize relevant previous work and provide an overview of the system in Section 2. Next we describe the relevant modules in detail: in Section 3, we detail the dialogue module using an example, and in Section 4, we discuss the robot's planner. This is followed by, in Section 5, a brief evaluation of the informal application of the system to human language elicited from naive subjects. Finally, we discuss our conclusions and future work in Section 6.

2. BACKGROUND

Planners have improved significantly in terms of scalability and speed (c.f. [12]), but this has come at the cost of standardizing assumptions about the nature of the input. In particular, modern planners typically assume full up-front knowledge of the world as well as of the underlying model that the robot is subject to—the *closed world and model* assumption (see section 4.2 for a detailed discussion of this). However, this assumption is unsuited for dynamic scenarios like search and rescue, where the world is constantly changing with new objects and conditions continually being discovered. Moreover, even the model is subject to change (e.g., when new capabilities are learned at run time). A number of systems have considered execution monitoring and plan repair or replanning upon the discovery of an inconsistent execution state, notably [11] and [7] among others. There has also been some work on systems that control service robots, the most relevant being [1]. Yet, no robotic architecture has the ability to integrate new discoveries at all levels—including run-time model updates in the planner—in order to take best advantage of the new information.

Similarly, many natural language processing systems require that all necessary knowledge be provided at startup. However, a number of projects have begun to investigate robotic architectures that can learn actions and action sequences. For example, [14, 18, 13] learn action sequences and a single associated precondition: the environment in which the sequence is learned is assumed to be the only environment in which the sequence is valid. While this assumption may be sufficient for certain tasks and in certain domains (e.g., kitchen tasks tend to be performed only in the kitchen) it limits the usefulness of the learned sequence (e.g., if the robot must be explicitly taught a sweeping task in each room of the house).

Similarly, [9, 8] does not allow postconditions to be specified, though preconditions are verbally stated. The input language is limited to a structured fragment of English. An example of the required input is, "Environment with initial conditions $\emptyset$. You start in r_1 [a location] with initial conditions $\emptyset$. Go to $\{r_1, ..., r_2\}$ infinitely often. Call Medic iff Person is found. Call Fireman iff Gas is detected."

[10] allow operators to specify an action, such as `grasp the doll`, and its stated effect, `be grasping the doll`. This corresponds to the following instruction: until agent x senses that it is grasping the doll, agent x should continue to perform a grasping action on the doll. However, the instruction has to be presented with a specific input structure: "Until you are grasping D; Holds you should grasp object D," which is then translated into a formal procedural description that allows the specification of preconditions but not postconditions. The formal language into which the commands are translated does list effects, but these effects are stopping conditions, rather than, as in our method, propositions that become true upon performing the action sequence.

Each of these projects addresses important aspects of action learning. However, a more comprehensive capability to, at run time, integrate knowledge learned via natural language at all levels throughout the architecture is necessary before robots will be able to, for example, use natural language input to update the planner's knowledge and even underlying assumptions. Robotic architectures must be developed that allow for flexible actions and plans, based on new knowledge derived from flexible natural language input. This necessitates the ability to expand concept knowledge both within the planner (e.g., new action concepts that are made available) and within the robot's dialogue module. In this paper, we describe a method of explicitly defining new actions, including pre- and postconditions, using natural language in a way that allows the robot's reasoner to be updated, making new capabilities available at run time.

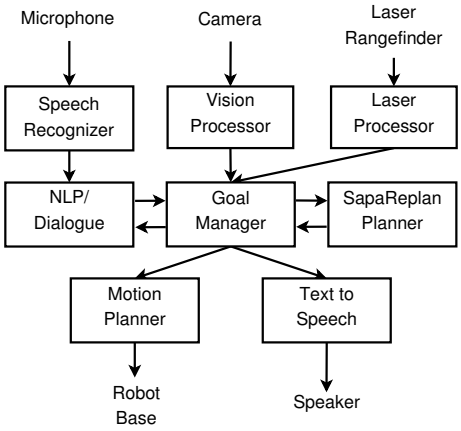

Figure 1: Architecture overview.

