RESEARCH-ARTICLE

"Do This Instead"—Robots That Adequately Respond to Corrected Instructions

CHRISTOPHER THIERAUF, Tufts University, Medford, MA, United States

RAVENNA THIELSTROM, Tufts University, Medford, MA, United States

BRADLEY OOSTERVELD

WILL BECKER

MATTHIAS SCHEUTZ

Open Access Support provided by:

Tufts University

Published: 26 August 2024
Online AM: 22 September 2023
Accepted: 07 August 2023
Revised: 08 July 2023
Received: 15 June 2022

"Do This Instead"—Robots That Adequately Respond to Corrected Instructions

CHRISTOPHER THIERAUF and RAVENNA THIELSTROM, Human-Robot Interaction Lab, Tufts University, USA
BRADLEY OOSTERVELD and WILL BECKER, Thinking Robots, Inc, USA
MATTHIAS SCHEUTZ, Human-Robot Interaction Lab, Tufts University, USA

Natural language instructions are effective at tasking autonomous robots and for teaching them new knowledge quickly. Yet, human instructors are not perfect and are likely to make mistakes at times and will correct themselves when they notice errors in their own instructions. In this article, we introduce a complete system for robot behaviors to handle such corrections, during both task instruction and action execution. We then demonstrate its operation in an integrated cognitive robotic architecture through spoken language in two tasks: a navigation and retrieval task and a meal assembly task. Verbal corrections occur before, during, and after verbally taught sequences of tasks, demonstrating that the proposed methods enable fast corrections not only of the semantics generated from the instructions but also of overt robot behavior in a manner shown to be reasonable when compared to human behavior and expectations.

CCS Concepts: • **Computing methodologies** → **Cognitive robotics**; **Discourse, dialogue and pragmatics**; • **Computer systems organization** → **Robotics;**

Additional Key Words and Phrases: Human-robot interaction

ACM Reference format:
Christopher Thierauf, Ravenna Thielstrom, Bradley Oosterveld, Will Becker, and Matthias Scheutz. 2024. "Do This Instead"—Robots That Adequately Respond to Corrected Instructions. *ACM Trans. Hum.-Robot Interact.* 13, 3, Article 40 (August 2024), 23 pages.
https://doi.org/10.1145/3623385

1 INTRODUCTION AND MOTIVATION

Natural language interactions with robots are becoming increasingly important both for tasking robots and for teaching them new knowledge (e.g., Reference [59]) as they enable more natural human-robot interactions (e.g., Reference [53]) and faster learning (e.g., Reference [51]). Most proposals for natural language understanding algorithms in robotic architectures, however, assume that human instructions are "perfect," i.e., *grammatical* (with no disfluencies and ungrammatical constructs), *sufficient* (for understanding or carrying out the task), and *consistent* (in their description of the knowledge or task). Yet, human instructors are not perfect and are likely to

This work was in part funded by ONR Grants No. N00014-22-1-2206 and No. N00014-18-1-2831.
Authors' addresses: C. Thierauf, R. Thielstrom, and M. Scheutz, Human-Robot Interaction Lab, 177 College Ave, Medford, MA, 02155; e-mails: {Christopher.Thierauf, Ravenna.Thielstrom, Matthias.Scheutz}@tufts.edu; B. Oosterveld and W. Becker, Thinking Robots, 12 Channel St #202, Boston, MA, 02210; e-mails: {Brad, Will}@thinkingrobots.ai.

2573-9522/2024/08-ART40
https://doi.org/10.1145/3623385

produce ungrammatical, insufficient, and inconsistent instructions at times, throwing off natural language understanding systems. More importantly, humans might change their mind as they are talking, which can lead to different forms of "corrections," for example, corrections within an utterance as in "get me the red, uhm, no the blue box." While such corrections can be handled by ignoring these errors during semantic parsing (e.g., Reference [7]), a robot agent may require more complex methods for recovery if execution is already underway. For example, consider the case when a robot is given an instruction to move items from one place to another, and the instructor has a change of mind once some items have already been moved. In that case, it is not just a matter of fixing the semantics of an instruction, changes have been affected in the world already and might have to be undone (if possible) to carry out the modified instruction.

There is currently little work on behaviors that address human corrections that either occur during robot tasking or during robot teaching.[1] The goal of this article is to propose general methods for handling human corrections that can occur both during initial teaching or tasking (prior to robot action), and during task execution (while the robot is already performing actions). We keep these methods general and discuss their operation in a manner agnostic to any particular architecture, though we also discuss our particular implementation.

The main contribution of this article is a system demonstrated on two different fully autonomous instructible robots, which can handle human corrections via the integration of (1) natural language semantics and associated methods for expressing and detecting corrections during task instructions and action execution and (2) novel methods for determining the appropriate corrective action by the robot. These methods will consider whether an already started action can be aborted and the corrected action can be immediately started, or whether already accomplished states in the environment must be undone first.

We start by providing background on dialogue-based natural language interactions and ways in which humans might correct themselves. We then describe the methods for detecting and handling human corrections as they are implemented within a cognitive robotic architecture depending on whether the correction occurs before execution, i.e., during the initial tasking or teaching phase, or during execution when the world state has already been potentially changed by the robot. We then describe a collection of proof-of-concept settings on different autonomous robots with different tasks, demonstrating the operation and utility of the proposed methods. We also report the results from a user study probing human understanding of corrective utterances that shows a close match with our proposed methods, with a 90% or better overall agreement. We conclude with a brief discussions of limitations and future work.

2 BACKGROUND AND RELATED WORK

Over the past decade, natural language-enabled robots have become an increasingly important focus of research in robotics as demonstrated by a recent review of work in that area entitled "Robots that Use Language" [59]. However, we observe that while the word "robust" is used in multiple places in the survey article, and our past work on robust instruction understanding is cited [7], there is little information in the text about attempts to make instruction understanding on robots more robust. This indicates to us space to explore regarding the problem of dealing with human corrections. Anecdotally, these systems are often "all or nothing": the utterance is either wholly correct, or the speaker is forced to deal with their error. Yet, human corrections in task-based contexts, often accompanied by disfluencies, are not at all uncommon. In our CReST

[1]While we will discuss existing literature in this space (Section 2), the bulk of corrections literature focuses on policy shaping and/or disfluencies. In contrast, we are interested in explicitly instructed robot behaviors and how these may be modified with a single instruction.

corpus [16], participants worked in human-human dyads to conduct a search throughout a series of rooms, with transcripts explicitly labeled with disfluencies (as standardized by Reference [35]). In Nicholson et al. [41], we further refined our analysis of the corpus, concluding that disfluencies and self-corrections are extremely common in collaborative tasks. Because of the key element these repairs play in communication, they need to be handled by robots.

Based on the human task-based dialogues in the CReST corpus, we then developed parsing methods for dealing with disfluencies and corrections within a single utterance [7]. These methods filtered disfluencies such as "uhms" or "ahs," repetitions like "the the," and were also able to correct so-called *false starts* as in "pass me the green, ah, no, ah, red box" (where the corrected utterance should simply be "pass me the red box"), which are more difficult to correct. Critically, these methods were all handled and implemented within the scope of only the utterance being actively processed.

In fact, much of the self-correction literature falls into this category: detecting disfluencies has been explored in the "**Transcript Disfluency Detection (TDD)**" domain. Promising solutions have been demonstrated using **Long Short-term Memories (LSTMs)** [61, 63], decision trees [57], neural machine translation models [14], and self-attentive neural syntactic parsing models [37]. With disfluencies detected, they can be addressed: the outputs of these approaches can become a part of a broader natural language processing system, allowing disfluencies to be parsed out and processing to continue. All of the above cited systems are non-embodied, but with some modifications they might be applicable, and, in general, any system that can interpret human intent more robustly would be applicable to our robot scenarios if integrated appropriately into the robotic architecture.

2.1 Natural Language Challenges of Human Instruction Corrections

However, we are not interested in disfluencies here. Unlike the previously cited works focusing on disfluency correction, we focus on a more challenging case where the instruction is valid by itself, but may be modified across instructions by a later utterance. These cases are more challenging because a parsing-only approach is not suitable: as we will show, multiple components across the robot architecture must work in tandem to resolve these cases. As members of a team communicate changes in the environment and goals, they will correct themselves, often mid-utterance or mid-task. This presents both a challenge in parsing and in robot behavior. While this article focuses on behavior, the unique challenge for the parser is worth considering.

