Transforming Robot Programs Based on Social Context

David Porfirio,[1] **Allison Sauppé,**[2] **Aws Albarghouthi,**[1] **Bilge Mutlu**[1]
(1) University of Wisconsin–Madison, Madison, Wisconsin, USA
(2) University of Wisconsin–La Crosse, La Crosse, Wisconsin, USA
{dporfirio,aws,bilge}@cs.wisc.edu, asauppe@uwlax.edu

ABSTRACT
Social robots have varied effectiveness when interacting with humans in different interaction contexts. A robot programmed to escort individuals to a different location, for instance, may behave more appropriately in a crowded airport than a quiet library, or vice versa. To address these issues, we exploit ideas from program synthesis and propose an approach to transforming the structure of hand-crafted interaction programs that uses user-scored execution traces as input, in which end users score their paths through the interaction based on their experience. Additionally, our approach guarantees that transformations to a program will not violate task and social expectations that must be maintained across contexts. We evaluated our approach by adapting a robot program to both real-world and simulated contexts and found evidence that making informed edits to the robot's program improves user experience.

Author Keywords
Human-robot interaction, interaction adaptation, program repair, model checking

CCS Concepts
•**Human-centered computing** → **Systems and tools for interaction design;** •**Software and its engineering** → **Automatic programming;** *Model checking;*

INTRODUCTION
As consumer products, robots that are designed for human interaction will be introduced to a wide range of environments. For example, a delivery robot might be introduced to hotels, hospitals, or corporate offices; an information desk attendant robot might be used in doctors' offices, corporate buildings, or airports; and a security robot might be deployed in banks, parking structures, or shopping malls. Each application might be designed for a broad range of contexts, such as a generic delivery robot, rather than a specific one, or it might be designed for a particular context but brought into a different one. Both design scenarios result in poor fit between the application and the social context due to variations in physical properties, task characteristics, and user preferences across settings. For instance, a delivery robot will face different levels of traffic, available space for navigation, user profiles, and types of delivery requests between a hotel and a hospital, or between two hotel chains where it is deployed.

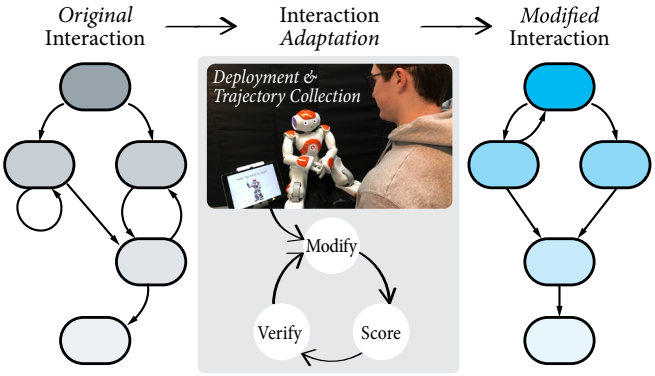


Figure 1. In this paper, we present a novel method that borrows from program synthesis to iteratively adapt robot programs to social context based on user input provided during an initial deployment stage.

Failure to design for contextual variability might result in ineffective and inappropriate user interactions. Prior research in human-robot interaction has shown that delivery robots programmed to repeatedly announce their presence until being acknowledged are not suited for workplaces with low interruptibility, such as in the emergency rooms of hospitals, but can otherwise be seen as being effective and a welcome addition to the social environment [18], and humorous robots that fail to consider an individual's sense of humor may be less amusing [34]. Furthermore, regardless of the context that a robot is placed in, the robot must adhere to various *correctness* guarantees that ensure that task expectations and universal social norms are met. For designers, creating interaction programs that are both *correct* with respect to these guarantees and *optimal* with respect to fitting a particular interaction context is challenging, often involving rigorous, iterative design processes and both domain and programming expertise.

How can robot developers close the gap between implementation and deployment or across deployments in robot programs due to changes in context, while ensuring that their applications achieve design goals across contexts? In existing approaches to designing for multiple contexts, the robot will automatically adapt its low-level behaviors to its users and environments. We argue that in addition to modifying low-level

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
https://doi.org/10.1145/3313831.3376355

behaviors, incorporating contextual knowledge into a program can be achieved by modifying the *structure*, and thus control flow, of the program itself in order to improve a robot's ability to make decisions within different contexts, such as deciding what to do if the human ignores the robot, eliminating useless behaviors or adding useful ones, and controlling the overall pace of the interaction. In existing approaches to adaptation, however, making heavy changes to the structure of a program may sacrifice its adherence to context-free task expectations and social norms, or its *correctness*.

We propose a lightweight approach informed by program synthesis and formal verification to transforming the structure of complex human-robot interaction designs based on social context. In our approach, depicted in Figure 1, the correctness of programs is maintained as they are transformed over the course of multiple cycles, or *epochs*, of users interacting with a robot. In Step 1, on any given epoch, we record example execution *traces* of end users interacting with the robot as paths through the interaction program. Upon completion of a trace, the user is asked to score the trace based on the quality of their experience. Rather than assigning values to individual states based on individual robot decisions, entire sequences of robot decisions are scored. Thus, our approach complements existing methods of adapting interactions to social contexts that optimize a robot's low-level behaviors within a static interaction program (e.g., [33, 17, 8]). Additionally, collecting a single score provided by the user instead of passively sensing user experience makes our approach easy to deploy on any robot and mitigates many of the privacy concerns that arise from video and audio data collection in public settings.

In Step 2, we perform automatic *program repair* by searching for an edited, or *transformed*, interaction that maximizes acceptance of positive traces and exclusion of negative traces within the interaction. During the search, we employ *model checking* [9] to ensure that the transformed program adheres to a baseline set of social norm and task expectation properties expressed in temporal logic [20], thus maintaining quality and functionality guarantees. In Step 3, e.g., on the next day, additional traces are collected and further edits to the interaction can be made. Our method to transforming an interaction presents a number of challenges: (1) a *large search space*, or the large space of possible transformations; (2) *sparse input*, or a small number of examples with limited labelling to indicate quality that we must use to inform edits; and (3) the need to carefully perform edits to always result in functional, high-quality interactions. Over time, however, our approach creates interactions of potentially better quality than their corresponding original interactions while adhering to guarantees for not violating basic social norms and task expectations.

In addition to our technical approach and its implementation, we present the design of and findings from two studies: an *online study* that assessed whether interaction designs that are adapted by our approach are seen as improvements by third-party evaluators and a *field study* that tested how user evaluations of an adaptive information desk attendant robot changed over the course of 12 days. The online study utilized a novel method, called *interaction vignettes*, that represents interactions at a high level of abstraction, in the form of play scripts, to enable third-party evaluators to assess the appropriateness of an interaction for a given context. Our field study assessed the effectiveness of the transformations performed by our approach in improving user experience and contextual fit in a real-world setting over multiple epochs.

Our contributions include:

- A novel *method* for transforming human-robot interaction programs to physical and social contexts based on sparse input from humans interacting with the robot (§3);
- An open-source *implementation* of our adaptation approach[1] as a program synthesis and repair algorithm (§3);
- A two-part *empirical evaluation* of our approach, which includes (1) a validation of the potential of our approach under the controlled conditions of an online user study, and (2) an evaluation of the effectiveness of our approach in real-world settings through a field experiment (§4).