2.1 System Overview

Figure 1 provides an overview of the DIARC architecture used below. In particular, it shows the tight integration of the dialogue module and the planner with the goal manager. The goal manager maintains a database of procedural knowledge (i.e., actions that the architecture is capable of performing, along with their pre- and postconditions and information about the types of entities that could be encountered; [15]). The goal manager attempts to ensure that progress is made toward the achievement of all of the agent's goals by collecting information from the sensor servers and dispatching commands to the effector servers as required. Actions for multiple goals can execute in parallel, however, when resource conflicts arise, they are mediated in favor of the goal with the highest priority.

When the dialogue module determines that speech input consists of a command, a new goal is generated to carry out the command and submitted to the goal manager, which

provides updates on the state of that goal as needed. Similarly, when a new action definition is distilled from natural language input, the dialogue module submits the new action to the goal manager, making it available to the goal manager from that point on. Natural language processing is described in greater detail in Section 3.

The planner's domain is initialized at system start-up by the goal manager, based on the contents of its knowledge base, ensuring that the planner has the most up-to-date information about the robot's capabilities. Moreover, because the planner can accept run-time updates, the domain can be augmented by newly-learned actions that can be used to improve subsequently-generated plans. Similarly, as the robot interacts with the world, the planner receives notification of state updates (e.g., external world states obtained from sensors, internal states such as sub-goal completion) from the goal manager; if an update triggers replanning, the new plan is returned to the goal manager in the form of an action script that can be directly executed to achieve the associated goals. The planning component is described in greater detail in Section 4.

This architecture is designed to enable the augmentation of a robot's capabilities at run time in response to operator input. For example, consider a search and rescue scenario in which a robot is searching a building that is unsafe for human exploration. Instead, the robot communicates with a human operator via a wireless link-up. At the beginning of the exploration task, the robot understands that it should enter any rooms it encounters as it traverses a hallway, and its procedural knowledge base includes an action sequence that will enter a room through an open door. Furthermore, its opportunistic planner, upon receiving a state update indicating a newly-discovered open doorway, produces a new plan to enter and search the room before continuing to explore the hallway. However, state updates regarding closed doors do not result in plans to enter and search rooms, as there is no known action sequence that will lead to the robot being in the room given the "closed door" precondition.

What if, during the search operation, the human operator is informed that the building's doors are all designed to unlatch when the fire alarm is triggered? In that case, the robot should be able to push the doors open, allowing it to search rooms behind closed doors. Without the ability to learn new actions via natural language and propagate that knowledge through the architecture, the operator is faced with a choice. The robot could be allowed to continue searching only rooms with open doors—but those could be the very rooms that are most likely to have been evacuated already. Or the architecture could be shut down while the operator adds that information and any associated actions to the robot's procedural knowledge base—but that would cost valuable time, and also assumes that the operator has the skills necessary to reconfigure the robot in this way.

Our solution, in contrast, allows the operator to transmit the information to the running system via natural language, associating appropriate pre- and postconditions with a previously-known action (initially applicable only in other contexts): "If you are at a closed door and you push it one meter, you will be in the room." The new information can be given by an operator with limited knowledge of the details of configuring the architecture and integrated with little or no delay, allowing the robot to search the rooms where it is most likely to find victims.

This motivating example will be used throughout the remainder of the paper, as we describe the unique features of the architecture that make it possible: sophisticated natural language processing capabilities, and an opportunistic planner that can accept world model updates, in addition to state updates, at run time.

3. NATURAL LANGUAGE PROCESSING

In this section, we describe the robot's dialogue module. For descriptive ease, it is described in a modular, sequential manner; in fact, however, all facets of processing occur incrementally and in parallel. At any time the module can be queried for a completely-processed semantic representation of the utterance as it currently stands.

In particular, this means that NLP and dialogue management are done in a tightly-integrated fashion. As utterances are recognized, mentions of entities are tracked, allowing incremental referent grounding. Other discourse information that is tracked includes, for example, points of confusion for the system, such as unknown words and utterances that do not fully describe the desired action.