For example, consider the instruction "pick up a box from the white table," followed by the correction "actually, pick up a box from the blue table." There are no disfluencies that the parser needs to repair here. Additionally, there are no cross-utterance dependencies as there would have been if an anaphor had been used in the second instruction (such as referential dependencies, as in "actually, pick one up from the blue table") or if the second utterance had been an ellipsis ("actually, from the blue table"). Instructions in both utterances are self-contained, but the word "actually" indicates an update to the previous instruction. Without this signifier, the listener would interpret the utterance as a new instruction to be completed in sequence, and not a correction. Note that this is different from explicitly canceling a previous instruction (e.g., "stop," "do not pick up a box from the white table anymore," "cancel current goal," etc.). Expressions like "actually," which signify that a correction is being made, are known as "editing terms" within linguistic research on self-repairing speech. Levelt [34] found that this specific type of editing term, as opposed to a disfluency or filler word like "uh," generally signified a fresh start in a correction: the entire utterance is restarted, rather than simply a word being corrected mid-sentence.

Work focusing on making natural language systems robust against these corrective utterances has been examined by previous researchers. Lemon and Gruenstein [33] builds on top of the work

in Lemon et al. [32], which (in part) discusses work allowing for robot instructions to be modified or corrected. They use a series of dialogue rules that allow for ambiguity corrections through dialogue as well as revisions (though their revision process is more limited to canceling and replacing behavior than ours). This is highly relevant to our presented work in that it does incorporate handling of corrective utterances within a robot software stack. The OntoAgent architecture [17] is also shown in Nirenburg and Wood [42] to be capable of handling corrections, as the OntoAgent architecture is capable of updating belief from natural language. Robot behaviors requiring modification are also explored in Appelgren and Lascarides [1] and She et al. [56], though neither involves correction, instead assuming no human error. Where we most differ with all these approaches, however, is in our approach's ability to perform the logical reasoning necessary to handle corrections involving changes to the environment, as well as performing corrections within a learned sequence of robot actions.

2.2 Instructing Robot Behavior in Moment-by-Moment Execution

Approaches to interactive task learning that incorporate correction largely focus on robots being corrected by humans on their behavior, as in Appelgren and Lascarides [2], Losey and O'Malley [36], and Fitzgerald et al. [18], whereas we focus on humans directly correcting their own error. Appelgren and Lascarides [2] does this using natural language in a rule-learning situation, similar to us, but assumes that there is no human error. In the approach to task learning presented by Losey and O'Malley [36], a robot can estimate the most preferred path based on corrections to its behavior that a human has made. As part of this process, it will also calculate its own uncertainty about its estimates of the human's preference. If a human accidentally corrects the robot down the wrong path, then corrects the robot down the right path, the robot's uncertainty about the human's preference will likely raise, prompting the robot to seek out more corrections that might clarify the matter.

However, the approach described in Losey and O'Malley [36] potentially requires multiple runs of the same task over time, making it more similar to previous works that have used language as a reward shaping strategy. This area of research has seen a wealth of recent developments, surveyed further in Luketina et al. [38]. In these approaches, the agents use spoken language to shape intended behavior of a reinforcement learning policy. For example, Matuszek et al. [40] presents an early example of using natural language to construct a reward function, which can be used for later training, as do Squire et al. [58] and Goyal et al. [21]. In the recent NeurIPS "**Language and Reinforcement Learning (LaReL)**" workshop, Röder and Eppe [47] propose a system in which reinforcement learning is used to perform incremental corrections of actions; much like Caselles-Dupré et al. [9], who demonstrate (again in a simulated environment) how a vague instruction can still be resolved through language for the benefit of the reinforcement learning agent. Notably, however, reinforcement learning approaches focus on the construction of a reward function, and, therefore, only produce a single behavior over the course of many corrections. In contrast, our approach focuses on explicitly grounded behaviors, which may include the selection of a trained policy but may also be based on some previously defined behavior.

Similar works have gone on to emphasize the value of interpreting human self-corrections in robot behavior, and have produced systems prioritizing this functionality. In Lynch et al. [39], the authors describe a robot system that allows natural language to specify complex behaviors, and they state interpretation of human corrections as a priority. However, the corrections they provide as examples are fully syntactically complete, e.g., "move the block away from the edge." Thus, they do not provide an explicit strategy for reasoning about ambiguity as we do. Works that focus on policy shaping can be viewed as handling self-corrections (e.g., using language to modify a policy producing kinematic trajectories [5, 6, 55]), but again fail to provide an explicit strategy for

reasoning about ambiguity. In further contrast to these approaches, we are not assuming a policy or past experience that must be refined over a series of language interactions. Rather, by using explicit representations of language and grounded behaviors, our approach repairs the robot's understanding instantly.

Handling ambiguity across utterances is an additional distinction between our work and the current state of the art. A notable exception is Broad et al. [4] (later reported on in Broad et al. [3]), which can handle multi-utterance corrections. In one example from their paper, the robot is instructed to "pick up the box," and then later told "from the top," showing that their system does handle these context-dependent multi-utterance corrections. Similarly, Cui et al. [13] introduces "LILAC" to produce a system capable of handling trajectory corrections to a low-dimensional control space, specified through real-time natural language. However, these are each trajectory correction tasks, while our approach focuses on language and task agnostic interpretation of the most feasible candidate for corrected behavior.

2.3 Moment-by-Moment Action Modifications

Human corrections can occur either in tight task-based dialogue interactions, or in teaching interactions. We demonstrated the former in a system reported in Scheutz et al. [49], in which the robot was able to handle the disfluencies through parsing, using a real-time, embodied, situated natural language system, in a way that also utilized contextual information.

The latter case of teaching interactions is easier to handle if a robot learns from instructions without executing any actions. Consider a case of this type where a robot learned an erroneous script (either due to human error or error on the robot side): here the human can simply give the robot conditional instructions for accessing the script, locating the entry that needs to be corrected, and the proper correction for it. For example, suppose the robot has been taught how to pass a knife to a human instructor, but the human instructor did not explicitly tell the robot to pass the knife forward, toward the person. When asked to execute, the robot will extend the arm with the knife straight up, regardless of where the humans hand is to receive it. A simple instruction can then be used replace the erroneous action with the corrected one (see Frasca et al. [20] for details):

> Brad: When you pass me the knife, replace move the knife forward with move the knife toward me.
> Robot: OK.

The result is an updated action script with the correct action parameter in the "`moveObj`" actions:

"`moveObj(self, knife, toward(Partner))`"

instead of simply

"`moveObj(self, knife, forward).`"

Now consider the case where the script is learned in tight dialogue interactions where the robot immediately starts to execute instructed actions. Note that the natural language processing aspects for determining the appropriate correction are roughly the same (even though the corrective utterance can be much shorter, as repeated context is often omitted for convenience). However, the execution aspects can be very challenging, because what the robot should do will depend on several factors. These factors include the time when the correction is issued in the instruction sequences, as well as how far into the instruction sequence the execution has advanced already. For example, in the above case, suppose the robot has already learned two versions of "pass," one with the additional target location argument "toward(X)" and one without. If the person simply said "pass me the knife" and while the robot is picking it up the correction "actually, pass it toward

me" is uttered, then the robot's motion planner should immediately make the path correction. Instead of following a straight trajectory, it has to query the vision system about the location of the interactor's hand and then quickly plan a path toward it.

Such a case may require implementation of custom execution strategies (e.g., a custom kinematic planner to enable on-the-fly corrections as in Park et al. [43]). In others, the speed of the task relative to the speed of computing the task may allow this to not be necessary. In the case of two-dimensional navigation of robot platforms, for example, planning a new path is fast enough that a new trajectory can be computed and begun before the ongoing one has completed. High-speed kinematic planning systems in wide use in the robotics community include Fox et al. [19], Kuffner and LaValle [26], and Kalakrishnan et al. [25]. Our approach benefits from works of this nature (our demonstrations specifically make use of Fox et al. [19]), but these works focus around the implementation of a single robot behavior. In contrast, we are more interested in the higher-level control of robot behavior: we assume that a robot with some set of behaviors exists, and we focus on how such a system may handle these more general behaviors being interrupted or changed.

3 AN INTEGRATED ARCHITECTURAL APPROACH FOR HANDLING HUMAN CORRECTIONS

As discussed in Section 2, recovery from human instruction errors that span multiple utterances is a complex process that can involve many components in an agent architecture. Relevant systems range from natural language understanding to dialogue management and task planning, but may also include robot controllers and low-level motion planners.

In this section, we first describe our architecture agnostic approach to handling corrections. This description includes the process used for detecting corrective utterances, and also how those utterances are handled, once detected. We consider the differences between corrections during teaching and tasking, and we pay special attention to the important tasking sub-case of the correction occurring after execution of the instructed task has begun. We conclude by discussing how our approach can be realized in a cognitive robotic architecture.

3.1 Architectural Requirements

While the demonstrations shown in Section 4 were implemented using the "**Distributed Integrated Affect and Reflection Cognitive (DIARC)**" Architecture, the proposed approach is viable for a wide variety of cognitive architectures that meet the following requirements:

(1) A human-robot communication modality (e.g., spoken language);
(2) A strategy for using communication to receive instruction and learn new behaviors;
(3) A safe and responsive action halting capability;
(4) A state-based representation of the environment for planning and perception.