RELATED WORK

Human-Robot Interaction Design—Designing interactions as state-transition systems is a familiar concept within HRI, used in visual programming environments (VPE's) such as Interaction Composer [6], RoboStudio [5], Choregraphe [21], Interaction Blocks [26], and RoVer [20]. In many of these VPE's, the states of an interaction represent subroutines involving a behavior or set of behaviors performed by the robot, and a response from the user. In *RoVer*, these subroutines are called *microinteractions*. Despite the potential to create robust interaction designs using a VPE, interactions will potentially fail to generalize to different contexts due to factors such as culture [14, 15, 31], education [16], humor and user preferences [34], and environmental factors [18].

Interaction Adaptation—Adapting intelligent agents, including robots, to social context is a well-studied problem. Dialog systems, for instance, employ various techniques such as reinforcement learning [29] or memory models [23] to adapt to the needs and preferences of users. Within social robotics, much of the existing work on adaptive systems involves modifying the behaviors of a robot to improve extended interactions with a single user [1, 32]. For instance, reinforcement learning was used to adapt a humorous robot to the different humor levels of an audience [34], as well as the linguistic style of a robot to suit the preferences of those with whom it interacts [22]. Prior work on adaptive robots meant to have short-term interactions with multiple human partners includes a bartending robot that uses reinforcement learning to select its responses to customers [13] and autonomous vehicles that use inverse reinforcement learning to adapt their driving to the behaviors of other vehicles on a road [24]. Work on adaptive robotic platforms extends to collaboration as well, where similar to our approach to adapting interactions, a robot deployed in the field adapts its execution of a manipulation task based on training labels provided by observers [28]. While reinforcement learning and inverse reinforcement learning are effective

[1] An open-source implementation of our technical approach is available at https://github.com/Wisc-HCI/interaction-transformation

in assigning reward to state, we propose assigning values to whole paths based on minimal input from the human.

Program Synthesis and Repair—To perform interaction adaptation, we build on the ideas of *automatic program repair and synthesis* from the software engineering and formal methods communities. Given a logical specification of a program and a set of allowable modifications, automated repair involves uncovering a modified program that satisfies the specification [10]. The primary distinction between our method and existing repair techniques like that of [10] is that our repair objective contains both weighted trajectories (examples) as well as temporal logic formulas. Thus, *reactive synthesis* algorithms and tools do not apply in our setting.

Our method for interaction adaptation is based on a version of the *counterexample-guided inductive synthesis* loop [30], in which an adapted interaction is iteratively synthesized to exclude undesirable behavior. During an iteration of synthesis, some possible ways to perform the automatic creation of a new system are through enumerative searches [19] or Markov Chain Monte Carlo (MCMC) [27, 7]. In the MCMC approach, the acceptance criteria for a modification is based on both the change in performance of the program after the edit is made and the change in correctness according to the program specification. Other work in program synthesis through genetic programming also uses the correctness of the program to create a fitness function that guides modification [11, 12, 35]. In the genetic programming approach, modifications to the program are first proposed and then either accepted or rejected based on the results from *model checking* [2] or testing, in which program *properties* are checked for violation. We also use model checking to accept or reject proposed modifications.

Our approach of modifying the transition system is also reminiscent of techniques for repairing probabilistic systems, which modify transition probabilities in order to satisfy a probabilistic property of the system [4, 3].

TECHNICAL APPROACH

Our goal is to transform an interaction program to fit a particular context, while ensuring that it satisfies a set of baseline properties. In this section, we (1) formalize the interaction adaptation problem and (2) present an algorithm for solving it inspired by work on automated program repair and synthesis.

Preliminaries: Interactions and Traces

We will formalize an interaction program I as a state transition system, where states are represented by behaviors that the robot may perform, and actions are represented by the responses that the robot can recognize from the human. For instance, the set of responses may be $\{Ready, Ignore\}$, which correspond to whether the human has acknowledged the robot at the end of the behavior or is ignoring the robot. As with prior work, states may be represented by *microinteractions* [20]—lower-level modularized interaction transition systems—effectively allowing us to incorporate proceduralization into our representation of interactions.

Formally, an interaction I is a tuple $(S, Act, \rightarrow, s_0)$, where S is the set of states within the system; Act is the set of allowable

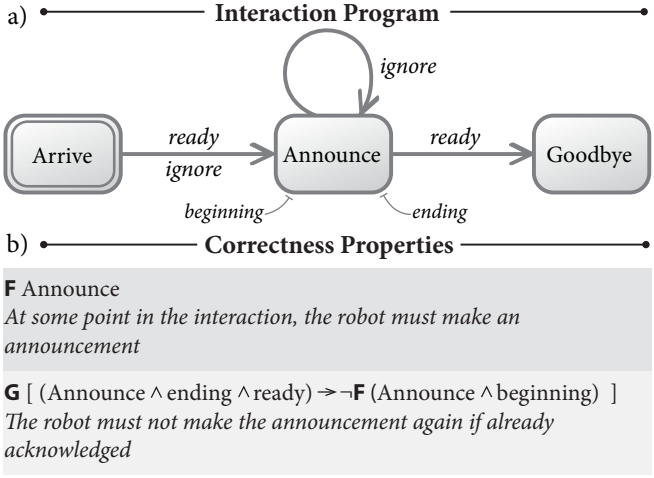


Figure 2. An interaction transition system: a robot arrives at a location and makes an announcement. Transition labels represent human actions. Correctness properties describe task-related rules and social norms to which the robot must adhere.

(expected) human actions; $\rightarrow \subseteq S \times Act \times S$ is the transition relation, representing how the state of the robot changes in response to human actions; and $s_0 \in S$ is the initial state. We denote states with no outgoing transitions as accepting (sink) states. In practice, states may have additional labels, which we omit here for brevity. The representation of interaction I in Figure 2 is an example of a transition system, where the initial state is *Arrive* and the accepting state is *Goodbye*.

A trace t through interaction I is a sequence of states and actions starting in s_0 and ending in an accepting state. We represent a trace as $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-2}} s_{n-1}$. Within the interaction in Figure 2, an example trace is Arrive $\xrightarrow{ready}$ Announce $\xrightarrow{ready}$ Goodbye. Intuitively, a trace is an execution of the interaction program exercised by a human and the robot.

The Adaptation Problem

After having observed a set of traces T, defining multiple human–robot interactions, we aim to transform, or *repair*, the interaction so as to improve its quality. At the end of recording a particular trace t, the human will be asked to rate the quality of the trace, which will be converted to a score in the real-valued interval $[-1, 1]$ and stored with the trace. Given trace t, we will use $w(t)$ to indicate its score.

The adaptation problem can be stated as the following optimization problem, where our goal is to construct a repaired interaction $R(I)$ that aims to eliminate low-scoring traces.

> Find a repair $R(I)$ that (1) maximizes $\sum_{t \in T \cap R(I)} w(t)$ and (2) satisfies a set of baseline correctness properties.

