Authoring and Verifying Human-Robot Interactions

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
As social agents, robots designed for human interaction must adhere to human social norms. How can we enable designers, engineers, and roboticists to design robot behaviors that adhere to human social norms and do not result in interaction breakdowns? In this paper, we use automated formal-verification methods to facilitate the encoding of appropriate social norms into the interaction design of social robots and the detection of breakdowns and norm violations in order to prevent them. We have developed an authoring environment that utilizes these methods to provide developers of social-robot applications with feedback at design time and evaluated the benefits of their use in reducing such breakdowns and violations in human-robot interactions. Our evaluation with application developers ($N = 9$) shows that the use of formal-verification methods increases designers' ability to identify and contextualize social-norm violations. We discuss the implications of our approach for the future development of tools for effective design of social-robot applications.

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

Author Keywords
Human-robot interaction; interaction design; authoring; visual programming; verification; program analysis

INTRODUCTION
Robots have long been envisioned as *social agents* assisting and interacting with people in day-to-day environments, making deliveries at an office [30], working as a receptionist [42], or working alongside a human worker on an assembly line [46]. The success of these robots relies greatly on their ability to offer a positive user experience by not only successfully completing their task, but by also adhering to the norms and expectations of human social behavior. Failing to adhere to social norms can decrease interaction quality to the point of causing interaction *breakdowns*. A delivery robot that continuously announces its presence until acknowledged by a human in a hospital emergency room might be so disruptive as to discourage further use of the robot [37]. Alternatively, the robot's attempts at interacting with people might go unnoticed, resulting in breakdowns in initiating social interaction [44].

Ensuring the success of robots in interacting with people will require interaction designers and developers to carefully design robot behaviors that will adhere to the social norms and expectations of the people with whom the robot will interact.

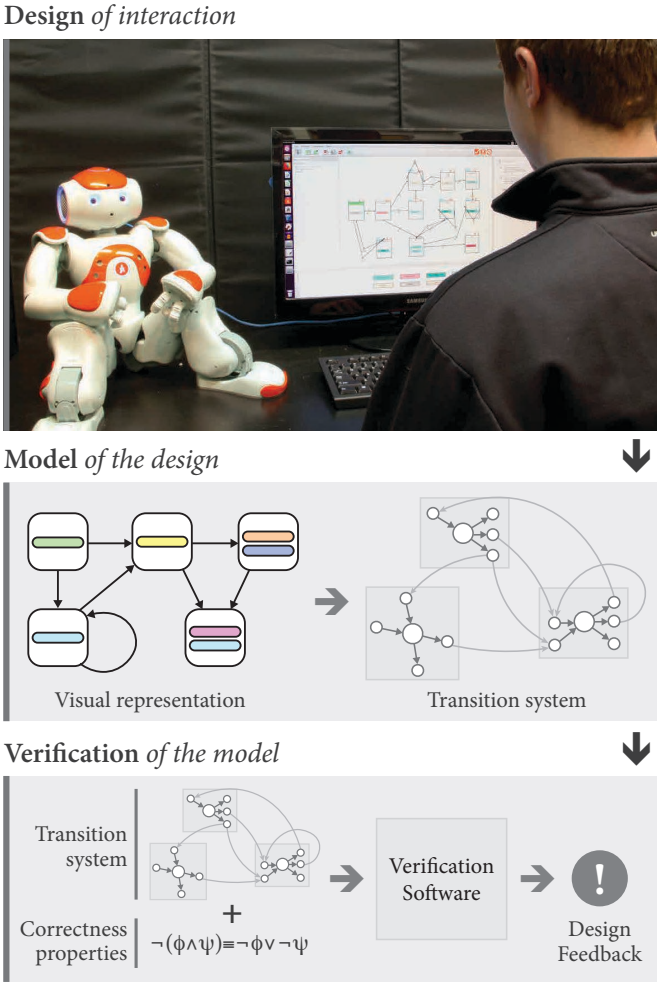

Figure 1. RoVer provides users with a visual environment to design interactions, represents these designs as transition systems, and verifies these systems to determine whether the interactions violate social norms.

UIST '18, October 14–17, 2018, Berlin, Germany

ISBN 978-1-4503-5948-1/18/10…$15.00
DOI: https://doi.org/10.1145/3242587.3242634

This design task, however, involves several challenges: social norms make up a large space of rules and conventions that are highly contextualized and potentially unknown to designers; following social norms requires appropriately coordinating behavior across several modalities such as gaze and gesture; and adherence to one social norm might cause the violation of another if the norms are not holistically taken into consideration. Although visual-programming [e.g., 45, 1] and learning-based [e.g., 22, 34] approaches can facilitate the design of robot behaviors, these approaches do not check that robot behaviors adhere to human social norms.