The first requirement is a communication mechanism between the human and robot. More specifically, this communication mechanism must enable the robot to receive instructions and interpret these instructions as requested behaviors. While we make use of spoken language, other communication strategies are also viable (such as in the previously cited Lemon and Gruenstein [33]). We also recommend that the robot be able to demonstrate its understanding of the requested goal to maximize human faith in the robot's intent to correct the behavior: we again make use of spoken language, in the form of the robot saying "OK" when an utterance is successfully mapped to an executable action.

Second, the ability to modify a learned sequence of actions requires the robot to be capable of learning these sequences ("action scripts") in the first place. No other portion of our implementation relies on this functionality, so if action script learning is not desired this need can be omitted.

However, we find these cases (particularly the one-shot learning scenario) to be particularly interesting and valuable, because this enables more effective interactions between human-robot teams by communicating increasingly complex instructions. This is particularly true when the learning of action script occurs in one-shot while leveraging an agent's existing knowledge about itself or its environment. One-shot action script learning is discussed further in Scheutz et al. [52].

With the ability to execute and obtain actions, the architecture thirdly needs the ability to halt action execution. This must occur as quickly as safety and human comfort will allow; otherwise, we run the risk of human disengagement with the overall system.

To perform actions that address changes in the environment, the architecture also needs to be capable of some degree of environment interpretation. Additionally, the architecture will need some form of planner. This environment interpretation must occur in a manner that can be provided as an input to this planner, which will receive this information alongside the new goal state specified by the aforementioned communication modality. To facilitate this, a planning language is used to describe knowledge of the environment, current and goal states, and actions available to the agent. In our implementation, we make use of the widely used **Planning Domain Definition Language (PDDL)**. This domain can then be solved by any PDDL-based planner (such as the "Fast-Forward" planner [24] we use).

There is a variety of architectures that can meet these needs and some widely discussed architectures that already do. The Soar cognitive architecture uses a symbolic representation of the world [27], and has been deployed on embodied agents [22, 28] with language interfaces [44]. While it does not perform planning in the classical sense (it instead uses a "decision cycle" [12, 29]), its explicit symbolic representation of progress toward a goal makes it equally feasible. The ACT-R/E architecture [60] provides an embodied version of the ACT-R architecture [46], and as discussed in Tellex et al. [59], this produces a robot capable of planning, interacting with the environment, and communication via speech. The ICARUS Architecture [30] has also been deployed on embodied agents (see Choi et al. [10], and also our work on integrating ICARUS into DIARC [50]). Although ICARUS does not prioritize human-robot language interfaces [11], there is nothing about ICARUS that prevents this, and so this functionality could feasibly be provided.

From this, we observe that it is useful to make use of an existing system, but that our approach is not tied to DIARC. These requirements remain broadly available and could feasibly be implemented in any of the discussed alternative architectures (and realistically many others, including those surveyed in Chong et al. [12]). We chose DIARC for mostly pragmatic reasons: the core features we require are already implemented in the form of comprehensive components integrated on a large variety of robot platforms [54]. Because our approach does not violate any of the assumptions DIARC makes to keep itself robot-agnostic, it is reasonable for us to conclude that our approach is also applicable to a variety of robot platforms (as we will demonstrate).

3.2 DIARC

DIARC, the "Distributed Integrated Affect and Reflection Cognitive" Architecture (described in more detail in References [48, 52, 54]) forms the base on which our demonstrations are implemented. We leverage several components that are already implemented in DIARC, as these meet many of the previously stated architectural requirements (see Figure 1).

We first utilize DIARC's semantic language parsing system: after collecting utterances using the off-the-shelf CMU Sphinx4 parser [31], utterances are converted to semantic representations as introduced and discussed further in Dzifcak et al. [15]. We combine this simple speech-to-text with a pragmatic inference system while parsing, which allows potentially vague statements to be resolved to explicit references that the agent can employ: words like "it" are linked to the appropriate referent, and references like "the block" are grounded to explicit world knowledge, if possible.

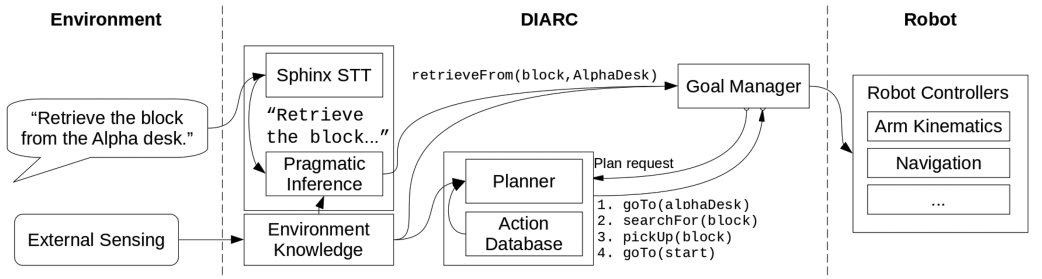


Fig. 1. A high-level control flow through parts of the DIARC architecture (rectangular boxes) used as an implementation platform for the proposed correction handling methods.

The end result of this process is a fully grounded request for behavior, which can be passed to the goal management system.

The goal manager is responsible for taking these requests for behavior and producing explicit executable actions that the robot can perform. The goal manager component handles both high-level goals derived from natural language instructions (e.g., "move the object to this position") as well as the executing of skills to satisfy these goals (e.g., "move gripper to coordinate," "close gripper").

When a desired behavior is determined, the goal manager is responsible for tracking which stage of progress the agent is currently in and should be executing. It does so with the assistance of a planning system (the "Fast Forward" planner [24]) and a known set of possible actions (the "action database"). The goal manager retains information about pre- and post-conditions of individual behaviors [8] in the form of a first-order logic. These anticipated state changes are represented alongside explicit observations from the environment (e.g., from a vision processing pipeline). Thus, the changes in the environment can be predicted and tracked (in addition to being explicitly observed). This combination of observation of the environment alongside predicting the environment meets the needs of describing problem domains in PDDL: the current state of the agent is known and hence can be provided as the start state; the goal state has been provided as the instructed goal; actions have been provided by the action database; action transition functions have been provided by the pre- and post-conditions of those same actions. This constructed problem domain can be solved by the planner. If a plan is found, then it is returned as a sequence of known robot behaviors.

Finally, the agent behaviors are executed on the robot thanks to manufacturer-provided hardware interfaces and widely used implementations of robot behavior strategies. In our implementation, we make use of ROS [45] to interface with the robot control software provided by the Fetch mobile manipulation platform [62]. In the case where the environment has deviated from the desired state and requires multiple steps to resolve, the post-conditions help to form the input to the planning and re-planning process, as we will outline. However, at all stages of operation the pre- and post-conditions are being monitored for fault detection at minimum. The Goal Manager component also performs the one-shot action learning from natural language instructions, which we modified to handle corrections during teaching appropriately. With all this functionality in place, DIARC is capable of taking complex spoken commands, interpreting their intent, and performing a series of robot behaviors that meet the needs specified by a human instructor.

3.3 Detecting Corrections in Human Instructions

For an agent to modify its behavior to reflect a corrective utterance, it must first be detected. This process requires both understanding of the semantic meaning of the utterance, as well as its context in the larger interaction between human and robot. A correction by itself is meaningless if the robot does not also understand the error that is to be corrected.

The detection process is initiated by interpreting the semantic meaning of a received utterance. In our particular implementation, we have focused on corrections that involve "fresh starts," where the incorrect command has already been parsed and potentially has already begun to be executed by the agent, and the correction comes in the form of a new, restarted utterance. There are several *direct* indicators in natural language of "corrective" instruction that we consider here.[2] The most obvious of these editing terms are adverbial modifications to the new instruction (i.e., "actually"), but correction can also be implied in preceding the new instruction with a negatory ("no, try the white table") or an interjection ("sorry, pick up the block from the white table").

To identify these key phrases in the parser, we used Heeman and Allen [23]'s parsing of a problem-solving dialogue corpus, in which there are 671 fresh starts and 1,128 editing terms. Heeman and Allen provide a list of all editing terms used more than once; by selecting only the ones that are significantly featured in fresh starts, we pruned their list down to the following: "no," "sorry," "I'm sorry," "wait," "excuse me," "oops," "I mean," "that is," "well," "actually," "I guess." As they discuss, these occur most frequently at the beginning of a fresh start, but can also occur at the end. We therefore implement support for parsing these explicit correction indicators at both the beginning and end of an utterance.