Here $T \cap R(I)$ denotes all traces in T that are accepted by $R(I)$. Therefore, the optimization problem aims to eliminate negatively scored traces and maintain positively scored ones.

We now define what we mean by baseline correctness properties. These are hard constraints added to ensure that the interaction does not break down after a repair, e.g., not fulfill

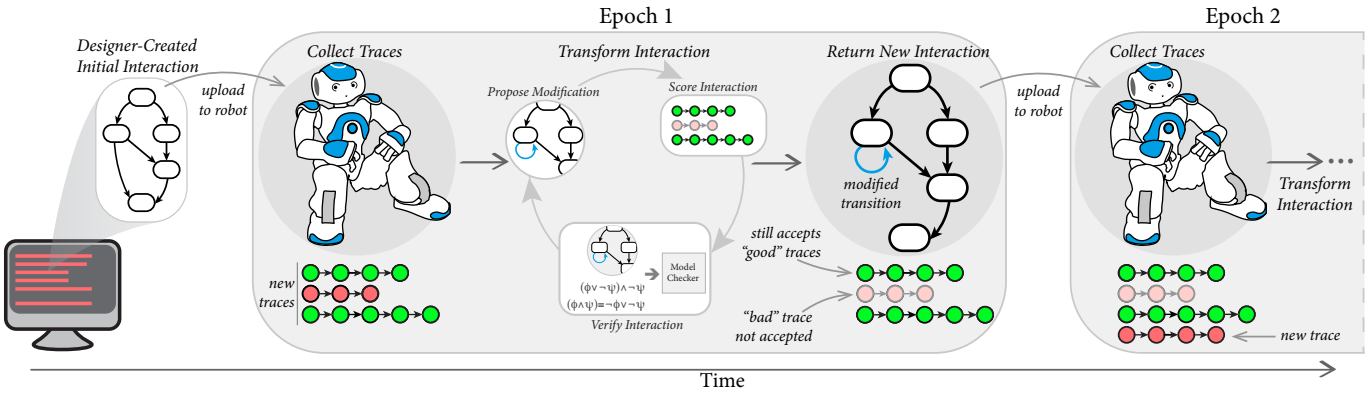


Figure 3. Overview of our adaptation process. Given an interaction model and set of baseline correctness properties, each round of adaptation records execution traces and searches for repairs until an interaction that increases human experience and adheres to the baseline properties is found.

its intended task or violate social norms. Our approach is inspired by the work on RoVer [20], where correctness properties are encoded in logic and checked with an automated prover, namely, a *model checker* [2]. Figure 2(b) shows two properties in linear temporal logic (LTL) along with their description. We refer the reader to Baier et al.'s [2] book for a formal introduction to LTL and Rover [20] for more examples of interaction properties encoded in LTL. For our purposes here, the English-language description will suffice. For example, in Figure 2, the property **G** [(Announce $\wedge$ ending $\wedge$ ready) $\rightarrow$ $\neg$**F** (Announce $\wedge$ beginning)] specifies that if the robot is ending its announcement and has been acknowledged by the human, then it will not initiate future announcements.

The Adaptation Algorithm

Our approach to adaptation, illustrated in Figure 3, is segmented into two distinct components: (1) trace collection and (2) solving the adaptation problem, which are iterated upon indefinitely or until stopping criteria is met. We refer to a cycle of trace collection and repair as an *epoch*. Below, we describe the components of an epoch and process for iterating epochs.

Collecting traces

In this step of an epoch, the interaction is deployed in a robot, and the robot interacts with humans. For each human that interacts with it, the robot will record traces as the sequence of n steps through the interaction $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-2}} s_{n-1}$. To ensure that traces with sufficient variability are collected, the robot may also generate a sequence of steps at runtime that satisfy the baseline properties but are not accepted by the interaction. For instance, if a particular state does not exist in the interaction, the robot may artificially include that state in the trace experienced by a user. We refer to the process of generating traces not accepted by the interaction as *mutation*.

Solving the Repair Problem

In this step of an epoch, we solve the adaptation problem based on the traces we have collected. The difficulty here is that the optimization problem results in a combinatorial explosion of possible modifications to I, and, to make matters worse, we have to ensure that the new, repaired interaction $R(I)$ satisfies correctness properties—an expensive process involving a model checker.

We define a possible repair $R(I)$ as a sequence of *edits* that manipulate the states and transitions of I. Some examples of edits may include, but are not limited to, swapping the robot's behavior within a particular state, redirecting a transition, adding a new state, and removing an added state.

Our repair algorithm searches for a repair by applying a sequence of edits $e_1, \ldots, e_n$ to the original interaction I. Specifically, it searches the space of edits in a breadth-first fashion, from the shortest to the longest (up to a fixed maximum length). For every sequence of edits, $e_1, \ldots, e_n$, it (1) constructs a new interaction $R(I)$, (2) computes the score $\sum_{t \in T \cap R(I)} w(t)$, and (3) checks if $R(I)$ satisfies the baseline properties using a model checker. To avoid weighting frequently-seen traces more heavily, we consolidate and average the scores for duplicate traces into a single unique trace. Finally, the algorithm returns the repaired interaction $R(I)$ that has the highest score and satisfies the correctness properties.

One outcome of edits is the removal of negatively-rated, or *bad*, traces from being accepted by the interaction while attempting to maintain acceptance of positively-rated, or good, traces. Through the removal of bad traces, possibly unrated traces in the interaction are introduced as a side-effect. For instance, in redirecting a transition to remove a bad trace, a new path through the interaction may be introduced that would not be accepted by the starting interaction. Mutations are another mechanism through which potentially unrated traces can be introduced. If a mutated trace is rated positively, then edits may be made to include it in the repaired interaction.

Iteration of Epochs

After obtaining an optimized interaction with a highest score, the resulting interaction will serve as input to a subsequent iteration of adaptation. The modified interaction will be deployed into a robot, and a new set of traces will be collected that will be appended to the set of previously collected traces. Thus, potentially unrated traces introduced in previous epochs may be experienced by users and rated positively, neutrally, or negatively. These new traces will be scored and used as input to the optimization algorithm. If at any point no further iterations of adaptation are desired, the interaction will be returned and the adaptation process will halt. Otherwise, the adaptation cycle can continue indefinitely.

Key Algorithm Optimizations

We now describe two key optimizations that make our algorithm's implementation practical.

Reducing Model Checker Calls

As the depth of search increases, the number of interactions that need their score calculated and correctness verified increases exponentially. A reader who is familiar with model checking will note that frequent calls to the model checker can significantly slow down the search. Therefore, in each call to the model checker, we collect *counterexamples* that show paths through the proposed interaction that lead to property violations. Before invoking the model checker next time, we first check whether the proposed interaction includes one of the discovered counterexamples, in which case calling the model checker is unneeded. Thus, the model checker is only called if the program is deemed correct with respect to the set of counterexamples. This idea of using counterexamples to avoid calling an expensive verifier has appeared in various guises in the program synthesis literature [7].