In this paper, we explore how concepts and methods from *formal verification*, a computational approach to proving that programs do or do not adhere to certain properties [50], can assist designers in ensuring that human-robot interactions adhere to social norms and in preventing interaction breakdowns. We developed a method for constructing human-robot interactions that supports automated detection of social breakdowns and implemented it into an authoring environment for robot behaviors, RoVer, which employs the formal-verification technique of *model checking* [6] to ensure that designed interactions satisfy social-norm specifications and task expectations. In RoVer, the model checker runs in the background, detects possible social-norm violations, and reports them to the designer who can then modify the design to ensure its adherence to the violated social norms. To bridge the gap between model checking and human-robot interaction, we formalize social norms and expectations in *temporal logic* [40], a common approach to specifying correctness properties of both software and hardware. We then demonstrate how human-robot-interaction models can be composed of micro-scale interaction units, called *microinteractions*, with state-based *transition systems* as each microinteraction's core component [6]. Modeling interactions as transition systems and specifying social norms in temporal logic enable the application of model checking to ensure that the interaction adheres to the norms.

The evaluation of our approach asked designers to use RoVer to construct interactions for a given scenario with specific context-based social-norm and task properties. We analyzed the quality of their designs and measured their experience with the verification-aided design.

In summary, our contributions are as follows:

Novel Approach to HRI Design—We present a novel approach to designing human-robot interactions in which social norms play a first-class role in the design process.

Verification-Backed Design—Our approach utilizes formal verification techniques where human-robot interactions are defined as state-transition systems and social norms are defined as logical properties over interaction states.

Development and Evaluation of RoVer—We implement our approach in RoVer, a new visual interaction authoring tool, and assess its benefits and limitations.

RELATED WORK

Our work builds on prior research in social robotics, interaction design, and formal verification. Below, we provide brief summaries of relevant work from these areas.

Social Robotics

The development of our verification-based design approach draws on the body of literature that establishes the design space for robot behaviors that interact with people [e.g., 36]. When these behaviors are designed appropriately, the robot's interactions with its users are more effective and positive. When they are not designed appropriately, interactions may result in social and task breakdowns. This literature, for example, describes norms for when the robot should look toward an object of interest with *referential gaze* as opposed to its partner with *affiliative gaze* and highlights that the proper use of these gaze cues can increase task outcomes and improve perceptions of the robot [4, 20, 35]. The literature also describes norms for *cognitive gaze aversion*, where a robot looks up before speaking to signal thought, *intimacy-modulating gaze aversion*, where a robot periodically looks slightly to the side while speaking or listening, and *floor-management gaze aversion*, where the robot averts its gaze to "hold" or looks toward its interlocutor to "pass" the speaking floor [38, 3, 5].

Social norms are highly context dependent such that a behavior effective in one context might be inappropriate in another. For example, a study of people's perceptions of a hospital delivery robot found that the robot's attention-seeking behavior was perceived as being "disruptive" at a hospital unit where staff had low interruptability but as being "friendly" at a unit where staff had high interruptability [37]. The appropriateness of social norms can also depend on the goals of the interaction. For example, in situated interactions, the use of *deictic gaze* might improve user task performance, while the use of *eye contact* might improve user perceptions of the robot [21].

Adherence to certain social norms might also result in conflict with other social norms. For instance, a robot interacting with a group of people might be designed to greet newcomers or bid farewell to people who leave [16]. Although issuing greetings and farewells are important social norms for most interactions [26, 42], adhering to these norms might result in interruptions in the flow of the conversation in the context of group interactions [38]. Thus, a robot's behaviors must be carefully coordinated to manage conflicting social norms.

Interaction Design

Interaction-design tools for robots range from low-level programming languages [e.g., 7] to visual programming environments [e.g., 11]. For example, the popular visual-authoring environment Choreograph, developed to program the NAO robot by Softbank Robotics, offers users the ability to visually program high-level behaviors while allowing access to low-level control of movements [41]. RoboFlow offers program expressiveness through a visual environment, although it focuses on task-based programming rather than specifying social behaviors for a robot [1]. RoboStudio [12], iCustomPrograms [11], Interaction Blocks [45], and TiVipe [33, 8] offer visual interfaces for the design of robot social behaviors with a focus on usability for non-programmers. Interaction Composer [16] also focuses on usability while providing a design framework to increase interaction quality.

Many of the tools above have modularized the robot's behaviors and actions into discrete components, and some have for-

malized different *patterns* of dyadic interactions into discrete state-transition structures [e.g., 45, 39, 18]—an approach we adopt in the current work to formalize our underlying representation of microinteractions. For instance, a simple "question-answer pair" pattern can be discretized into states for "asking" and "answering" with transitions connecting each state. This representation supports modularity and scalability and enables systematic computational analysis.