NLP utilizes a number of initial knowledge sources, in particular, a dictionary containing primitives that can be combined into complex forms; an annotated corpus to train the syntactic parser; and optionally a database containing contextual knowledge (e.g., about interactions between entities). Entries in this database are represented in the same way as are semantic representations produced by the dialogue module (and perceptual modules such as vision) so that the database could in principle be built entirely from things the robot is told or is able to learn.

To demonstrate key steps of the process, we describe here how the dialogue manger handles the example sentence, "if you are at a closed door and you push it one meter, you will be in the room". The goal of semantic processing for this sentence is to identify the action definition, preconditions, and postconditions. Specifically, we must correctly segment the natural language input as follows:

preconditions you are at a closed door
action definition you push it one meter
postconditions you will be in the room

We will first describe reference and anaphora resolution, and then parsing, which produces a literal semantic representation. The final subsection will describe semantic manipulation in accordance with real-world principles known to the robot. After that, we will describe how the new action is given to the goal manager.

3.1 Reference Resolution

Each time a discourse entity (currently identified by the existence of a noun or a pronoun) is encountered during sentence processing, an object is created to hold all information about that entity. Each object may represent an independent entity, or may be reference to another entity, indicating that the two entities corefer. For example, when an indefinite phrase "a closed door" is encountered, an object is created that holds the information that it represents an entity of type **door** which exhibits the property **closed**. According to the determiner "a" this object represents an independent entity, which is first asserted to exist in this statement (i.e., it must not refer to a previous discourse entity).

By contrast, when a definite phrase such as "the room" is encountered, an object is created that holds the information that it represents an entity of type **room**; however, based on the determiner "the", the entity must be a reference. This alerts the robot that it must locate some referent (e.g., among previous discourse entities or by using its perceptual modules). Anaphors such as pronouns are treated similarly.

3.2 Parsing

As implied above, the goal manager requires a partial parse, ideally with breaks between distinct phrases, in order to integrate the new information into its knowledge base. Such a parse is assumed, and is realistic only in the current domain-limited context. In section 5.1, we briefly touch on this issue when trying alternate phrasings of our instruction, but future work must explore ways of compensating for a less-than-idea partial parse.

Given such a parse, the preconditions, action definition, and postconditions may be derived from a variety of verbal input frameworks such as: "if [preconditions are true] if [action definition is performed] then [postconditions will be true]", "if [preconditions are true] and [action definition is performed] then [postconditions will be true]", or "if you [action definition] when [preconditions are true] then [postconditions will be true]".

For this reason, a dependency-based parser that uses lambda calculus to produce semantic representations [3] is trained on a set of phrases labeled as a precondition, an action, or a postcondition. The phrases were extracted from the CReST corpus [6] and manually modified. Each phrase is returned as a pair comprising a semantic representation and one of these labels. For example, the input "if you are at a closed door and you push it one meter you will be in the room" results in the list of labeled phrases:

entities $door(x) \wedge agent(q) \wedge Rs.object(s) \wedge Rr.room(r)$[1]
pre $closed(x) \wedge outside(q, x)$
action $pushed(q, s, 1)$
post $in(q, r)$

In other words, the robot has segmented the natural language input into several distinct pieces of information. (1) the entities under discussion include a door, an agent, an ungrounded object s (which must be grounded in terms of previous discourse entities), and an ungrounded room represented (which must be grounded in terms of previous discourse entities or the environment). (2) This action is relevant only when the agent is outside the closed door (3) from which position, if the agent pushes s 1 meter, (4) the agent will then be in the room. The literal meaning of the input has now been established, but some important information is still missing. In particular, what object must be pushed, and what room is the robot entering? The semantic manipulation step provides the missing pieces.

[1]The operator R indicates that its argument is a reference which must be grounded to a discourse entity, a real-world entity, or a **theoretical entity**, an entity that the robot infers must exist (e.g., if a door is mentioned a room must exist because doors occur only in relation to rooms). An entity of this type will be introduced in the next subsection.