Potential and Fault Localization

To reduce the search space of edits, we perform *potential* and *fault* localization, a process that involves searching user traces for transitions and states that, if changed, would most likely increase human experience. In our approach, we perform *fault* localization by ranking transitions and states based on their frequency in bad traces, and *potential* localization by ranking transitions and states based on whether, if changed, they would match a transition or state that is frequently observed in good traces. After a ranking is assembled, we determine an "editable" set of transitions and states that is some percentage of the size of the interaction. For instance, fault and potential localization of 30% would search for the 30% of transitions and states within the whole interaction that, if changed, would most likely increase the score of the interaction.

EVALUATION

To evaluate our approach to transforming human-robot interactions, we employed an *online study* conducted using the Amazon Mechanical Turk (MTurk) marketplace and a *field study* involving a real-world, long-term deployment of two robots at a university building accessible to the public. The purpose of the MTurk study was to test the efficacy of our approach under optimal interaction conditions—a high number of users and ratings per individual trace, and mutations that require few edits to be accepted in the interaction. Our evaluation utilized a novel evaluation method, called *interaction vignettes*, that asked MTurk users (Turkers) to rate conversational transcripts of simulated interactions. With Turkers, the interaction context shifts to include their preferences for how the robot should act. The field study aimed to demonstrate the efficacy of our approach in real-world settings by adapting a human-robot interaction program to a specific context over the course of 12 days. Using the data collected from the field study, we additionally performed a qualitative analysis of user emotions and attitudes towards the robot and present several themes resulting from this analysis that inform future iterations of our approach to transforming interaction programs.

In both studies, we test our hypothesis that transforming interactions through our approach will improve the quality of end-users' experience when interacting with the robot (H1). We also test our hypothesis that our approach will improve the contextual fit of the interaction, or users' perception of how effective the robot is at interacting with people within the context that it is placed in (H2). We will first describe the human-robot interaction program and task criteria that were used in each study, followed by the results from the online study and the results from our field study.

Starting Interaction

Both studies focused on a robot who sits at a welcome desk in a large research institution and whose job is to answer questions about both the building and itself. Figure 4 (top-left) describes all potentially necessary behaviors for the robot, and Figure 4 (top-right) describes the responses from the human that the robot can recognize. Using this interaction space, we designed an initial interaction by hand containing seven states within which are assigned the robot behaviors *Greeting*, *ListOut*, *Prompt*, *Validate*, *Answer*, *ReferToDesk*, and *Farewell*. Beginning at the state with the *Greeting* behavior, the state transitions of the initial interaction are defined below:

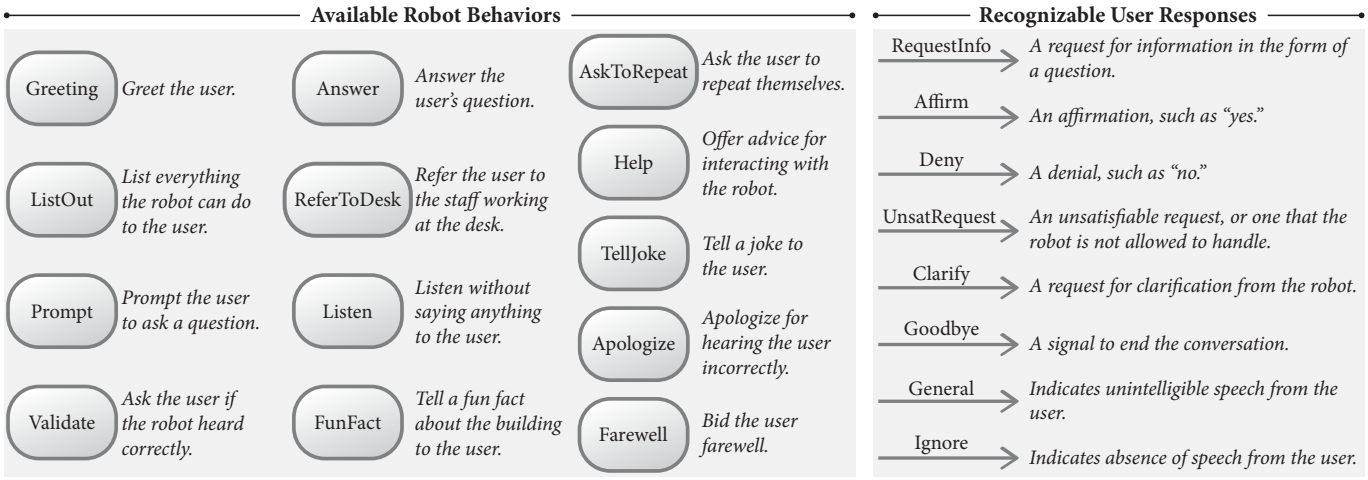
1. *Goodbye* always leads to the state with *Farewell*.
2. *UnsatRequest* always leads to the state with *ReferToDesk*.
3. *RequestInfo* always leads to the state with *Validate*.
4. *AskClarify* or *General* statements always lead to the state with *ListOut*.
5. *Ignore* always leads to the state with *Prompt*.
6. If in the state with the *Validate* behavior, an *Affirm* leads to the state with the *Answer* behavior. Otherwise, *Affirm* leads to the state with the *ListOut* behavior.
7. If in the state with the *Validate* behavior, a *Deny* leads to the state with the *Prompt* behavior. Otherwise, *Deny* leads to the state with the *ListOut* behavior.

In the starting design, we introduced various structural particularities intended to allow room for the interaction to improve. Specifically, the robot overuses *ListOut* to constantly remind users of the questions it can answer. Additionally, when ignored, the robot "demands attention" [18] with the *Prompt* behavior. Lastly, the starting interaction intentionally excluded potentially entertaining behaviors such as *TellJoke*.

Our starting design is also subject to the baseline properties shown in Figure 4 (bottom). We refer to one of the properties as a *job requirement*, or a rule required by the staff members working at the welcome desk. The remaining properties are rules that the robot should adhere to for the interaction to make sense. Subject to these properties, we allow edits to be made that (a) swap the robot's behavior within a particular state, and (b) redirect transitions to other states.

Online User Study

Our first evaluation assessed whether interactions adapted using our approach better fit their context and improved user



Correctness Properties

LTL Form	*Property Description*
G *UnsatRequest* → **X** ReferToDesk	***Job Requirement**: If the human makes an unsatisfiable request, the robot must refer them to the desk.*
G *Goodbye* → **X** Farewell	*Whenever the human says goodbye, the robot must respond by saying farewell.*
F Farewell	*The robot must say farewell at some point.*
G Farewell → **X** END	*The interaction must end after the robot says farewell.*
¬(AnswerQuestion | Validate) **U** *RequestInfo*	*The robot cannot answer or clarify a question unless the human asks a question.*
G (*Affirm* ∧ Validate) → **X** AnswerQuestion	*The robot must answer the most recently asked question if the human affirms that the robot heard correctly.*
G *RequestInfo* → **X** (AnswerQuestion | Validate)	*When the human asks a question, the robot must respond by answering or clarifying the question.*

Figure 4. (top-left) The robot behaviors that are available in our interaction space. (top-right) The user responses that the robot can recognize within our interaction space. (bottom) The correctness properties in LTL that interactions in our interaction space are subject to.

experience. Below, we describe our evaluation approach, study design, and findings.