These terms are all flagged by the natural language understanding system as a fresh start correction-type utterance rather than an ordinary instruction. This flagging triggers the next step in the recovery process. If the new utterance shares structural elements with the speaker's last utterance, then we confirm its status as corrective (this weeds out potential utterances like "Wait for me to give you the block," which use our keywords in a non-corrective context).

While this approach is limited—not all fresh starts involve these editing terms, or any editing terms at all—we argue that this is an acceptable drawback as we are focused in this article on exploring what an autonomous agent should do *after* it receives a corrective utterance to a previous instruction. For a more comprehensive treatment of detecting such instructions, see Lemon and Gruenstein [33].

Once an utterance is identified as corrective, the semantics of the rest of the utterance are modified to imply that the speaker intends for the robot to cancel its current goal and replace it with a new, potentially similar, goal. Even if the current behavior of the robot need not be changed by the correction, we still wish to cancel and replan in case the new information provides any alternate, preferred methods of achieving the goal. After detection has occurred and the semantic representation of the utterance has been updated, the full details of the corrective utterance can be appropriately handled by the architecture.

3.4 Handling Corrections During Teaching

The context in which a correction is uttered by a human interlocutor affects the semantic intent of that utterance. A correction that occurs while teaching the representation of a new task is different than a correction that changes a given task instruction to a different instruction.

Consider the following correction specified when teaching a task requiring navigation to "position y":

> Human: Go to position x.
> Robot: OK.
> Human: Actually, go to position y.
> Robot: OK.

[2]There are also *indirect* indicators that have a different function based on context such as "maybe get a different one," "maybe wait for me to be closer for handing it over," etc. The indirect indicators require more reasoning and are harder to realize. For that reason, we will focus on editing terms.

If this sequence is retained as instructed, then the nature of an action instruction system combined with a naïve cancellation approach will cause the action to be saved as (1) go to an incorrect location, (2) cancel that action, and (3) go to the correct location. To avoid this extraneous behavior, the action learning process must be modified to exclude goal cancellation and to properly prune the set of retained actions in the script. In this case, we interpret a corrective utterance as an instruction to re-assess the prior goal. To ensure that this process occurs with the full context of the overall action script, this occurs when learning of the action script concludes. We use this as an opportunity to perform any necessary cleaning of the script before saving for later use.

3.5 Handling Corrections While Executing

If a corrective utterance is received mid-action, then the ongoing action must first be stopped so that the behavior can be modified. The next step is to determine, given both the ongoing action and the instructed action, what the appropriate robot behavior is. Given the nature of the utterance, it is safe to assume that the speaker wishes for this new instruction to be carried out in place of the existing action. However, the execution of the original action may have produced now-unintended side effects. We will initially describe the system operating prior to any side effects taking place, before expanding into that set of cases in Section 3.6.

The easiest way to prevent unintended side effects of the first action is to stop the execution of the first action as quickly as possible. The first step in canceling the unintended behavior is determining what exactly the speaker is trying to correct. To do this, the natural language understanding system must locate the corrected instruction in the dialogue history of the conversation, and determine the combined pragmatic meaning of that original instruction and its correction. If the correction in itself contains a complete instruction, i.e., "actually, pick something up from the Alpha table," then no information from the original instruction is necessary, and the new goal can be constructed and submitted normally. In other cases, however, the correction may be abbreviated as an instruction with an ambiguous placeholder verb ("actually, *use* the Alpha table"). A substitution must be made in the original instruction that will replace the appropriate section in the original instruction with the new information.

Substitution in these cases is helped not only by matching morphological representations (i.e., "table" in "the alpha table" and "the beta table"), but also by the use of "types." During the parsing process, every lexical symbol in an instruction is labeled with its syntactic and semantic type. This means that during the pragmatic processing of a correction, the syntactic and semantic types can be used to determine what needs to be substituted in the original instruction.

Isolating the non-placeholder parts of the correction results in "the alpha table," which carries the semantic type of "Location." In the original instruction, "pick up the block from the beta table," "the beta table" is also semantically typed as a "Location," distinguishing it from "block," which is an "Object." Assembling a complete instruction of "pick up the block from the alpha table," the pragmatic understanding system can thus pass this corrected instruction on to be submitted as a new goal.

This approach operates in all cases where the robot is currently following an action but has not yet made any changes to its external environment. Examples include "drive to location x," "look at object y," and may include "grab object z" (on the condition that the utterance occurs prior to the object being grabbed).

3.6 Handling Corrections After Environment Changes

In many cases, or even most cases, a corrective utterance will not be received before robot behavior has taken place. Making progress on the initially specified goal may have introduced effects that are now undesirable. In these cases, the intention of the corrective utterance is to both achieve the

newly specified state of the world, as well as to undo the erroneous instruction to the minimum extent necessary.

When a correction is detected for an action that is currently being executed, the execution is first halted to re-assess the current state of the world so that next steps toward the new goal can be planned. The state of the world is a set of facts in the form of logical predicates that describe the world and the outcomes of various actions. These predicates, being informed by the architecture's observations and belief system, are leveraged to produce a symbolic representation of the current world in the same way goals are specified.

Some effects of the robot's actions may not be relevant to the intent of the instruction: for example, precise joint angles or object positions. These values are not represented as logical predicates and are not considered: the agent considers the logical predicate representing the state (for example, `on(plate,soda)`) and not the measured object positions that produce this state. It is acceptable to ignore information like this, because they are unrelated to the intention of the goal.

By comparing the set of predicates in the goal state and the set of predicates in the known world state, the portions of the agent's environment that are in conflict with the new goal can be explicitly represented: the agent is aware of the predicate logic representing its current state, and the predicate logic representing the new goal state. It is then a reasonably trivial logical operation to determine which aspects of the current state are not relevant to the future goal and can be considered in conflict. To correct the unintended side effects, these conflicting predicates are negated to construct a goal environment that does not contain the conflict (e.g., `on(plate,soda)` is erroneous, and so we produce `not(on(plate,soda))`). This goal environment is submitted to the task planner as a new goal state prior to the execution of the new instruction. Once the undo goal has been achieved, the world has been returned to the state it was in prior to the initial instruction. The agent is now at a "clean" state, as though the conflicting portions of the environment state had never been reached. The planning problem can now resume, in which it attempts to go from this "clean" state to the new end-goal. Using this approach, any state that can be reached by the planner can now be achieved by the agent. This does make the critical assumption that states can be undone, which is not always a safe assumption: something broken cannot be trivially un-broken, and something communicated cannot be un-communicated. However, many robot tasks are easy to undo the effects of movement, assembly, picking/placing, and many others can be reversed in a small number of actions. Assuming the states are produced by behaviors in this category, they will have been addressed.

4 PROOF-OF-CONCEPT EVALUATION SCENARIOS

We observe two distinct sets of cases when executing corrections, and demonstrate several scenarios of increasing complexity. In the first set, there have not been changes to the environment. The process of correcting, then, becomes largely rooted in the syntactical parsing of the correction to modify a single behavior. Examples of this nature are presented as Scenario 1, Scenario 2, and Scenario 3. In Scenario 1 (Section 4.1.1), a navigation goal is specified and then corrected. In Scenario 2 (Section 4.1.2), a taught task is corrected mid-teaching. In Scenario 3 (Section 4.1.3), the robot begins to follow a pre-determined sequence of events, but one step within the sequence is changed during execution.

In the second set of cases, the environment has changed. The act of correcting the current behavior will therefore require undoing certain actions to produce the desire end state before resuming the task. To demonstrate a case of this nature, we conclude with Scenarios 4 and 5 (Section 4.2), where collecting a sequence of objects must be undone to obtain the newly desired collection. All demonstrations are also provided in video format.[3]

[3]https://youtu.be/e3Q6UlpiYxw

4.1 Scenarios 1 to 3: Action Substitution

We first observe the implementation of this approach in a standard robotics stack on a mobile manipulator. The human and robot work to retrieve an object from one of two locations, where only the human has the knowledge of which location is the appropriate of the otherwise equally valid locations.

These scenarios all occur in an environment we constructed for the Fetch, a mobile manipulator by Fetch Robotics [62]. In this environment, the two tables with two blocks provide equally valid locations at which to grab an object. The locations of each table (labeled "alpha" and "beta") have been pre-determined and are known to the Fetch through the aforementioned pragmatics approach (Section 3.5), allowing it to navigate to either using its provided SLAM stack.

4.1.1 Simple Action Correction. In the simplest case, the robot is taking a single action that is corrected before completion. We demonstrate this case using the following dialogue:

> Human: Go to the Beta desk.
> Robot: OK. [*Robot starts to drive toward the desk labeled "beta".*]
> Human: Actually, go to the Alpha desk.
> Robot: OK. [*Robot turns and drives toward the desk labeled "alpha".*]

As outlined in Section 3.5, the correction in behavior occurs through cancellation of the current goal followed by over-writing with the new goal. This case is the most trivial to correct, because the robot has not changed anything in its environment, and because there is no ambiguity in the correction that the parser must navigate: the robot can simply begin this new action. The other scenarios, however, will require more consideration on behalf of the robot.