3.3 Semantic Manipulation

Assumptions and gaps in the input sentence are addressed by consulting the background knowledge accessible to the dialogue module, which includes the following: $\forall x[door(x) \rightarrow \exists y, z.room(y) \wedge location(z) \wedge doorconnected(x, z, y)]$ or "for each door, there exists a room y and a space z that is connected to that room by the door". To be "outside a door" is to be in z, giving us the following: $\forall x[door(x) \rightarrow \exists y, z.room(y) \wedge location(z) \wedge doorconnected(x, z, y) \wedge (outside(q, x) \leftrightarrow at(q, z))]$. Logically combining this with the explicit semantics, we arrive at:

entities $door(x) \wedge agent(q) \wedge Rs.object(s) \wedge Rr.room(r) \wedge room(y) \wedge location(z)$
pre $closed(x) \wedge outside(q, x) \wedge doorconnected(x, z, y) \wedge at(q, z)$
action $pushed(q, s, 1)$
post $in(q, r)$

Now the robot knows about the particular room y that is connected to a location z outside the door, which allows an agent at z to enter y. However, entities s and r remain unresolved. In this case, the sentence is a theoretical description of an action and its consequences, so all entities need only be grounded within the discourse, *not* in the real world. One consequence of this is that entities that contain assertions of existence such as "a door" need not be grounded. However, if the sentence were an instruction/statement pair such as "push a door 1 meter and then you will be in a room," the robot would need to ground "a door" to a single, perceivable real-world entity in order to follow the instruction.

As described above, resolution is done by comparing what is known about such a reference with what is known about each independent entity and identifying those that are semantically compatible. If this leaves several candidates, a selection is made based on heuristics such as recency of mention and parallel grammatical function. In this utterance, there are two entities flagged for resolution. The first is the pronoun "it". In this case, only one object has yet been mentioned, the door. Thus "it" can be resolved right away, resulting in the following semantic representation:

entities $door(x) \wedge agent(q) \wedge Rr.room(r) \wedge room(y) \wedge location(z)$
pre $closed(x) \wedge outside(q, x) \wedge doorconnected(x, z, y) \wedge at(q, z)$
action $pushed(q, x, 1)$
post $in(q, r)$

The second reference, the noun phrase "the room", refers to an entity not found in the *explicit* content of the sentence. However, as we saw above, the entity was implied. This implication was made explicit through the inclusion of the robot's background knowledge of the world. Thus we resolve r and arrive at:

entities $door(x) \wedge agent(q) \wedge room(y) \wedge location(z)$
pre $closed(x) \wedge outside(q, x) \wedge doorconnected(x, z, y) \wedge at(q, z)$
action $pushed(q, x, 1)$
post $in(q, y)$

These parts can now be assembled and sent to the goal manager as described next.

3.4 Submitting the New Action

The new capability is introduced into the goal manager via a method call: `associateMeaning(action definition, preconditions, postconditions)`. This tells the goal manager that it needs to `associate` a particular action sequence (the `action definition`) with the given `preconditions` and `postconditions`. Note that in this case, as in many others, there are other postconditions, or side effects (e.g., the door will be open), but for the current example, if we have entered the room, the fact that, out of all potential ways the room could be entered, it happened to be through the mechanism of opening a closed door, is not explicitly treated.

Once the goal manager has processed the new capability, it can be used in response to subsequent requests by the operator, and also presented to the planner. We describe how the planner utilizes this information next.

4. PLANNING CAPABILITIES

An indispensable requirement in supporting a flexible dialogue module on a robot is a robust planner that can handle changes to the world model and the problem at hand, and re-plan taking these modifications into account. If the human specifies (during execution) that the agent must push the door to a room in order to enter that room, the robot must be in a position to understand the implications of that directive. If there are goals that can only be achieved by entering that room, the robot must update its understanding of the world and infer that the new capability now allows it to achieve those goals. It is these tasks that are performed by the planner: (1) the task of updating the robot's model of the world and understanding the implications of those changes, and (2) processing changes to the facts and goals in the robot's knowledge.

We employ the `Sapa Replan` [4] planner, an extension of the metric temporal planner `Sapa` [5]. `Sapa Replan` is a state-of-the-art planner that can handle actions with costs and durations, partial satisfaction of goals and changes to the world and model via replanning. Of these, the most relevant to the problem of dynamic natural language input is the ability to model and use changes to the world to the robot's advantage. The planner is mainly concerned with two such changes: changes to the world facts and the problem at hand currently, and modifications to the overall model of the world that the robot is using.