Interaction Vignettes

To test our approach with a large number of users in a short amount of time, we devised a novel method that involves generating *interaction vignettes* in the form of play scripts, describing the speech of all parties in a hypothetical interaction. The generated vignettes represent interactions at a level of abstraction that enables the judgment of the appropriateness of the interaction for a given context and lacks details that might mask the high-level behaviors that our approach targets and that would be costly to design. These vignettes were probabilistically generated, including the content of the human-user speech, from observations in the field about what users are most likely to do in response to specific behaviors. A sample vignette is provided in Figure 5.

```
Robot: Hey there!
Human: Where can I eat around here?
Robot: Did you want to know the dining options in
       the building?
Human: Yes, please.
Robot: If you're looking for lunch, try the
       mediterranean restaurant on the far-east
       side of the building! There is also a
       coffee shop right behind me.
Human: Okay, thank you.
Robot: See you later!
Human: (leaves interaction)
```

Figure 5. A sample *interaction vignette* between a human and a robot, similar to what Turkers saw in our online user study.

Study Design

To establish the effectiveness of our approach in improving the appropriateness of interactions for a given context, we designed a 2×1 within-participants study that manipulated whether interactions were *adapted* or *non-adapted* and that measured the *user experience* and *fit* of an interaction to its context as the dependent variables.

Measures

Our subjective measures aimed to measure how participants perceived the behaviors of a robot, as presented in an interaction vignette, given the context and included eight items. The first item measures the *score* of the interaction from the perspective of a third-party observer using a three-point Likert scale with the question "How did the interaction go?" The score of a vignette is meant to indicate the quality of the trace that generated the vignette, but is not meant to measure user experience or contextual fit. Subsequently, a seven-point Likert scale was used for four items measuring the perceived user experience of the human in the vignette (Cronbach's $\alpha = 0.92$) and three items measuring the perceived contextual fit of the interaction (Cronbach's $\alpha = 0.85$). The items measuring user experience focused on the effectiveness of the responses and the flexibility of the robot, as well as the naturalness of the interaction and perceived user satisfaction. The items measuring contextual fit focused on the appropriateness and goal-

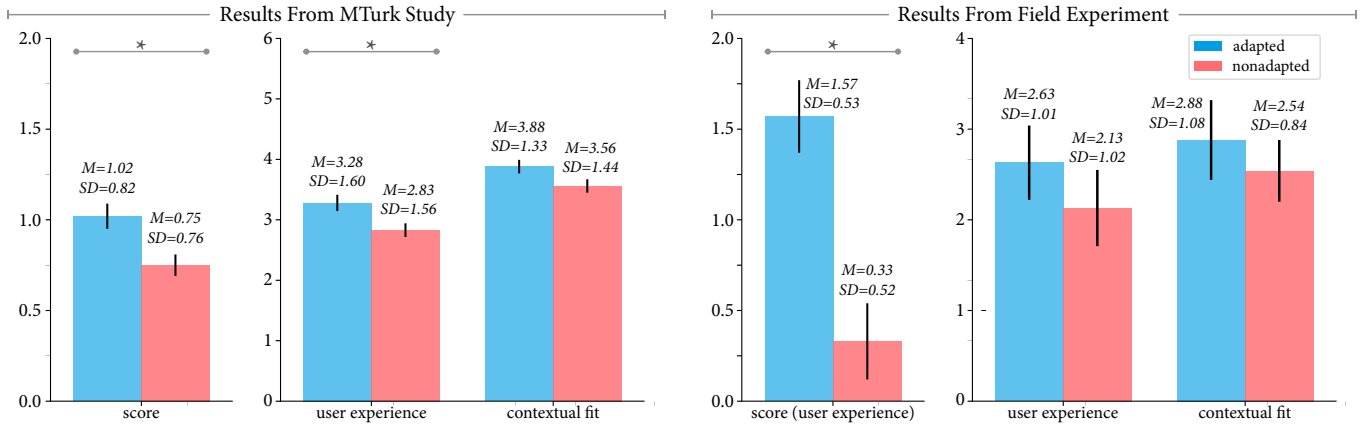


Figure 6. Interaction score, user experience, and contextual fit from the MTurk (left) and field (right) studies. "*" denotes statistically significant results.

supporting ability of the robot, and whether it was suited to its surrounding environment. We randomized the order in which the items measuring user experience and fit appeared.

Procedure

Our online study involved two phases. In the first phase, we created and deployed 100 interaction vignettes from the starting interaction onto MTurk. Half of these vignettes were mutated by treating non-mutated traces as templates. Mutated traces were identical to their non-mutated counterparts except for one or two changes entailing the insertion or deletion of a state or the modification of an existing state or transition.

After the completion of the first phase, we used the scores for each trace provided by Turkers to transform the interaction program with fault/potential localization set to 30%. We performed a second BFS search starting from the interaction program returned from the first search, simulating the completion of two epochs in the first phase of our user study. In the second phase of our study, we created and deployed another 50 interaction vignettes from the updated interaction, and 50 from the starting interaction, onto MTurk.

The study followed a protocol that was reviewed and approved by our Institutional Review Board (IRB). Participants first reviewed and signed a consent from. They were then provided with a description of the interaction context and information on the task. Next, participants reviewed six *interaction vignettes* and responded to a questionnaire for each vignette that included our subjective measures. They received compensation upon approval of their work.

Participants

We recruited 55 participants from the MTurk online marketplace for each phase of our study, for a total of 110 participants. We discarded data from seven of these participants because either data was missing from their responses, or because they failed our "attention check" question, in which we asked participants to answer a question in a specific way. The primary language of each participant was English. The participants were paid $6 USD/hour, and each participant received $1.25 USD for participating for approximately 10–15 minutes.

Results

Transformations—The final interaction accepted a set of good and neutral traces with four mutations: (1) from *Greeting* the *Ignore* response will cause the robot to transition to the state with *ListOut*, (2) from *Validate* the *General* response will cause the robot transition to the state with *Answer*, (3) from *Validate* the *Ignore* response will also cause the robot to transition to the state with *Answer*, and (4) from *Answer* the *Affirm* response will cause the robot to transition to the state with *Farewell*. In removing bad traces, the final interaction accepted an edit in which from *Greeting* the *Affirm* response will cause the robot to transition to the state with *Prompt*.

Interaction Scores, User Experience, & Contextual Fit—We tested our results from the second phase of our MTurk study for significance with a repeated measures analysis of variance (ANOVA). The interaction score for the transformed interaction ($M = 1.02$, $SD = 0.82$) was significantly higher than the score provided for the starting interaction ($M = 0.75$, $SD = 0.76$) ($F(1,46.79) = 7.655$, $p < 0.01$). Perceived user experience for the transformed interaction ($M = 3.28$, $SD = 1.60$) was also significantly higher than the perceived user experience of the starting interaction ($M = 2.83$, $SD = 1.56$) ($F(1,45.27) = 7.5346$, $p < 0.01$). Lastly, we did not find a significant difference in perceived contextual fit between the transformed interaction ($M = 3.88$, $SD = 1.33$) and the starting interaction ($M = 3.56$, $SD = 1.44$).