4.1.2 Action Correction During Teaching. In this demonstration, a new action is taught to the robot in the form of a sequence of known actions instructed through natural language to produce a sequence of actions that is saved for later execution (we typically refer to this as an "action script").[4] The correction occurs as a result of being instructed to drive to the incorrect location.

The full dialogue and actions are as follows:

> Human: Retrieve the block from the Alpha desk.
> Robot: I cannot retrieve something from the Alpha desk, because I do not know how to retrieve something from the Alpha desk.
> Human: I will teach you how to retrieve something from the Alpha desk.
> Human: First, approach the Beta desk.
> Human: Actually, approach the Alpha desk.
> Human: Then, grab the block.
> Human: That is how you retrieve the block from the Alpha desk.
> Human: Retrieve the block from the Alpha desk.
> Robot: OK. [*The robot drives to the desk labeled "alpha" and picks up the block.*]

Note that after each human instruction, the robot provides a verbal confirmation of "ok" when successful (omitted for repetition) to help reinforce the human's perception of the robot's understanding. Note also the corrective utterance during the teaching process: the instruction "retrieve the block from the Alpha desk" is erroneously taught with the command to "approach the Beta desk" before being corrected with "Actually, use the Alpha desk."

To execute the correct behavior, the robot trims down this set of instructions to only "use the Alpha desk," performing the necessary substitution to assume the correct action is "approach the

[4] As previously cited, this process is covered in more detail in Reference [52].

(a) Robot has begun execution of "Retrieve the block from the Alpha desk."

(b) "Actually, use the Beta desk."

(c) Fetch performing the action on the Beta desk as corrected.

Fig. 2. Action script correction during execution. The left desk is labeled "Alpha" and the right desk is labeled "Beta."

Action Script 1: `retrieveFrom(`x, y`)`, where x is an object at known location y.

```
approach(y):
    goTo(y)
    raiseArm()
    driveUntilObstacle()
graspObject(x)
```

Alpha desk." The remainder of the instructed task sequence remains unchanged, allowing the robot to then be instructed to perform the full task: the human partner asks the Fetch to "retrieve the block from the Alpha desk," and the Fetch correctly drives to the appropriate desk before grasping.

It is important to note that the corrected behavior is not stored verbatim: rather, the substitution process has correctly mapped the pragmatic reference "Alpha desk" within "actually, use the Alpha desk" to "retrieve the block from the Alpha desk." The value of such an approach is that the behavior is stored in the more general form of a variable: "retrieve from x" will contain "go to x." The learned behavior is therefore generalized such that the human partner could now instead say "retrieve the block from the Beta desk," which would cause the robot to correctly navigate to the Beta desk (despite learning this action in the context of the Alpha desk).

The one-shot learning process then maps these goals to actions by making use of the pre-defined `graspObject(`x`)` and `approach(`y`)` actions. We have constructed our script with only two actions for brevity, but arbitrary instruction lengths are equally valid in our implementation. Additionally, we do not draw a distinction between hard-coded behaviors (such as `graspObject(`x`)`), or behaviors made of other action scripts: `approach(`y`)` is comprised of `goTo(`y`)`, `raiseArm()`, and `driveUntilObstacle()`, although this is hand-written and not instructed. For the full action script, see Action Script 1: `retrieveFrom(`x, y`)`.

This script is now resolved to containing exclusively "primitive" behaviors (atomic, hard-coded functionality) and can therefore be executed by calling `retrieveFrom(`x, y`)` through the utterance "retrieve x from y."

4.1.3 Action Correction of an Action Script. In the final set of cases where no environment changes have occurred, the robot begins with the previously defined `retrieveFrom(`x, y`)` action script. The correction occurs during the execution of this action script, which is provided visually as Figure 2 and is produced by the following language (analyzed as Figure 3):

HUMAN: Retrieve the block from the Alpha desk.
ROBOT: OK. [*Robot starts to drive toward the desk labeled "Alpha".*]

Person says	"Retrieve	the block	from the	Alpha desk."
Parsed as	`retrieveFrom(`	`object`,		`location`)
Resolved to	`retrieveFrom(`	block,		AlphaDesk)
Person says:	"Actually		use the	Beta desk."
Parsed as	Edit			`location`
Resolved to	`retrieveFrom(`	block,		BetaDesk)

Fig. 3. Breakdown of utterances submitted as instruction, and then interpretation as a correction.

HUMAN: Actually, use the Beta desk.
ROBOT: OK. [*Robot turns and drives toward the desk labeled "Beta" and picks up the block.*]

When the human provides the first instruction (to "retrieve the block from the Alpha desk"), the robot interprets this command as `retrieveFrom(block, AlphaDesk)` and begins the action script by executing the command `goTo(AlphaDesk)`. It is as this action concludes that the human specifies the new location of the "Beta desk." To handle this correction, the same substitution process as scenario 2 is used, allowing the robot to infer that "use the Beta desk" should specifically correspond to the action `approach(BetaDesk)` in place of `approach(AlphaDesk)`. The robot then correctly halts the approach to the Alpha desk, instead approaching the Beta desk, before resuming with the remainder of the script by grasping.

Importantly, we do not impose any restrictions on the length of existing action scripts. Instead, searching for ambiguity occurs in reverse chronological order, where the most recent behavior is considered for suitability, then the next most recent, and so on. From this, we can infer that substantially longer scripts are equally feasible: although the script used in this illustrative example only has a length of four primitive actions to be considered, it is reasonable to assume that a larger or shorter length script will remain solveable. However, more thorough verification, with both varying lengths of scripts and varying levels of ambiguity, is necessary to confirm that this heuristic effectively resolves ambiguity and remains within human expectation.

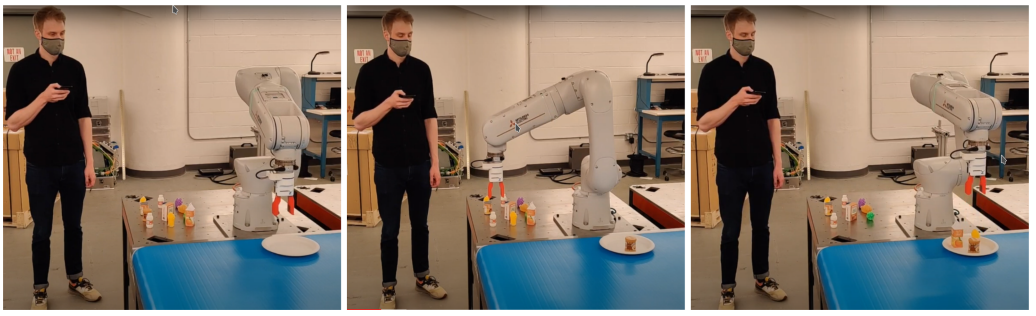
4.2 Scenarios 4 and 5: Correction During Environment Changes

We now present two scenarios that demonstrate corrective utterances that have been received after the execution of the initial instruction has started and that may involve changes to the robot's environment that it must address. In the first scenario (Scenario 4) when the correction occurs, none of the effects of the actions taken thus far differ from the effects of the corrective action. In the second scenario (Scenario 5) the effects produced by actions taken when following the initial instruction are different than the effects of the corrective action.

In these scenarios, a human partner is interacting with a robotic arm (the MELFA ASSISTA RV-5AS-D, and IoT conveyor belt) to collaboratively perform a food packing/assembly task. DIARC has been configured to include a pre-taught delivery behavior. Using this behavior, given the ID of a pre-taught recipe, the robot achieves the world state where all of the items in the recipe are present on the plate on the conveyor belt.

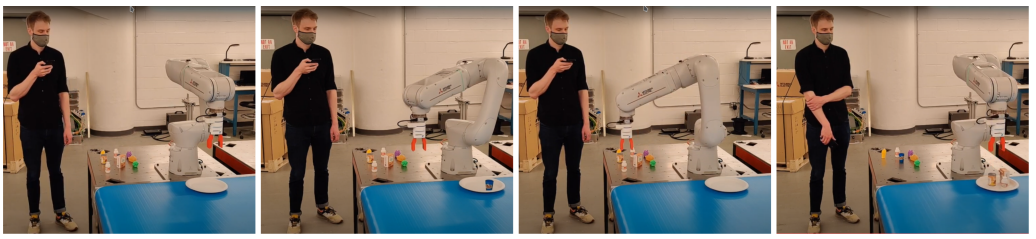
At the beginning of the interaction the partner teaches the robot three recipes that are used in the demonstration scenarios. These recipes are internally represented in DIARC as target world states to be achieved during the delivery action. These world states are of the form of a series of predicates that describe the state of a set of foods on a plate.