Formal Verification

Formal verification involves systematic analysis of programs against a set of program properties or expectations [50]. The field has a long history—dating back to Turing [47]—and many different approaches to the problem. To be able to reason about social norms, we chose to utilize the rich foundations of *model checking* [6] and *temporal logic* [40], as interactions can be viewed as state machines and social norms as properties over traces in the state space that make up the interaction. Interactions can also be viewed as human-in-the-loop systems, in which control is shared between the human and the robot. Prior work synthesizes such systems from temporal-logic formulas [32]. Other work in *program synthesis* enables designers to specify the constraints of a system and its environment in temporal logic and provides them with feedback in the form of suggested additional environmental constraints [31, 2], contrary to our approach of manually designing interactions and receiving feedback from pre-existing temporal logic specifications. While program synthesis requires a full specification of the system, *automatic program repair* requires only a partial specification to correct a faulty program [19]. *Interactive* repair involves automatic modifications to a program based on designer corrections to the program output [24].

Applications of verification to robotics includes the checking of mission-critical properties [14] and using temporal logic and model checking to generate motion plans [15, 27]. Within *social* robotics, recent work includes applying formal verification to ensure safety and trustworthiness [48, 13, 49] and theoretical investigations of human-machine trust through the lens of probabilistic temporal logic [23, 28]. Prior work also applies verification to check mission-critical properties of human-robot teamwork [9]. To our knowledge, no prior work applies formal verification techniques to human-robot interaction design with a focus on checking the appropriateness of a robot's social conduct. This paper seeks to bridge this gap.

TECHNICAL APPROACH

This section presents our technical approach to authoring and verifying human-robot interactions by (1) formalizing the building blocks of our approach and (2) illustrating its application to an example design problem in RoVer.

Building Blocks

Our approach consists of three primary building blocks:

1. *Microinteractions* represent the most granular form of an interaction between a human and a robot. For instance, a question-answer microinteraction may involve the robot asking a question and the human responding.
2. *Groups* encapsulate multiple microinteractions that run in parallel and make up more complex interactions. For instance, asking a question and gesturing toward an object can be grouped together into an "inquiry" group.
3. *Interactions* include sequences of groups that form a complete interaction, such as exchanging greetings, asking a question, and bidding farewell performed in a sequence.

Our approach also enables interaction designers to specify social norms as logical statements and to automatically verify if their interaction adheres to the set of specified norms.

Robot Delivery Scenario

Throughout this section, we will use the example interaction of "a robot delivering a package to a human" to illustrate how a designer might use our tool to avoid violating social norms. We assume that our example robot is humanoid with gaze and gesturing capabilities and that the designer is given the following specifications: (1) the interaction begins when the robot issues a greeting to the human; (2) both the robot and the human are already in close proximity; (3) after the greeting, the human is attentive, and the robot must authenticate the human's identity; (4) after the human has signed for the delivery, the robot must give the delivery to the human; and (5) after the delivery has been successfully completed, the robot must bid farewell to the human.

In the scenario above, the robot must adhere to numerous task requirements and social norms including the following:

Greeting norm: The robot should issue a greeting at the beginning of the interaction [26].

Farewell norm: The robot should issue a farewell at the end of the interaction [42].

Speech norm: The robot should not interrupt the human's speech while the human has the speaking floor [43, 38].

Gaze norm: The robot should coordinate its gaze during the delivery such that it engages in referential gaze by looking at the package during handoff [35] and otherwise in intimacy-modulating gaze aversion by shifting its gaze slightly from side-to-side while speaking [5].

Noninterruptability norm: The robot will properly exit the waiting cycle after the previously inattentive human acknowledges its presence and not perpetually re-issue unneeded greetings or alerts before being acknowledged [37].

In our illustration of RoVer, we will describe how our example delivery interaction scenario can be implemented, how the social norms can be specified and verified, and how robot behaviors that do not adhere to the norms might be revised.

Semantic Foundations

Before we proceed with describing the building blocks of our approach, we outline its semantic foundations.