Field Study

Study Context and Implementation

Our field study took place in a research institute located on the campus of a large university. The building occupants include researchers and research support staff, but the ground floor includes a more diverse population that is open to the public and features public events and dining options. The ground floor also contains a welcome desk for assisting visitors.

The setup for our field study is shown in Figure 7. We deployed two Softbank Robotics Nao robots[2] on either side of

[2]https://www.softbankrobotics.com/us/NAO

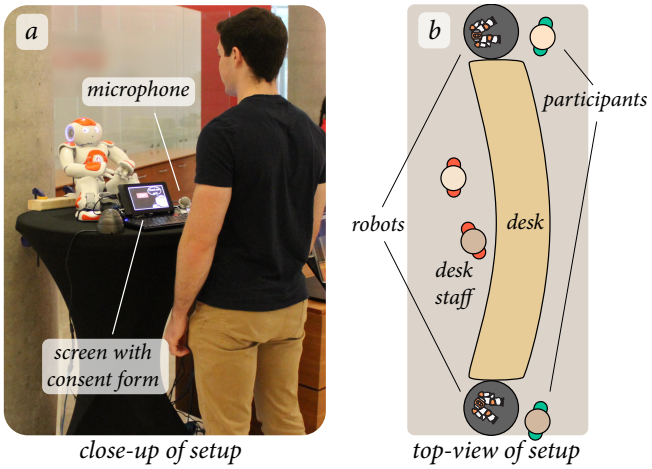


Figure 7. Our study setup. (a) Each robot was placed on a cocktail table and equipped with an external microphone and a touch-screen computer for participant consent and input. (b) The two robots were placed on the two sides of an information desk in a research institute building.

the welcome desk. One robot ran the initial interaction program throughout the study, while the other ran an interaction program that had been adapted from the traces collected thus far on both robots. At the start of the experiment, both robots ran the initial interaction program. On each day of the study, we randomized the placement of the robot running the adapted interaction to minimize bias resulting from robot location.

Measures

We measured user experience and contextual fit with a nine-item questionnaire presented to users at the end of their interaction with the robot. The first item measured the users' interaction experience as the *score* of an interaction trace and uses a three-point Likert scale with the question "How was your experience?" Thus, unlike the MTurk study, the quality of the trace is tied to user experience. Subsequently, we used a five-point Likert scale for four items measuring the users experience in greater depth (Cronbach's $\alpha = 0.90$) and four items measuring the perceived contextual fit of the interaction (Cronbach's $\alpha = 0.83$). The items were similar to those in the MTurk study, with an extra item for contextual fit for whether the robot aligned with user preferences. We randomized the order in which the five-point Likert scale items were presented.

We analyzed the video recordings following a modified *emotion coding* process [25], in which we coded the videos for participant facial expressions and speech utterances that indicated basic emotions or attitudes toward the robot. For each participant, we noted the timestamps at which facial expressions or speech utterances occurred that differed from a "neutral" state, extracting a coding scheme comprised of a set of in-vivo codes from the timestamps. To determine reliability, a secondary coder trained on the coding scheme was asked to code 15% of the timestamps (Cohen's $\kappa = 0.78$).

Procedure

We ran our study between 8 o'clock am and 5:30 pm over the course of twelve days. The study setup was fully self-run in that passersby interested in interacting with a robot could initiate the study themselves. The touch-screen computer placed in front of the robot (shown in Figure 7) guided participants through providing informed consent. Upon providing consent, the robot initiated the interaction. At any point during the study, participants could terminate the interaction prematurely by pressing a button on the screen. After the interaction ended either by the button-press or through an exchange of farewells by the participant or robot, the participant was guided through the nine-item questionnaire on the touch-screen computer. Mutations were made at runtime for under one quarter of users and were more drastic than in the MTurk study, ranging from inserting, deleting, and modifying states and redirecting transitions to generating multiple steps in a trace at random.

After a day of collecting data, we performed a union of the set of traces collected on that day with the set of existing traces collected on previous days to create a newly combined set of traces. The new set of traces then served as input into the transformation algorithm, which was run overnight for between nine and twelve hours. In order to promote hill-climbing, we parallelized the algorithm such that one core ran the algorithm with the most up-to-date interaction as input, while another core ran the algorithm with the original interaction as input. Thus, edits made on a previous day could easily be undone if the core running the algorithm with the original interaction as input returned the better interaction.

Participants

A total of 70 participants (69 unique individuals, and 1 individual who interacted with the robot twice) provided informed consent to interact with the robots over the course of 12 days (5 epochs). We discarded 31 participants because they fell into one of the following three categories: (1) the video showed or there was a strong reason to believe that a minor interacted with the robot or was present during an interaction; (2) the video showed a separate party other than the person who consented to participate assuming responsibility for answering some or all questions on the questionnaire; and (3) the participant did not answer any questions on the questionnaire. 26 discarded data points were observed on the day of a nearby children's event during which many minors interacted with the robot. Discards were much less frequent on other days of the study. Of the 39 participants remaining, all successfully completed and scored the interaction. Of these participants, three did not provide their gender, and five did not provide their age. Of the participants who provided this information, 22 were male and 14 were female, and the average age of participants was 30.1 ($SD = 13.3$), ranging from 18 to 61. Lastly, the video equipment failed with two participants, restricting our video analysis to the remaining 37 participants.

Results

Transformations—Edits to the interaction occurred without accepting any mutated traces. Beginning at the starting interaction, we observed one edit made in the first epoch. In the second epoch, the previous edit was undone and three new edits were introduced. In the third epoch, the changes from the previous epoch were undone and two new edits were introduced. In the fourth epoch, the changes from the previous epoch were kept, namely (1) from *ListOut* the *Ignore* input

causes the robot to transition to the state with *ReferToDesk*, and (2) from *Validate* the *Ignore* response will cause the robot to transition to the state with *Farewell*. In the new edit, from *Greeting* hearing *RequestInfo* will cause the robot to immediately answer the human's question by transitioning to the state with the *Answer* behavior. No further changes were made.

Despite not accepting mutations, the repair algorithm accepted a limited number of new traces into the interaction. Of the 39 users that interacted with the robot, three experienced traces that can only be accepted by the final interaction, one assigning a positive score to their trace, while the others assigning a neutral score. More data from a longer-term study can enable more definitive conclusions about the ability of our approach to introduce novel *good* traces. Other edits were performed to remove edits with a net negative score from the final interaction. Fifteen users experienced traces that are only accepted by the starting interaction, eight assigning a negative score, while seven assigning a neutral score to their interaction.

User Experience & Contextual Fit—We performed one-tailed t-tests to compare user experience and contextual fit between the adaptive ($n = 7$) and the non-adaptive ($n = 6$) conditions in the final epoch of our field study. Users' scores in the adaptive condition ($M = 1.57, SD = 0.53$) were significantly higher than those in the non-adaptive condition ($M = 0.33, SD = 0.52$), $t(11) = 4.23,\ p < 0.01$. However, user experience did not significantly differ between the adaptive ($M = 2.63, SD = 1.01$) and non-adaptive conditions ($M = 2.13, SD = 1.02$). Contextual fit also did not significantly differ between the adaptive ($M = 2.88, SD = 1.08$) and non-adaptive conditions ($M = 2.54, SD = 0.84$).