4.1 Changes to the Model

The planning model is an abstraction of the dynamics of the world and is used by the planner to guide the robot in its pursuit of the various goals assigned to it. The constituent components of such a model are usually the types of objects in the agent's world, the predicates that describe the relationships between these objects, and the actions available to the agent. An initial model is made available to the planner (and hence the robot) along with a description of the world (described in section 4.2). Changes to this model are generally harder to process than changes that are restricted to specific facts, since they involve the modification of several facts that are distributed throughout the planner's knowledge base. Additionally, verifying the consistency and correctness of an executing plan—a non-trivial task at best when facts about the world are changing—is rendered even more difficult by the extensive role that the model plays in the process of plan synthesis.

Unlike changes to facts in the world that can either be discovered by the robot's sensors or specified by a human in the loop, changes to the model are more likely to be entirely specified by human operators using natural language. One reason for this is the fact that such updates are much more complex and often invoke the robot's capabilities and the dynamics of the world's evolution. Examples of utterances that force updates to the model include, for example, statements that give new information about how entities in the world are interrelated and how the robot should interact with real-world entities, such as the directives produced by naive human users in Section 5. These are additional capabilities that are being specified to the robot during execution, and they may result in a change in achievability as far as the current goals to be fulfilled are concerned.

Currently, such directives are supported in the planner by modifying the planner's knowledge base. A reference to the concerned action[2] is given and the new capability is added to the effects of that action. The planner process is then restarted and the search for a new plan begins, since the added capability may have altered the reachability of some goals. We demonstrate in the following section that, in the example scenario, this method is sufficient to achieve our current goals. When considering the problem of plan validity under modifications to the planning model in general, though, the system may either replan from scratch, or update and re-use the current plan.

Replan from Scratch: Given a new version of the domain model (with updates), the planner runs again in order to come up with a plan that completely replaces the currently executing plan. This is the approach that we employ currently.

Plan Re-use: The planner analyzes the current plan with respect to the updates received and adds, removes, or updates an action.

The addition of an action to the domain does not affect the validity of the current plan. Other metrics associated with the problem may change, since a new plan may now be available, but no change is necessitated if a sufficient plan is already executing. Likewise, the removal of a non-participating action (i.e., an action that does not participate in the currently executing plan), no change is necessary.

However, removal an action that does participate in the currently executing plan is a more complex case, and requires a detailed analysis of the sub-goals (and top-level goals) that the action supports, and the nature in which its removal will affect those goals. Similarly, when parts of an action are updated (addition or deletion), a more complex analysis of the various commitments entailed by the currently executing plan is required.

While a full analysis of these methods followed by a selective implementation of the best approach is work for the future, the effectiveness of the current approach is demonstrated in the next section.

4.2 Changes to the World

Dynamic environments like search and rescue also have a stream of constantly updating information about world facts

[2]For this work, we restrict our attention to directives that expand on the robot's capabilities, and that are associated with a particular action.

associated with them. This information is quite distinct from changes to the overall model, since it is specific to that particular instance and the state of the world at the current instant. Any agent that seeks to effectively achieve a given set of goals in such conditions must be in a position to process and take advantage of this information. Agents that are unable to do so run the risk of executing plans that are no longer valid for the given circumstances. Knowledge about these changes can come from two main sources, as mentioned previously: the first is through the robot's own exploration of the world, and the feedback received via its sensors; the other is through natural language (i.e., human-issued knowledge and directives). The planner must take these changes into account and ensure that the plan that is currently being executed remains valid and effective. If this ceases to be the case, the robot must replan, taking into account the new knowledge and any changes to objectives, and pass a new plan for execution on to the agent.