A *state s* of the interaction between a robot and its user represents the condition of the human and the robot at a particular point in time. For instance, in state *s*, the human might be *speaking* while the robot is *silent* or *listening*. Note that the

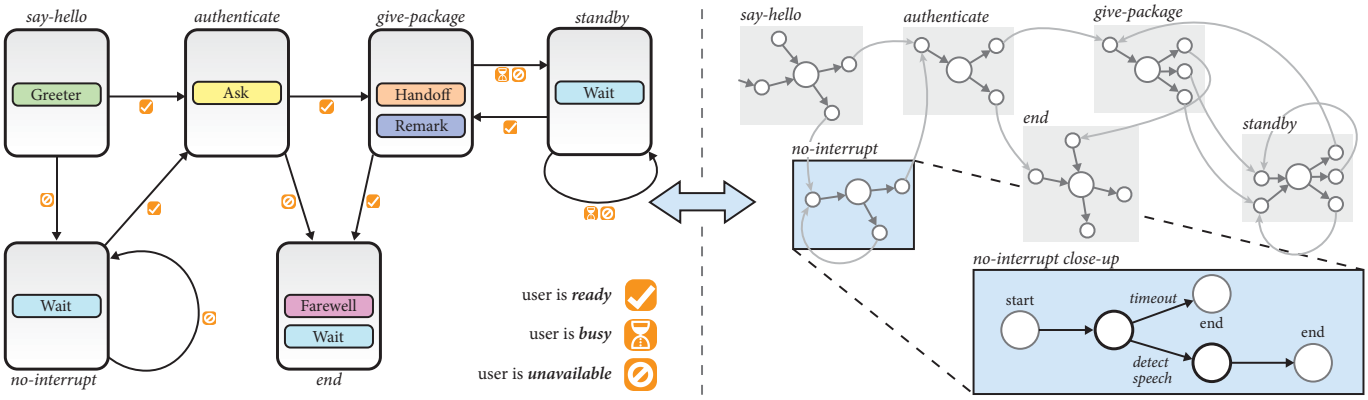

Figure 2. Left: The design of a delivery interaction as implemented in RoVer. Right: A state-space representation of the delivery interaction shown on the left. The gray boxes represent microinteractions one of which is highlighted in blue and expanded to show labels on transitions and states.

amount of information tracked by a state can be arbitrarily complex, depending on robot capabilities and the granularity with which we wish to represent the interaction.

Intuitively, every action performed by the human or the robot—e.g., the human beginning to speak—transitions the interaction from some state s to some state s'. We thus view an interaction in our system as a *labeled transition system*, which defines a relation from *input state* s to some *output state* s', depending on the *actions* (labels) taken by the human and the robot.

Figure 2 (right) shows an abstract view of the labeled transition system representing the final design for our illustrative delivery example. In RoVer, designers do not work explicitly with labeled transition systems; instead, they design interactions using higher-level syntactic building-blocks (as shown on the left of Figure 2), which are in the background semantically interpreted as transition systems for purposes of verification.

Microinteractions

Our work modularizes patterns of interaction into *microinteractions* that include both robot and human behaviors. Our tool, RoVer, incorporates the concepts proposed by Kahn et al. [25] for using design patterns in human-robot interaction: patterns must (1) capture physical and social interaction as opposed to solely human or robot behavior, (2) be abstract in specification and parameterizable, enabling multiple instantiations, (3) be modular, and (4) organized hierarchically where more complex patterns encapsulate smaller patterns.

Semantics of Microinteractions

The core building block in our approach is a microinteraction—the smallest and simplest possible interactions between a human and a robot. Examples include the *Ask* microinteraction, which involves the robot asking a question and the human either answering or not answering the question. Formally, we define a microinteraction m as a labeled transition system. A microinteraction thus begins in some initial state s and, depending on the actions taken by the human and the robot, progresses through multiple states until it arrives at an output state s', when the microinteraction is considered complete.

Illustrating Microinteractions

RoVer supplies a library of *parameterizable microinteractions* that interaction designers can customize to suit their needs. Thus, designers do not need to create microinteractions from scratch, but they simply customize existing ones. Each parameter in a microinteraction can change its structure or behavior. The parameters to *Ask* include the question that the robot asks and the answers that the robot can understand. In Figure 2, microinteractions are denoted by colored boxes.

In *Ask*, the parameters do not change the structure of the transition system but instead change the behavior of the microinteraction at runtime by linking the answers that the robot can recognize to specific outputs. Within our delivery example, the robot can be set to confirm the human's name with the question, "Are you <name>," to understand the answers "Yes" and "No," and link them to particular outputs. If the robot does not hear speech from the human at all or does not understand the answer even after a few requests for repetition, *Ask* will consider the human to be unable to proceed.

Groups

Semantics of Groups

In order to construct complex, nuanced interactions, our approach allows composing *groups* of microinteractions. For instance, a group may combine the pointing behavior from one microinteraction and the gaze behavior from another. Formally, a group g consists of a set of microinteractions $\{m_1, m_2, \ldots, m_n\}$. Semantically, we view g as the *parallel composition* of the n microinteractions, denoted as follows:

$$m_1 \parallel m_2 \parallel \ldots \parallel m_n$$

In other words, we consider microinteractions as n processes that execute in parallel. Consider a scenario where a robot is expected to hand an object to a human while it greets the human. This design can be achieved by constructing a group $g = \{m_g, m_h\}$, composed of, in parallel, the *Greeter* and *Handoff* microinteractions, denoted by m_g and m_h, respectively.