4.2.1 World State Not Affected. In this scenario (depicted in Figure 4) the robot receives an initial instruction from the partner, which it begins to execute. During execution the robot receives a corrective instruction. In this case the corrective instruction does not require the robot



(a) "Deliver a meal number two." (b) "Deliver a meal number two with fries instead of grapes." (c) Meal delivered

Fig. 4. Scenario 4: World state not affected by initial instruction.



(a) "Deliver a meal number two." (b) "Deliver a meal number one instead" (c) Undo state achieved (d) Meal number one delivered

Fig. 5. Scenario 5: World state affected by initial instruction.

to modify the state of the world to execute the new instruction, it only needs to change its task plan.

The first instruction that the robot receives is:

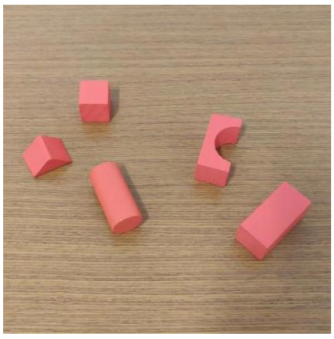
Operator: "Deliver a meal number two."

From this utterance, the parsing system relates "meal number two" to a state where peanut butter, orange juice, and grapes are all on the plate, and submits it as the current goal to the goal management system. To resolve this goal state, the goal manager begins to pick up the peanut butter and place it one the plate (Figure 4(b)). It is interrupted, however, by a correction:

Operator: "Deliver a meal number two with fries instead of grapes."

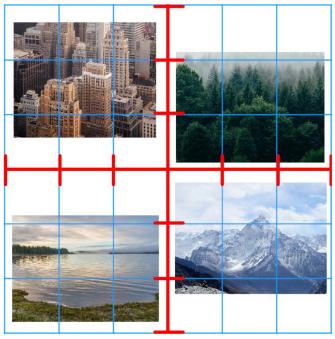
Of particular note in this utterance is "with fries instead of grapes," which signals to the parsing system a substitution in the known state associated with "meal number two." Throughout making progress on the initial goal, DIARC has continued to leverage known effects of actions to maintain an understanding of the current state of its environment. By comparing this new goal state to the known current state of the world, the agent finds that no modifications to the current world are necessary. However, the robot does need to generate a new task plan to achieve the newly specified goal state, which it performs (Figure 4(c)).

4.2.2 World State Affected. In this scenario (depicted in Figure 5) the initial instruction does produce effects that are different from the effects of the corrective instruction. To achieve the desired result the robot needs to undo the effects of the initial instruction before it executes the correct instruction.



(a) One of the images from the "blocks" task, where participants were asked to modify action given a new shape description.

(b) One of the images from the "tools" task, where participants were asked to modify action given a new object.



(c) Image from the "moves" task, where participants were asked to modify action given either a new relative movement or new absolute movement instruction.

Fig. 6. Example images as presented to participants.

To begin, the robot receives an instruction:

Operator: "Deliver a meal number three."

As with before, DIARC resolves this instruction to a goal state (in this case, a plate with soda, fries, and chicken). The robot executes the delivery action to achieve the desired state, the first step of which is picking up the soda and placing it on the plate (Figure 5(b)). However, the robot then receives the corrective instruction:

Operator: "Deliver a meal number one instead."

This meal is again resolved to a goal state. However, this new goal state is in conflict with the current state of the agent's environment. The new goal requires a plate of peaches, cereal, and milk, yet the current environment contains a plate with a soda. To complete this new goal, the conflicting state must first be resolved. DIARC constructs a new goal of not having the soda on the plate, and this goal is resolved (Figure 5(c)) before continuing with the new goal of resolving meal number one (Figure 5(d)).

5 ONLINE USER STUDY

Having demonstrated that our approach produces functional robot behavior, we need to ascertain that those robot behaviors match human performance and human expectations in the same settings.

5.1 Methodology

We constructed an online survey of 30 participants (fluent native English speakers living in the US, and approximately evenly distributed between self-identified male and female). The survey contained three task sections, in which participants were presented with an image (see Figure 6) and a set of text utterances instructing them to "take an action" in the depicted scenario. Participants were then asked to click anywhere on the image to indicate what they believed they were being instructed to do.

Our study took place over three tasks in three scenarios. In the "blocks" task, participants were asked to click on the block they were instructed to pass, with corrections focusing on descriptors (e.g., "the round one" or "the triangle one"). One such instruction read:

INSTRUCTOR: "Grab the square block."
INSTRUCTOR: "Actually, try the arch one."

Additionally, the blocks task contained eleven unique corrections (our pruned list from the previously cited [23] as discussed in Section 3.3). Participants saw 14 prompts in this condition.

In the "tools" task, we assessed how overall referents may be substituted. Participants saw a set of unique tools and were asked to select the appropriate object. Corrections intentionally used an under-specified action to avoid biasing the correction ("try" instead of "turn" or "hit," etc.), and the initial action is equally viable with the original object and corrected object (e.g., participants were not asked to "hammer" with the pliers). One such instruction read:

INSTRUCTOR: "Turn it with the socket."
INSTRUCTOR: "I mean, try the pliers."

Participants saw 9 prompts in this condition: 3 with no correction, and then 6 with our reduced set of corrective indicators: "actually," "sorry," "instead," "I mean," "no," and "oops."

The final task assessed how actions may be substituted by exploring how participants interpreted movement commands. A Cartesian plot with other distinct locations was shown, and participants were asked to start at (0,0) and click wherever they felt the instructions had them end up. We assessed absolute movements, with one such instruction reading:

INSTRUCTOR: "Go to the lake."
INSTRUCTOR: "Actually, go to the city."

We also assessed relative movements, which are more challenging, because they assume that a movement has taken place that must be undone. For example, one such instruction read:

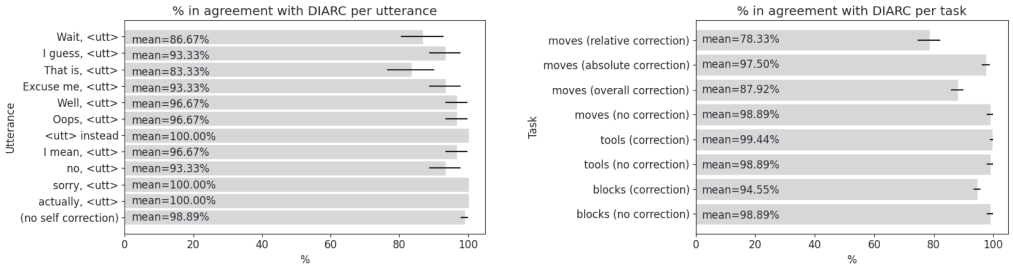
INSTRUCTOR: "Move down."
INSTRUCTOR: "Move up instead."

Participants saw 12 questions from this task (four without correction, four with relative correction, four with absolute correction).

The study began with a consent form, and within each task section questions were presented in random order between participants (to avoid any ordering effects) and concluded with an attention check question. There was a total of 36 evaluation questions. Response inputs allowed clicking anywhere on the image to avoid influencing participant responses, and all procedures were overseen by the Tufts University Social, Behavioral & Educational Research (SBER) Institutional Review Board to ensure ethical experiment deployment. The goal of the study was to determine how people would interpret instructions to modify behavior given descriptors, target objects, and behaviors.

5.2 Comparisons to Human Subject Data

The DIARC architecture was configured for a simulated environment. Rather than using a robot's sensor and effectors to interact with a real-world environment, DIARC was isolated from any interaction or perception. Instead, environment inputs were provided as a given to the agent as already-known states: DIARC was informed of the existence of the various blocks, tools, and locations without the distraction of any image processing. Similarly, the ability to interact with the environment was replaced with actions that did nothing, but were otherwise identical and allowed us to record the actions taken by the agent. This fully isolates all other factors and exclusively examines DIARC's ability to interpret instruction. This way the architecture was able to perceive all objects in the simulation as they were presented to our study participants, and the instruction interpretations of human participants can be compared to DIARC's interpretation.



(a) Percentage of participants who followed correction instructions for each corrective indicator in the "blocks" task (n=30). Error bars indicate standard error from the mean.

(b) Percentage of participants who took the same action DIARC performed across each task (n=30). Error bars indicate standard error from the mean.

Fig. 7. Summary of user study results.

All questions presented to the study participants are then presented, verbatim, to DIARC. After answering all questions, we compare the objects and actions proposed by DIARC to the objects and actions proposed by participants. Across all questions from all participants, 997 of the 1080 decisions made by participants matched decisions made by our proposed algorithms. We performed a 1-sample proportions test on these results (with continuity correction) and found that with 99% confidence subjects agree with DIARC in 9 to 9.4 of 10 cases.