The `Sapa Replan` planner that directs the robotic agent is comprised of two main components – the execution monitor and the planner – working in a tight loop. The planner receives the initial state information and goals and dispatches a plan to the monitor that is sent to the robot for execution. The monitor then listens in for updates from the world, which can be generated from either of the two sources mentioned previously. In particular, updates can include new objects, timed events, goal additions or modifications, and a new time point from which execution is expected to resume. The update syntax is as follows:

```
(:update
 :objects
    red3 - room
 :events
    (at 125.0 (not (at red2)))
    (at red3)
    (visited red3)
 :goal (visited red4) [500] - hard
 :now 207.0)
```

These problem updates cause the monitor to interrupt the planner and restart it after updating the internal problem representation. However, handling the arrival of new goals that are contingent upon objects and facts initially unknown to the agent is a challenge for most modern automated planners. This is due to the fact that planners assume a closed world and model at the outset—that is, they expect full knowledge of the initial state and the model of the world, and up-front specification of all goals. Information that is left unspecified initially is assumed to be false in order to gain closure over the knowledge about the world. Instead of this, what is required is a way to exclude certain classes of objects from this closure requirement, and a way to specify additional information about such objects (such as goals associated with them). To this end, we extended `Sapa Replan` with the concept of *Open World Quantified Goals* (OWQG) [17] which are constructs that provide a compact way of specifying conditional reward opportunities over an open set of objects.

For example, a human commander may assign the robot a soft goal of reporting the location of all injured humans, with the additional information that these injured humans may be found inside rooms. Such a directive would be translated to an OWQG using the following syntax:

Figure 2: A Videre ERA equipped with a USB web-cam, a Hokuyo Urg laser range-finder, a microphone, and a quad-core Linux PC for onboard processing exercising its newly-acquired capability of opening closed doors to look for red boxes.

```
(:open
  (forall ?r - room
    (sense ?hu - human
     (looked_for ?hu ?r)
      (and (has_property ?hu injured)
           (in ?hu ?r))
  (:goal (reported ?hu injured ?r)
           [100] - soft))))
```

While space does not allow a full exposition of the syntax and the manner in which OWQGs are handled by the planner, additional information may be found in [16].

5. DISCUSSION

The natural language processing and planning capabilities described in Sections 3 and 4 were implemented and tested on the robot shown in Figure 2 in a scenario similar to the one described in Section 2 (see the video at The video at `http://www.youtube.com/watch?v=NEhBZ205kzc`). The search target in the demonstration is a red box and a microphone is used instead of radio communication, but otherwise the demonstration is faithful to the original motivating example. Sphinx 4 was used for speech recognition.[3].

Although this example demonstrates that the architecture is able to learn the action and use it for problem-solving, it is clear that to be of general use, the robot must be able to learn more than one action. Furthermore, examples produced by researchers intimately familiar with the details of their own systems, while instructive for demonstrating the robot's operation, are necessarily informed by detailed knowledge of the system, so it is unclear how widely successful even this single example would be given an uninitiated user. Therefore, in this section we first discuss the system's inherent general scalability (i.e., to different action descriptions), then investigate how successfully the robot is able to

[3]`http://www.cmusphinx.org`

handle the types of natural language instructions produced by naive users for this particular example using the results of a preliminary pilot eliciting natural language input from a small sample of untrained users.

5.1 System Scalability

So far, only one example has been presented. However, several facets combine to contribute to the system's scalability to other scenarios. In particular, the dialogue module is characterized by the ability to scale up to larger domains due to three key features. First, the parser is trainable from annotated data. Second, given an initial set of primitives, new terms describing complex combinations of these primitives can be learned and added to the dictionary. Third, parses often fail around certain predictable points such as conjunctions. These points can indicate, among other things, phrase breaks, allowing a parser's weak point to be used to produce a more flexible set of of processable input types. These characteristics combine to contribute to the dialogue module's extensibility, producing the ability to successfully analyze the semantics of new natural language input.

Moreover, the interface via which a new capability is submitted to the goal manager is flexible enough to accept a wide variety of definitions, requiring only the basic elements described in Section 3: the action description, a (possibly empty) list of preconditions, a list of postconditions (or effects) of the given action sequence, and (optionally) a name. Those elements are sufficient to make the new capability available for use by the goal manager, and to allow it to forward the information to the planner.

Finally, `Sapa Replan`'s flexible mechanisms for defining new domain elements make it possible to update the world model at run time with any construct that would be permissible at system start-up. This, combined with a design that allows it to respond to changes to the world, ensure that any newly-learned capability can be injected into the running planner and used to improve subsequent plans.