Illustrating Groups

In the example delivery scenario, the interaction designer may wish for the robot to speak to its user as it hands the package. In RoVer, as illustrated in Figure 2, groups are depicted by boxes containing microinteractions; for instance, the *give-package* group composes *Handoff* and *Remark*.

Each group will have a set of possible input and output states that the user can use to compose groups (as we will see later). The possible inputs and outputs are included in Figure 2. *Ready* indicates that the robot believes that the human is ready to proceed within the interaction. *Unavailable* indicates that the human has signaled in some way or the robot has reason to believe that the human is not ready to proceed. *Busy* indicates that the robot does not know whether the human is ready or not, e.g., when the human hesitates to proceed in the task. RoVer provides the designer with information on a group's inputs and outputs in the form of icons placed over the representation for each group (not included in Figures 2–3 for readability).

Interactions

Semantics of Interactions

At a high level, a network of groups transitioning among each other creates a comprehensive transition system that computationally represents the interaction as a whole.

An *interaction* is a labeled transition system that is constructed by composing groups in two different ways. First, we can simply compose two groups—g_1 and g_2—*sequentially*, executing the first group and then the second, for instance, executing a group with *Greeter* followed by another group with *Ask*. Second, we can compose groups *conditionally*, where g_1 is followed by g_2 depending on some condition on the output state of g_1. For instance, if the human ignores the robot after a greeting, as done by shoppers observed by Satake et al. [44] in a shopping mall, the interaction could proceed to bidding farewell instead of proceeding with the delivery. An example of using multiple conditions on a transition is shown in Figure 2, through which the interaction may transition from *give-package* to *standby* if the human is busy or unavailable. All groups in Figure 2 are composed conditionally.

Regardless of how groups are composed, RoVer employs type-checking to ensure that the outputs of g_1 are a subset of the allowable inputs of g_2. If transitions extending from g_1 are composed conditionally, RoVer checks to ensure that each output state of g_1 is included in a condition exactly once.

Illustrating Interactions

Figure 2 presents a final design for the delivery interaction that satisfies the criteria previously outlined. The transitions (arrows) between groups denote the temporal ordering of events. The icons around a particular transition denote the end state that must be met within the source group in order to transition to the target group. For instance, from the *give-package* group, if the human is busy and does not accept the package, the robot will transition to a waiting state, where it waits indefinitely until the human acknowledges the robot with speech.

Specifying and Verifying Social Norms

Specification

Now that we have defined and illustrated the building blocks of our technical approach, we discuss how social norms can be specified. A key observation of this work is that many social norms can be succinctly expressed as formulas in *linear temporal logic* (LTL), a widely used logical grammar for specifying correctness properties of software and hardware designs [6].

At a high-level, LTL enables specifying what events should and should not happen and in what temporal order they should happen. In the context of the previously outlined social norms of our delivery robot, LTL provides us with *modal operators* to formalize such statements. For instance, using the *global* operator (denoted **G**), we specify conditions that should hold in every state. Using the **G** operator, the speech norm of "respecting conversational floor" can be formalized in LTL as:

$$\mathbf{G}\ \texttt{humanSpeaking} \longrightarrow \neg\ \texttt{robotSpeaking}$$

In other words, *in any state s* of the interaction, if the human has the speaking floor, *then* ($\rightarrow$) the robot should not speak. Here, `humanSpeaking` and `robotSpeaking` describe the set of states in which the human and the robot speak, respectively. Similarly, **G** allows us to check for potential conflicts in the robot's gaze across concurrently executing microinteractions:

$$\mathbf{G}\ \neg(\ \texttt{referentialGaze} \wedge \texttt{intimacyModulatingGaze}\ )$$

Here, `referentialGaze` and `intimacyModulatingGaze` describe the set of states involving different gaze types. Within individual microinteractions, gaze behavior is designed to satisfy this social norm. If one microinteraction executing the handoff runs concurrently with another executing the speech, the above LTL property checks that both microinteractions will not attempt to concurrently execute both gaze behaviors.