Figure 7(a) shows the consistency for each of the employed correction strategies. Note that the "that is, <utt>" correction strategy, which performs the worst of the per-utterance cases, does not appear in the other two tasks. Regardless, the order of corrective indicators is randomized throughout all tasks to ensure no biases could occur as a result of the language used to indicate corrections.

The results show that participants overwhelmingly interpret explicit corrections in the same manner as DIARC in the blocks scenario ($\mu = 0.95$, $\sigma = 0.002$) and in the tools scenario ($\mu = 0.99$, $\sigma = 0.002$). In the movement scenario, we find that participants overwhelmingly match the interpretation of DIARC in the absolute movement cases ($\mu = 0.97$, $\sigma = 0.0005$), but less so in the relative movement cases ($\mu = 0.79$, $\sigma = 0.02$). This could be due to ambiguity introduced by the correction, though the number of deviant responses is too small and the responses themselves not systematic enough to determine why some subjects deviated from what we took to be the normatively correct response (as implemented in DIARC). We believe that the lower performace of DIARC in this case will not be a problem overall, as DIARC's interpretations still largely matched that of the participants.

6 DISCUSSION: LIMITATIONS AND FUTURE WORK

We have focused on the two distinct cases of corrections within teaching action scripts and cases where changes to the environment must be managed to address the new goal. However, a unique set of problems arise when these cases are merged: if during the action teaching sequence a correction occurs that requires undoing of actions, then what should be learned? Clearly the robot should not learn the behaviors that led to the incorrect state in the first place, and by the same token, should not bother to retain behaviors that do not contribute to the end goal. In many such scenarios the solution may be to make note of what actions have to be undone or have no effect, and remove these from the overall learned action. However, a yet more challenging problem would involve cases where the robot is unaware of some cause-and-effect, leading to incorrect removal or retaining of some actions. In these scenarios, it will become necessary to correct the correction (perhaps specified by the human partner through some form of dialogue).

Similarly, self-corrections may remain ambiguous after correction, because the substitution of one portion implies the substitution of another. For example, consider the motivating example from Section 4.1.3, in which the robot must "actually use the beta desk." We have seen that this produces a reasonable instruction for the agent to employ: `retrieveFrom(block, BetaDesk)` when starting with `retrieveFrom(block, AlphaDesk)`. However, there is still potential for ambiguity here: if the object on the `BetaDesk` is a `ball` instead of a `block`, this action cannot succeed because the referent object has not been updated alongside the referent location. The ambiguity here stems from the speaker's assumption that the listener will adapt their action given knowledge of the world: with one possible referent for the action, the agent should instead `retrieveFrom(ball, BetaDesk)`. A similar case arises when examining how referent objects may change the appropriate action. For example, if the robot is told to "use the hammer on the screw," and then "actually, use the screwdriver": the first instruction requires a striking action while the second requires a turning action.

In each of these cases, the agent needs a deeper understanding of how its environment may shape the appropriate behavior. However, this is not a trivial problem. Inferring that a referent should be substituted may be functionally valid but normatively unacceptable, and may also fail to resolve to a singular referent. It remains unclear how the appropriate referent can be selected here. Similarly, when a new action should be inferred, it is not clear which action should be inferred: prying and attaching can be accomplished by both a hammer and a screwdriver, so which strategy should the agent employ? The agent requires a deeper understanding of the task at hand, and again requires an understanding of what action is the normatively appropriate to select. Knowledge of affordances and task effectiveness may also be key factors in selecting the appropriate behavior.

Additionally, we have focused on using the human partner as the sole source of correction. However, the nature of this approach is such that the architecture is both monitoring the believed state of the world and capable of planning to obtain new states in the world. This potentially enables the system to correct itself. Some corrections will be straightforward: if a grasping action does not result in holding the block, the solution is to attempt a new grasp. However, other corrections may lead to more drastic plans involving compound actions, navigation into new spaces, and so on. This presents additional challenges in of itself, but in the human-robot interaction context also presents challenges in making sure that this autonomy occurs in a manner that is acceptable to the individuals occupying the same space as the robot.

Finally, our approach is bottlenecked by existing implementations and the current state-of-the-art in robot behavior implementations and natural language parsing, among other implementation details. Our approach of performing corrections at the task-level makes the key assumption that such a correction is possible (when in reality, a correction may fail because the proposed behavior is unsuitable). Unsurprisingly, in our testing we found that speech-to-text software does not achieve 100% accuracy, kinematic planners will occasionally fail, and path planning will occasionally fail to produce a feasible plan. While our approach works to address these by providing the capability for human intervention, and the state-of-the art in various robot behaviors continues to progress, a successful correction remains dependent on reliable behaviors.

In particular, existing natural language parsing techniques pose a challenge. We focused on the largely unambiguous explicit corrections, but many corrections are more rooted in uncertainty and context (e.g., "maybe a different one"). Similarly, we are also forced to assume that the parses provided to the system are correct and valid. Human self-corrections may compound in a way that is unclear, to a point where it may not be possible to resolve the intent without a dialogue taking place. There may also be cases where corrections are improperly parsed or interpreted, producing corrections that are not valid. It is additionally possible that the correction is not valid to begin with.

In these cases, the desired behavior of the agent may not be clear. One appropriate behavior might be to halt operation and allow a correction of the faulty behavior, while another would be to fail vocally (again, to allow correction of the faulty behavior). However, these each assume that a corrected instruction can be known to be faulty, and this will often not be the case. In yet-another challenging set of scenarios, parsing may be forced to consider ambiguous, yet synonymous language: even groups with the same language or cultural background may find many ways of referencing the same action or object, opening challenges for interpretation. While these are language parsing problems largely beyond our scope, and we argue our approach would continue to operate in these cases on the condition that the correction can be adequately detected, we acknowledge an area for improvement by leveraging future progress in the natural language processing space.

7 CONCLUSIONS

We have presented a novel approach to the problem of human uncertainty in human-robot interaction by providing a cognitive architecture with the capability to adapt to changing human needs. Specifically, we have provided a general approach to handling corrections during actions or teaching, both with and without changes to the environment. We demonstrated our methods through integration in a cognitive architecture with 5 on-robot use cases, as well as an additional 35 cases that we compared to results from a human subject experiment. Through the integration of the proposed methods into a cognitive robotic architecture, we were able to show how future robots can systematically and correctly handle human instructions with subsequent corrections in a way that meets human expectations.

REFERENCES

[1] Mattias Appelgren and Alex Lascarides. 2019. Learning plans by acquiring grounded linguistic meanings from corrections. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS'19)*. 1297–1305.

[2] M. Appelgren and A. Lascarides. 2020. Interactive task learning via embodied corrective feedback. In *Autonomous Agents and Multi-Agent Systems*, Vol. 34. https://doi.org/10.1007/s10458-020-09481-8

[3] Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, and Brenna Argall. 2017. Real-time natural language corrections for assistive robotic manipulators. *Int. J. Robot. Res.* 36, 5–7 (2017), 684–698.

[4] Alexander Broad, Jacob Arkin, Nathan Ratliff, Thomas Howard, Brenna Argall, and Distributed Correspondence Graph. 2016. Towards real-time natural language corrections for assistive robots. In *Proceedings of the RSS Workshop on Model Learning for Human-Robot Communication*.

[5] Arthur Bucker, Luis Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, and Rogerio Bonatti. 2022. LaTTe: Language trajectory TransformEr. Retrieved from https://arXiv:2208.02918

[6] Arthur Bucker, Luis Figueredo, Sami Haddadinl, Ashish Kapoor, Shuang Ma, and Rogerio Bonatti. 2022. Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'22)*. IEEE, 978–984.

[7] Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. 2010. Robust spoken instruction understanding for HRI. In *Proceedings of the Human-Robot Interaction Conference*. 275–282.

[8] Rehj Cantrell, Kartik Talamadupula, Paul Schermerhorn, J. Benton, Subbarao Kambhampati, and Matthias Scheutz. 2012. Tell me when and why to do it! Run-time planner model updates via natural language instruction. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI'12)*. ACM, New York, NY, 471–478. https://doi.org/10.1145/2157689.2157840

[9] Hugo Caselles-Dupré, Olivier Sigaud, and Mohamed Chetouani. 2022. Overcoming referential ambiguity in language-guided goal-conditioned reinforcement learning. Retrieved from https://arXiv:2209.12758

[10] Dongkyu Choi, Yeonsik Kang, Heonyoung Lim, and Bum-Jae You. 2009. Knowledge-based control of a humanoid robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 3949–3954.

[11] Dongkyu Choi and Pat Langley. 2018. Evolution of the ICARUS cognitive architecture. *Cogn. Syst. Res.* 48 (2018), 25–38.