To test this scalability, we tried a variety of different phrasings on the same instruction. Simple changes such as alternating conjunctions (e.g., "if" replacing "and", "then" replacing null) and reordering information (e.g., "if you push a closed door one meter when you are at it then you will be in the room") worked easily. A resolution error arose with "to go into a room when you are at a closed door push it one meter" — "it" was incorrectly found to be coreferent with "room". Similar errors will be corrected with more fine-grained semantic knowledge of object types.

Next, an assortment of disfluencies were tried. "Uh" insertion in a variety of locations (e.g., between prepositional phrases and their head verb, between object phrases and their head verb) posed little problem. One case resulted in a failure to attach "closed" to "door" (i.e., the system would not understand this instruction is needed only when the door is not already open). Repetition of non-content words (e.g., "to uh to go") was also unproblematic, with the repetitions left unattached. Repetition of content phrases was slightly more problematic. "Push it uh push it one meter" resulted in two sequential steps, "push it" and "push it one meter" being added to the instructions.

However, even though these examples were constructed after the system was designed (i.e., they had no effect on the design of the system), they likely are not representative of the kind of language naive users would produce. In the next section, we elicit language from naive users.

Table 1: Naive users instructions to Gogol.

continue forward	✓
detect a closed door push on the door and enter	✓
open the door	✓
go straight into the door and keep pushing	
go up to a wall and push on it	✓
um you need to push on the door to make it open	
push the door open	✓
if you detect a closed door push on it	✓
when you think you've hit a wall push on it	✓
push on every wall you find	

5.2 Naive Subject Evaluation

Because several aspects of any NLP system are heavily reliant on phrasing, it is important to understand what type of natural language *naive* users produce if we want to evaluate the general applicability of the system.

As a pilot study, a small sample of naive users (10) were shown diagrams depicting Gogol, a non-humanoid robot, as it navigates a hallway searching for blocks. The users were told that Gogol is able to sense both open and closed doors. To maintain consistency with the demonstration run, "closed" doors are actually depicted slightly open and subjects were told that Gogol considers them "closed". They were then told, "Unbeknownst to him, if he detects a closed door, he can push on it. Besides moving him into the room, this action moves the door into a position that Gogol considers 'open'." Subjects were then instructed to "tell Gogol how to go into rooms with closed doors". The instructions did not explicitly state that Gogol did not know the meaning of the verb "open", only that he did not know **how** to perform the action. "Open" was not used as a verb anywhere in the description of the scenario or the instruction.

Users responses are shown in Table 1. The dialogue module correctly parsed seven of the ten responses. Most of the resulting instructions were syntactically simple, with only two users making use of more than simple imperatives. While all instructions included an action or series of actions only half included something identifiable as a pre- or post-condition. All instructions assumed the action "push" did not require a distance or any explicit stopping point (i.e., no one told Gogol how far or long to push or when to stop). Most problematically, some responses suggested that their speakers held incorrect assumptions about the robot, in particular a lack of trust in the robot's ability to detect closed doors. In the case where the robot was instructed to push on "every wall [it] finds". the robot may find that it follows instructions perfectly, but still is unable to complete its goals in a timely manner.

In short, it is clear that in its current state, the system is unlikely to work with completely naive users, as they don't understand the key elements that a robot needs to integrate and use a new command. For scenarios such as the one envisioned here, this is probably not a fatal shortcoming, as search and rescue robot operators can reasonably be expected to receive training that affords them a level of familiarity sufficient to understand the relevant issues without, requiring that they have a developer's view of the architecture. On the other hand, the capabilities described here

could also be of great general utility as robots become increasingly common in our society. The results of this small user study suggest that this will require substantial improvements to the architecture.

6. CONCLUSIONS AND FUTURE WORK

We presented a robotic architecture featuring a planner that uses discovered information to produce new and updated plans. The robot can learn action sequences with defined pre- and postconditions from natural language descriptions, and immediately apply this knowledge to improve planning. We demonstrated a successful real-world implementation on a physical robot and showed that the system is scalable to a large set of possible inputs, but found that it is unable to handle input from naive users with limited understanding of the robot's capabilities. In addition, we noted limitations in the planner's update process.