Video Analysis

Nine codes arose from our analysis of participant videos: *enjoying*, *impressed*, *distracted*, *confused*, *startled*, *irritated*, *uneasy*, *disappointed*, and *disapproving*. We defined codes based on distinct facial features and vocalizations emitted by participants, such as "fidgeting, a nervous smile or laugh, quickly shifting gaze looking around at no-one in particular or people outside of the study area" for *uneasy*. We present four themes from our analysis and their implications for future iterations of our approach, identifying participants as P1-37.

Theme 1: Ambiguous Responses to Robot—We observed participants rate their interaction negatively despite their video data indicating enjoyment. For instance, when P7 asked the robot, "Where is the parking," the robot did not recognize her speech and responded by prompting her to ask a question. P7 responded with laughter despite her most recent query and all of her other queries not being fulfilled, but still assigned a low score to her trace. Conversely, we also observed some participants, such as P24, weigh their interaction positively despite being startled and exhibiting unease because of the robot. In both cases, either the behaviors or the trace ratings of P7 and P24 misrepresented their actual experience. Thus, future iterations of our approach must utilize more sophisticated methods for both capturing and inferring user experience.

Theme 2: Diminishing Experience—We observed across multiple participants that the positive-valence codes *enjoying* and *impressed* were observed in the beginning of their interaction with the robot, while negative-valence codes dominated the rest of their interaction. For instance, at the beginning of her interaction with the robot, P2 expressed enjoyment in the form of excitement when the robot said "Hello," followed by disappointment, irritation, and disapproval. Other participants exhibited the same pattern of initially expressing enjoyment or being impressed, followed by negative reactions to the robot. The implications of this finding are that users' expectations of the quality of the interaction are being broken early-on. Future iterations of our approach should thus collect information about exactly where in a trace the quality of the interaction shifts and then perform targeted edits to the interaction.

Theme 3: Startling the User—We observed four participants being startled by the robot's "Hello" and then exhibiting signs of unease, sometimes resulting in the interaction being terminated early. For instance, after being startled by the robot, P24 turned to the staff members sitting behind the welcome desk and asked "Am I allowed to do this?" After exhibiting more uneasy behavior—namely standing in a stiff manner, looking behind her, and putting her hand in front of her face—P24 ended the interaction, not having said a word to the robot. This case, combined with other instances of participants being startled or uneasy at the beginning of the interaction, demonstrates the critical role that the start of the interaction plays in shaping user experience. Our approach would benefit from being able to target edits to the start of the interaction when user experience is initially poor.

Theme 4: Breaking Correctness Properties—We observed multiple participants prematurely ending the interaction, often immediately after expressing *irritation*, *disapproval*, or other negative emotions and attitudes. This finding describes an instance in which users' goals—wanting to end the interaction—conflicted with a model checking property. In this case, the robot immediately ends the interaction, breaking the property **F** `Farewell`. Our approach can be improved to handle such conflicts. One possible solution is to model *leaves interaction* as a recognizable human response. Another possible solution could be to modify the property itself in the event of a conflict.

DISCUSSION

We discuss the effectiveness of our approach in increasing user experience and contextual fit, finding improved interactions, and making informed modifications.

Overall Effectiveness

We hypothesized that our approach to transforming robot programs would increase user experience (H1) and the perceived contextual fit (H2) of the interaction. Under the controlled conditions of the MTurk study, we found that the perceived user experience of the interaction increased over data collected in one epoch. Additionally, in the field study, we found a significant increase in the final interaction scores over five epochs. Although it is possible that these significant differences arose due to poor quality of the initial interaction, our baseline correctness properties are purposefully non-restrictive such that numerous edits could potentially make the interaction worse,

such as the robot issuing multiple greetings within the interaction. Regardless, we found no significant difference in the the four-item measure of user experience in the field study. Thus, H1 is only partially supported. In both studies, there was no significant difference in contextual fit between the starting and the final interactions. Thus, H2 is not supported.

In the MTurk study, we believe that the lack of difference in contextual fit is partially due to Turkers being removed from the interaction context, thus hindering their ability to accurately gauge whether the robot behaves appropriately. In the field study, we believe the lack of difference in both measures is partially due to the small amount of data that we collected. Collecting more data over more epochs would further characterize sequences of robot behavior, induce further edits, and potentially heighten the differences in user experience and contextual fit. Other factors such as the loudness of the robot and deficiencies in speech recognition and parsing that are not fixed in our approach to transforming interactions may dilute improvements made to the structure of the interaction, thus limiting improvements to user experience and contextual fit.

Algorithm Effectiveness

The findings of our evaluation suggest that our approach is able to remove bad traces from interactions over time. While the MTurk study demonstrated the ability to generate novel good traces through mutations, the field study showed that more work is necessary to uncover the ability of our approach to generate novel good traces under realistic conditions. Additionally, the brute-force nature of our approach restricts its scalability. However, we believe BFS to be similarly effective to state-of-the-art MCMC approaches for overnight computational time frames. Recent work has shown that enumeration can be often more effective than MCMC [19]. Our experience has been similar, and we therefore opted for a BFS approach, as it is simple to implement and debug. Lastly, we found our approach to making mutations in the field study to be largely ineffective. The large random changes made to the robot's behavior at runtime were often too drastic to be up-taken within an overnight run of the repair algorithm. Conversely, our MTurk study shows that smaller positively rated mutations, such as redirecting a single transition, are more easily taken up by our transformation algorithm, as four of the five edits made to the interaction in the MTurk study resulted from mutations.

Making Informed Edits

Transformations to interactions that remove bad traces, maintain good traces, and include positive mutations are directly informed by user feedback. These edits inadvertently cause new, possibly unrated traces to become available for future end-users to experience, which may be removed or maintained in future epochs due to further user feedback. Although user feedback is based on interaction quality, there is little evidence for how each individual edit is tied to context. Some edits intuitively increase interaction quality more than others. In particular, the edit in the field study in which from *Greeting* the *RequestInfo* response causes the robot to transition to *Answer*, as well as the edits in the MTurk study in which from *Validate* the *General* or *Ignore* responses cause the robot to transition to *Answer* all increase the chance that the robot will answer questions. However, whether or not these edits are situated in the context of their respective interactions is unclear.

Limitations and Future Work

The scalability of our BFS approach is our primary algorithmic limitation, hindering convergence to a globally optimal interaction. Under short computational time frames, larger interactions and restrictive sets of properties will cause our approach to experience even greater difficulty in hill-climbing, and in a worst-case scenario, continue to return the original interaction. Although our approach is effective within an overnight time frame, future work should improve our repair algorithm to scale better and hasten convergence. The inability to edit parameters within specific robot behaviors, such as the robot's speech volume or gesturing and gaze behaviors, is another algorithmic limitation of our approach. Future work should investigate how our approach can work in tandem with existing adaptation approaches to adapt both the structure of an interaction and the parameters for individual robot behaviors.