[12] Hui-Qing Chong, Ah-Hwee Tan, and Gee-Wah Ng. 2007. Integrated cognitive architectures: A survey. *Artific. Intell. Rev.* 28 (2007), 103–130.

[13] Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. 2023. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. 93–101.
[14] Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu. 2019. Adapting translation models for transcript disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6351–6358.
[15] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. 2009. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'09)*. Kobe, Japan.
[16] Kathleen Eberhard, Hannele Nicholson, Sandra Kuebler, Susan Gundersen, and Matthias Scheutz. 2010. The Indiana cooperative remote search task (CReST) corpus. In *Proceedings of the Language Resources and Evaluation Conference (LREC'10)*.
[17] Jesse English and Sergei Nirenburg. 2020. OntoAgent: Implementing content-centric cognitive models. In *Proceedings of the Annual Conference on Advances in Cognitive Systems*.
[18] Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. 2021. Modeling and learning constraints for creative tool use. *Front. Robot. AI* 8 (2021). https://doi.org/10.3389/frobt.2021.674292
[19] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* 4, 1 (1997), 23–33.
[20] Tyler Frasca, Bradley Oosterveld, Meia Chita-Tegmark, and Matthias Scheutz. 2021. Enabling fast instruction-based modification of learned robot skills. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.
[21] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. 2019. Using natural language for reward shaping in reinforcement learning. Retrieved from https://arXiv:1903.02020
[22] Scott D. Hanford, Oranuj Janrathitikarn, and Lyle N. Long. 2009. Control of mobile robots using the soar cognitive architecture. *J. Aerospace Comput. Info. Commun.* 6, 2 (2009), 69–91.
[23] Peter A. Heeman and James F. Allen. 1999. Speech repairs, intonational phrases, and discourse markers: Modeling speakers' utterances in spoken dialogue. *Comput. Linguist.* 25, 4 (dec 1999), 527–571.
[24] Jörg Hoffmann. 2001. FF: The fast-forward planning system. *AI Mag.* 22, 3 (2001), 57–57.
[25] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. 2011. STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 4569–4574.
[26] James J. Kuffner and Steven M. LaValle. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the Millennium Conference: IEEE International Conference on Robotics and Automation (Cat. No. 00CH37065)*, Vol. 2. IEEE, 995–1001.
[27] John E. Laird. 2019. *The Soar Cognitive Architecture*. MIT Press.
[28] John Edwin Laird, Keegan R. Kinkade, Shiwali Mohan, and Joseph Z. Xu. 2012. Cognitive robotics using the Soar cognitive architecture. In *Proceedings of the AAAI workshop on Cognitive Robotics (CogRob@AAAI'12)*. Citeseer.
[29] John E. Laird, Paul S. Rosenbloom, and Allen Newell. 1986. Chunking in soar: The anatomy of a general learning mechanism. *Mach. Learn.* 1 (1986), 11–46.
[30] Pat Langley, Kathleen B. McKusick, John A. Allen, Wayne F. Iba, and Kevin Thompson. 1991. A design for the ICARUS architecture. *ACM Sigart Bull.* 2, 4 (1991), 104–109.
[31] K.-F. Lee, H.-W. Hon, and Raj Reddy. 1990. An overview of the SPHINX speech recognition system. *IEEE Trans. Acoust. Speech Signal Process.* 38, 1 (1990), 35–45.
[32] Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. 2001. Information states in a multi-modal dialogue system for human-robot conversation. In *Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog'01)*. Citeseer, 57–67.
[33] Oliver Lemon and Alexander Gruenstein. 2004. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Trans. Comput.-Hum. Interact.* 11, 3 (Sept. 2004), 241–267. https://doi.org/10.1145/1017494.1017496
[34] Willem J. M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition* 14, 1 (1983), 41–104. https://doi.org/10.1016/0010-0277(83)90026-4
[35] Robin J. Lickley. 1998. *HCRC Disfluency Coding Manual*. Human Communication Research Centre, University of Edinburgh.
[36] Dylan P. Losey and Marcia K. O'Malley. 2018. Including uncertainty when learning from human corrections. In *Proceedings of the 2nd Conference on Robot Learning (Proceedings of Machine Learning Research)*, Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (Eds.), Vol. 87. PMLR, 123–132. Retrieved from https://proceedings.mlr.press/v87/losey18a.html
[37] Paria Jamshid Lou and Mark Johnson. 2020. Improving disfluency detection by self-training a self-attentive model. Retrieved from https://arXiv:2004.05323

[38] Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. Retrieved from https://arXiv:1906.03926

[39] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. 2022. Interactive language: Talking to robots in real time. Retrieved from https://arXiv:2210.06407

[40] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2013. Learning to parse natural language commands to a robot control system. In *Proceedings of the 13th International Symposium on Experimental Robotics*. Springer, 403–415.

[41] H. Nicholson, K. Eberhard, and M. Scheutz. 2010. Um...I don't see any: The function of filled pauses and repairs. In *Proceedings of the 5th Workshop on Disfluency in Spontaneous Speech*. 89–92.

[42] Sergei Nirenburg and Peter Wood. 2017. Toward human-style learning in robots. In *Proceedings of the AAAI Fall Symposium on Natural Communication with Robots*.

[43] Jae Sung Park, Biao Jia, Mohit Bansal, and Dinesh Manocha. 2019. Efficient generation of motion plans from attribute-based natural language instructions using dynamic constraint mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA'19)*. IEEE, 6964–6971.

[44] Jordi-Ysard Puigbo, Albert Pumarola, Cecilio Angulo, and Ricardo Tellez. 2015. Using a cognitive architecture for general purpose service robot control. *Connect. Sci.* 27, 2 (2015), 105–117.

[45] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y. Ng et al. 2009. ROS: An open-source robot operating system. In *Proceedings of the ICRA Workshop on Open Source Software*, Vol. 3. Kobe, Japan, 5.

[46] Frank E. Ritter, Farnaz Tehranchi, and Jacob D. Oury. 2019. ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisc. Rev.: Cogn. Sci.* 10, 3 (2019), e1488.

[47] Frank Röder and Manfred Eppe. 2022. Language-conditioned reinforcement learning to solve misunderstandings with action corrections. Retrieved from https://arXiv:2211.10168

[48] P. Schermerhorn, J. Kramer, T. Brick, D. Anderson, A. Dingler, and M. Scheutz. 2006. Diarc: A testbed for natural human-robot interactions. In *In Proceedings of the AAAI Robot Workshop*. AAAI Press.

[49] Matthias Scheutz, Rehj Cantrell, and Paul Schermerhorn. 2011. Toward humanlike task-based dialogue processing for human robot interaction. *AI Mag.* 32, 4 (2011), 77–84.

[50] Matthias Scheutz, Jack Harris, and Paul Schermerhorn. 2013. Systematic integration of cognitive and robotic architectures. *Adv. Cogn. Syst.* 2 (2013), 277–296.

[51] Matthias Scheutz, Evan Krause, Brad Oosterveld, Tyler Frasca, and Robert Platt. 2017. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th International Conference on Autoomous Agents and Multiagent Systems*.

[52] Matthias Scheutz, Evan Krause, Brad Oosterveld, Tyler Frasca, and Robert Platt. 2017. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS'17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1378–1386.

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

[54] Matthias Scheutz, Thomas Williams, Evan Krause, Bradley Oosterveld, Vasanth Sarathy, and Tyler Frasca. 2019. An overview of the distributed integrated cognition affect and reflection DIARC architecture. *Cogn. Architect.* (2019), 165–193.

[55] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. 2022. Correcting robot plans with natural language feedback. Retrieved from https://arXiv:2204.05186

[56] Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL'14)*. 89–97.

[57] Elizabeth Shriberg, Rebecca Bates, and Andreas Stolcke. 1997. A prosody only decision-tree model for disfluency detection. In *Proceedings of the 5th European Conference on Speech Communication and Technology*.

[58] Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding English commands to reward functions. In *Proceedings of the Conference on Robotics: Science and Systems*.

[59] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annu. Rev. Control, Robot. Auton. Syst.* 3 (2020), 25–55.

[60] J. Gregory Trafton, Laura M. Hiatt, Anthony M. Harrison, Franklin P. Tamborello, Sangeet S. Khemlani, and Alan C. Schultz. 2013. ACT-R/E: An embodied cognitive architecture for human-robot interaction. *J. Hum.-Robot Interact.* 2, 1 (2013), 30–55.

[61] Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2785–2794.
[62] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. 2016. Fetch and freight: Standard platforms for service robot applications. In *Proceedings of the Workshop on Autonomous Mobile Service Robots*.
[63] Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional LSTM. Retrieved from https://arXiv:1604.03209

Received 15 June 2022; revised 8 July 2023; accepted 7 August 2023