LTL also allows us to talk about future events using the modal operator **F**. This operator enables us to check the farewell social norm with the LTL syntax below, in which `Farewell` refers to the set of states within the *Farewell* microinteraction:

$$\mathbf{F}\ \texttt{Farewell}$$

As a more complex example, LTL provides the *next* operator **X** to indicate something that should be true in the immediate next state and the *until* operator **U** to indicate that ϕ should remain true until ψ happens and that ψ must happen. We check the noninterruptability social norm with the LTL syntax as below:

$$[\ \texttt{robotSpeaking} \rightarrow (\ \mathbf{X}\ \neg\ \texttt{robotSpeaking} \vee \mathbf{X}\ \texttt{humanReady}\ )\ ]\ \mathbf{U}\ \texttt{humanReady}$$

For simplicity, we use **X** in the above property to denote the next microinteraction. Thus, the property translates to "until the human is ready, if the robot is speaking, then either it will not be speaking in the next microinteraction, or the human will be ready to proceed." In other words, before its user is ready, the robot should not engage in multiple consecutive microinteractions involving robot speech. Applying this property to real-world human-robot interactions would address situations where the robot repeatedly makes announcements to, and eventually irritates, unavailable users [37].

Lastly, to check the greeting norm, we simply must ensure that the initial state corresponds to *Greeter*.

Verifying Social Norms

The interaction in Figure 2 satisfies all of the norms outlined at the beginning of this section. To illustrate how these social norms might be violated, we will walk through constructing the interaction in the following paragraphs.

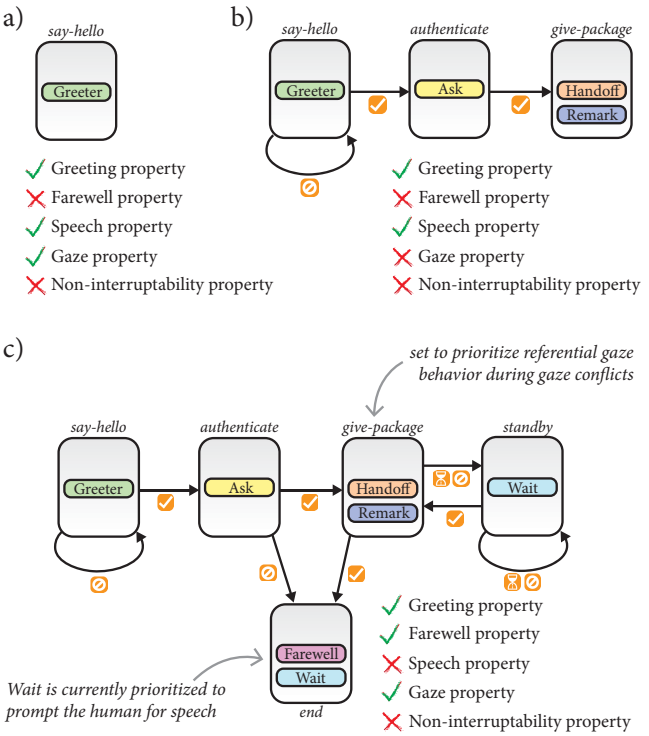

Figure 3. A walk-through of the construction of the delivery interaction. Satisfied and violated properties are shown at each step. The complete interaction is shown in Figure 2.

Figure 3a depicts the start of the design. This simple interaction starts with a greeting, satisfying the greeting social norm. The absence of a *Farewell* microinteraction leads to the violation of the farewell property. The noninterruptability property is violated because there exists a path through the interaction, consisting only of *Greeter*, in which the human is never ready to proceed. The human may ignore the robot's greeting, and the interaction will terminate. The speech and gaze properties remain satisfied since no two microinteractions are composed concurrently. For the violated properties, RoVer presents feedback in real-time during design as a list of descriptions in plain English language, such as stating, *"The interaction does not end with a farewell,"* to provide feedback on the violation of the farewell norm. For some properties, RoVer can pinpoint the source of the violations to specific groups and provide this information to the designer, while RoVer may not precisely localize other property violations.

Figure 3b depicts a minimally functional delivery interaction in which the robot greets the human, confirms the human's identity, and hands off the package while making a remark. The farewell property remains violated, and there now exist two points of premature termination: after the *authenticate* group, e.g., if the robot fails to confirm the human's identity in *Ask*, and after the *give-package* group. Additionally, the parallel execution of both microinteractions in the *give-package* group has caused a potential conflict between referential gaze behavior in *Handoff* and intimacy-modulating gaze behavior in *Remark*. RoVer handles gaze properties differently from other violations in that if a gaze violation is possible, then RoVer will ask the designer to prioritize a behavior. It is then