As the initial evaluation revealed, naive users are not necessarily effective at providing sufficient information to fully specify the learned action sequence; additional mechanisms will likely be required to successfully deploy the architecture for use with the general public. We are considering two approaches to this problem: detecting missing information in operator descriptions during the natural language processing phase so the robot can request clarification, and performing a "mental" simulation with the new capability to determine whether it seems likely to be effective in the real world.

When the planner's world knowledge and/or model knowledge is updated, the current plan is replanned from scratch, which as previously described is not necessarily the most effective method. Future work will include evaluating the possible ways of deciding whether and how to reevaluate a plan and implementing the most successful procedure.

7. ACKNOWLEDGMENTS

This work was in part funded by ONR MURI grant #N00014-07-1-1049.

8. REFERENCES

[1] Sugato Bagchi, Guatam Biswas, and Kazuhiko Kawamura. Interactive task planning under uncertainty and goal changes. *Robotics and Autonomous Systems*, 18(1-2):157–167, 1996.

[2] Rehj Cantrell, Paul Schermerhorn, and Matthias Scheutz. Learning actions from human-robot dialogues. In *20th IEEE International Symposium on Robot and Human Interactive Communication*, 2011.

[3] Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. Robust spoken instruction understanding for HRI. In *Proceedings of the 2010 Human-Robot Interaction Conference*, March 2010.

[4] William Cushing, J. Benton, and Subbarao Kambhampati. Replanning as a Deliberative Re-selection of Objectives. *Arizona State University CSE Department TR*, 2008.

[5] Minh Binh Do and Subbarao Kambhampati. Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, 20(1):155–194, 2003.

[6] Kathleen Eberhard, Hannele Nicholson, Sandra Kübler, Susan Gunderson, and Matthias Scheutz. The Indiana "Cooperative Remote Search Task" (CReST) Corpus. In *Proceedings of LREC-2010*, Valetta, Malta, 2010.

[7] Russell Knight, Gregg Rabideau, Steve Chien, Barbara Engelhardt, and Rob Sherwood. Casper: Space exploration through continuous planning. *IEEE Intelligent Systems*, pages 70–75, 2001.

[8] Hadas Kress-Gazit, Georgios Fainekos, and George J Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.

[9] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. From structured english to robot motion. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, 2007.

[10] Pat Langley, Nishant Trivedi, and Matt Banister. A command language for taskable virtual agents. In G. Michael Youngblood and Vadim Bulitko, editors, *AIIDE*. The AAAI Press, 2010.

[11] Solange Lemai and Félix Ingrand. Interleaving temporal planning and execution: IxTeT-eXeC. In *Proceedings of the ICAPS Workshop on Plan Execution*. Citeseer, 2003.

[12] Silvia Richter and Matthias Westphal. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[13] Paul E. Rybski, Jeremy Stolarz, Kevin Yoon, and Manuela Veloso. Using dialog and human observations to dictate tasks to a learning robot assistant. *Journal of Intelligent Service Robots, Special Issue on Multidisciplinary Collaboration for Socially Assistive Robotics*, 1(2):159–167, April 2008.

[14] Paul E. Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. Interactive robot task training through dialog and demonstration. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 49–56, New York, NY, USA, 2007. ACM.

[15] Matthias Scheutz, Paul Schermerhorn, James Kramer, and David Anderson. First steps toward natural human-like HRI. *Autonomous Robots*, 22(4):411–423, May 2007.

[16] Kartik Talamadupula, J. Benton, Rao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology*, 1(2), 2010.

[17] Kartik Talamadupula, J. Benton, Paul Schermerhorn, Rao Kambhampati, and Matthias Scheutz. Integrating a closed world planner with an open world robot: A case study. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI '10)*, Atlanta, GA, July 2010.

[18] Kevin Yoon and Paul E. Rybski. Teaching procedural flow through dialog and demonstration. In *Proceedings of IROS'07, the 2007 International Conference on Intelligent Robots and Systems*, pages 807–814, San Diego, CA, Oct–Nov 2007.