Our evaluation also has a number of limitations. Primarily, we did not evaluate whether our approach can target edits to different contexts or how individual edits are tied to context. Given the practical challenges of administering field studies in parallel, future work may test whether and how adaptations can be targeted to simulated contexts within a laboratory environment. The amount of data collected and epochs performed present another limitation to our evaluation. The data collected in the field study remained sparse over the course of all five epochs, restricting the ability of our approach to effectively address the large variation in user profiles that are common in the field. Our MTurk study collected much more data, but all within a single epoch. Further epochs may still introduce novel negative traces into the interaction that would require even more epochs to remove. Furthermore, a different set of mutations would have likely introduced a different set of edits into the interaction, potentially affecting user experience or contextual fit. Thus, future evaluations must include more data and more epochs to study convergence within our approach.

CONCLUSION

We present an approach to transforming the structure of human-robot interaction programs while adhering to guarantees about the robot's behavior. Our approach involves recording execution traces of end-user interactions with the robot and asking users to score the traces based on the quality of their interaction. We then edit the program using breadth-first search such that execution traces with good scores are accepted by the interaction program, while execution traces with bad scores are not accepted. Our evaluation shows evidence that our approach to transforming interactions improves user experience, but does not show that our approach improves contextual fit.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation (NSF) awards 1651129 and 1925043 and an NSF Graduate Research Fellowship. We thank Bengisu Cagiltay for her assistance in analyzing data, Debbie Edge for her assistance administering the field experiment, and the Wisconsin Institutes for Discovery (WID) for hosting our field experiment.

REFERENCES

[1] Muneeb Ahmad, Omar Mubin, and Joanne Orlando. 2017. A systematic review of adaptivity in human-robot interaction. *Multimodal Technologies and Interaction* 1, 3 (2017), 14.

[2] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

[3] Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, CR Ramakrishnan, and Scott A Smolka. 2011. Model repair for probabilistic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 326–340.

[4] Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Kwiatkowska, Hongyang Qu, and Lijun Zhang. 2013. Model repair for Markov decision processes. In *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on*. IEEE, 85–92.

[5] Chandan Datta, Chandimal Jayawardena, I Han Kuo, and Bruce A MacDonald. 2012. RoboStudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2352–2357.

[6] Dylan Glas, Satoru Satake, Takayuki Kanda, and Norihiro Hagita. 2012. An interaction design framework for social robots. In *Robotics: Science and Systems*, Vol. 7. 89.

[7] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, and others. 2017. Program synthesis. *Foundations and Trends® in Programming Languages* 4, 1-2 (2017), 1–119.

[8] Chien-Ming Huang, Maya Cakmak, and Bilge Mutlu. 2015. Adaptive Coordination Strategies for Human-Robot Handovers. In *Robotics: Science and Systems*.

[9] Michael Huth and Mark Ryan. 2004. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA.

[10] Barbara Jobstmann, Andreas Griesmayer, and Roderick Bloem. 2005. Program repair as a game. In *International Conference on Computer Aided Verification*. Springer, 226–238.

[11] Colin G Johnson. 2007. Genetic programming with fitness based on model checking. In *European Conference on Genetic Programming*. Springer, 114–124.

[12] Gal Katz and Doron Peled. 2008. Model checking-based genetic programming with an application to mutual exclusion. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 141–156.

[13] Simon Keizer, Mary Ellen Foster, Zhuoran Wang, and Oliver Lemon. 2014. Machine Learning for Social Multiparty Human–Robot Interaction. *ACM transactions on interactive intelligent systems (TIIS)* 4, 3 (2014), 14.

[14] Hee Rin Lee and Selma Sabanović. 2014. Culturally Variable Preferences for Robot Design and Use in South Korea, Turkey, and the United States. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction (HRI '14)*. ACM, New York, NY, USA, 17–24. DOI: http://dx.doi.org/10.1145/2559636.2559676

[15] Hee Rin Lee, JaYoung Sung, Selma Šabanović, and Joenghye Han. 2012. Cultural design of domestic robots: A study of user expectations in Korea and the United States. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 803–808.

[16] Joseph E Michaelis and Bilge Mutlu. 2017. Someone to read with: Design of and experiences with an in-home learning companion robot for reading. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 301–312.

[17] Noriaki Mitsunaga, Christian Smith, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2008. Adapting robot behavior for human-robot interaction. *IEEE Transactions on Robotics* 24, 4 (2008), 911–916.

[18] Bilge Mutlu and Jodi Forlizzi. 2008. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 287–294.

[19] Phitchaya Mangpo Phothilimthana, Aditya Thakur, Rastislav Bodik, and Dinakar Dhurjati. 2016. Scaling up superoptimization. In *ACM SIGARCH Computer Architecture News*, Vol. 44. ACM, 297–310.

[20] David Porfirio, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2018. Authoring and Verifying Human-Robot Interactions. In *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, 75–86.

[21] Emmanuel Pot, Jérôme Monceaux, Rodolphe Gelin, and Bruno Maisonnier. 2009. Choregraphe: a graphical tool for humanoid robot programming. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 46–51.

[22] Hannes Ritschel, Tobias Baur, and Elisabeth André. 2017. Adapting a Robot's linguistic style based on socially-aware reinforcement learning. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 378–384. DOI:http://dx.doi.org/10.1109/ROMAN.2017.8172330

[23] Sherry Ruan, Liwei Jiang, Justin Xu, Bryce Joe-Kun Tham, Zhengneng Qiu, Yeshuang Zhu, Elizabeth L Murnane, Emma Brunskill, and James A Landay. 2019. QuizBot: A Dialogue-based Adaptive Learning System for Factual Knowledge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 357.

[24] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. 2016. Planning for Autonomous Cars that Leverage Effects on Human Actions. In *Robotics: Science and Systems*.

[25] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.

[26] Allison Sauppé and Bilge Mutlu. 2014. Design patterns for exploring and prototyping human-robot interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1439–1448.

[27] Eric Schkufza, Rahul Sharma, and Alex Aiken. 2013. Stochastic superoptimization. In *ACM SIGARCH Computer Architecture News*, Vol. 41. ACM, 305–316.

[28] Elaine Schaertl Short, Adam Allevato, and Andrea L Thomaz. 2019. SAIL: Simulation-Informed Active In-the-Wild Learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 468–477.

[29] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research* 16 (2002), 105–133.

[30] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. *ACM Sigplan Notices* 41, 11 (2006), 404–415.

[31] Haodan Tan, Dakuo Wang, and Selma Sabanovic. 2018. Projecting Life Onto Robots: The Effects of Cultural Factors and Design Type on Multi-Level Evaluations of Robot Anthropomorphism. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 129–136.

[32] Adriana Tapus, Maja J Mataric, and Brian Scassellati. 2007. Socially Assistive Robotics. *IEEE Robotics & Automation Magazine* 14, 1 (2007), 35–42.

[33] Adriana Tapus, Cristian Ţăpuş, and Maja J Matarić. 2008. User-robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy. *Intelligent Service Robotics* 1, 2 (2008), 169.

[34] Klaus Weber, Hannes Ritschel, Ilhan Aslan, Florian Lingenfelser, and Elisabeth André. 2018. How to shape the humor of a robot-social behavior adaptation based on reinforcement learning. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*. ACM, 154–162.

[35] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically finding patches using genetic programming. In *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 364–374.