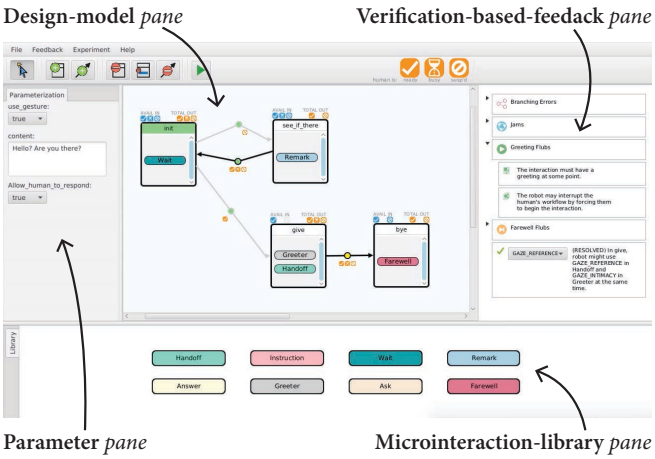

Figure 4. The user interface for RoVer, including a *design-model pane* that serves as the canvas for the designer to construct interactions, a *parameter pane* that provides contextual parameter options for behaviors and microinteractions, a *library pane* that provides a draggable library of available microinteractions, and a *feedback pane* that provides the designer with feedback based on verification analysis.

left to the designer to select the referential gaze behavior. Now, the noninterruptability property remains violated, because the self-loop on the *say-hello* group makes it possible for the robot to speak in two consecutive microinteractions before being acknowledged by the human.

Figure 3c shows an almost-complete interaction. The gaze property violation has been addressed by prioritizing a specific gaze behavior. If referential gaze is successfully prioritized, the robot will keep its eyes on the package even while simultaneously speaking to the human. Similarly, the farewell property violation has been addressed, as all paths through the interaction lead to *Farewell*, assuming that the human does not keep the robot waiting forever without responding (which is known as a *fairness* constraint in model checking [6]). The noninterruptability property remains violated for the same reasons as before. The speech property is now violated as the robot speaks within *Farewell* and by default prompts the human to speak in *Wait*. The parallel execution of *Wait* and *Farewell* might cause the robot to interrupt the human. *Farewell* must contain robot speech, but *Wait* can be parameterized such that the robot does not prompt the user to speak.

In the final version of the delivery interaction, shown in Figure 2, the violation of the speech property has been fixed by parameterizing *Wait*, and the violation of the noninterruptability property has been addressed by adding the *no-interrupt* group, so that the robot waits silently for the human's attention.

Implementation

RoVer is implemented in Java version 8 and uses the PRISM Model Checker [29] to perform verification of interaction designs in the backend.[1] The user interface for RoVer is depicted in Figure 4. Interaction designs can be saved in XML format. At any point while designing an interaction, users can simulate their interactions on a robot platform, which causes an XML

[1]The source code for RoVer is publicly available as open-source at https://github.com/Wisc-HCI/RoVer.

file of the interaction to be sent to the robot and used as input to guide the execution of the interaction.

For our implementation and testing, we used a Softbank Robotics NAO robot. We implemented the code to execute interactions in the robot in Python version 2.7, using version 1.4 of NaoQi [41]. Each microinteraction corresponds to a Python function that is called as necessary during the execution of the interaction. The parameters for each microinteraction are given as inputs to each function. Microinteractions that are composed concurrently are multi-threaded on the robot.

EVALUATION

In addition to describing the design and envisioned use of RoVer we present a preliminary user study that examines the effects of our verification-based approach on the human-robot interaction design process. The study tested our central hypothesis that feedback from verification will improve the process of authoring human-robot interactions, specifically (1) reduce the amount of time spent looking for and fixing errors, (2) improve the ability to identify and contextualize design errors by decreasing the discrepancy between perceived and actual errors, (3) decrease the overall effort required by participants, (4) enhance the quality of interaction designs, and (5) improve ease of finding, interpreting, and fixing errors. Below, we describe our study design, participants, and results.

Study Design

We designed a between-participants study with two conditions: (1) *assisted*, where participants received feedback from verification, and (2) *nonassisted*, where they did not receive feedback. All participants used the implementation of RoVer described in the previous section.

To better assess the effects of verification on the outcomes discussed in our hypotheses, we contextualized the study in a *design-interpret-fix* pipeline in which all participants first designed their interaction without verification. At the *interpret* stage, participants were given information on the errors in their design according to their condition. Finally, participants had the opportunity to fix errors. Participants did not have access to simulating their designs on a robot. Preliminary trials with RoVer showed that users in the nonassisted condition relied heavily on simulation, and providing this option would have confounded our testing of the use and effects of verification.

Interaction Design Scenario

We provided designers with the six microinteractions shown in Figure 2. Additionally, RoVer included the microinteractions *Answer* that involved the robot answering a question from the human and *Instruction* that involved the robot issuing an instruction to the human. We asked participants to design an interaction with an informational robot at the entrance to a university building. The design specifications for the interaction included the following: (1) the robot must wait for a human to approach it, (2) greet each human that approaches it, (3) answer questions for as long as the human continues to ask them, and (4) say goodbye to the human at the end of the interaction. The specific list of task and social expectations for this interaction, which we converted to LTL, is shown in Table 1. We also included the same speech property and similar gaze properties discussed in the previous section. We categorized each expectation as either *Social* or *Task/Social*, depending on whether the expectation was related to the robot's social conduct or its performance of its task, which in the case of an informational robot had both task and social implications.

Procedure

At the start of the experiment, participants were told that they would be designing an interaction between a robot and its user using RoVer. Figure 5 depicts the study setup. We also informed participants that they would not be able to use the robot to simulate their interactions. After providing informed consent, we trained participants on the basics of designing interactions using a modified version of RoVer with feedback from verification and type-checking deactivated. The training process involved watching a video on the basics of microinteractions, watching a second video describing the modified RoVer, and a hands-on activity that guided participants through the creation of an example interaction. We then briefed each participant on the interaction-design scenario and gave them a list of task and social norm expectations that their designs had to satisfy (Table 1). Participants were given as much time as they needed to design their interactions.

After participants indicated that they had finished their designs, we asked nonassisted participants to attempt to manually verify their interactions by reviewing a copy of the list of task and social norm properties printed on paper and record which properties they believed their designs violated and where the violations occurred. For assisted participants, we turned on verification and type-checking within RoVer so that participants could see a snapshot of which properties their designs violated, and—for the properties that could be pinpointed to a specific area in the interaction—where the violations occurred. We then asked each participant in the assisted condition to record which properties they believed their design violated and where the violations occurred. All participants received a printed image of their designs that they could annotate.

After participants looked for errors in their designs and feedback was turned off in the assisted condition, we asked participants to fix the errors that they had flagged. During the entire

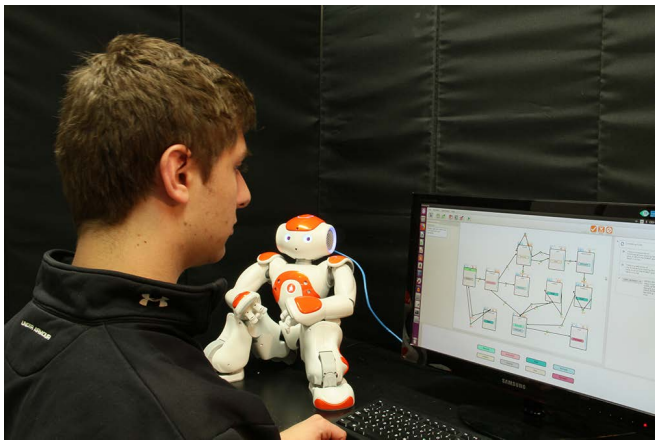

Figure 5. An experimenter demonstrating the setup of the user study, including the authoring environment on a desktop computer on the right and the NAO robot to the left of the computer.

Table 1. List of social and task norms that participants were asked to satisfy when designing their interactions. "Begin" and "end" refer to the sets of states at the beginning and end of microinteractions respectively, and "deadlock" refers to states where no further transitions are possible. (*) X refers to the next group, not necessarily the next state. (**) Negating an LTL property is a technique for reasoning about *some* paths as opposed to all paths. If the negation is satisfied, then the actual property is unsatisfied, and vice versa. Translations are provided for the more complex properties.

Waiting Expectations

Category	*Description*	*LTL Form*
Task/Social	The robot must not interrupt the workflow of any human while it is waiting for a human to approach it.	[robotSpeaking →(**X** ¬ robotSpeaking∨ **X** humanReady)] **U** humanReady* (translates to *until the human is ready to proceed, if the robot is speaking in one group, in the next group either the human will be ready or the robot is silent*)

Greeting Expectations

Category	*Description*	*LTL Form*
Task/Social	The robot should issue a greeting at some point.	**F** (Greeter ∧ begin)
Task/Social	The robot should not greet humans that have not approached it.	**G** ¬ (Greeter∧ humanUnavailable∧ begin)
Social	The robot should never greet the same human twice.	**G** [Greeter ∧ end → ¬ (**F** Greeter∧ begin)]

Answering Expectations

Category	*Description*	*LTL Form*
Task/Social	The interaction should not end before the robot has answered any questions.	**F** Answering
Task/Social	After successfully answering a question (the human is ready), the human should be able to ask arbitrarily many more questions if they want.	¬ [**G** (Answering ∧ humanReady ∧ end → **F** Answering ∧ begin)]**
Task/Social	After successfully answering a question (the human is ready), the robot does not need to continue asking questions.	¬ [**G** (Answering ∧ humanReady ∧ end → ¬ (**F** Answering ∧begin))]**
Task/Social	After unsuccessfully answering a question (the human is unavailable), the robot should not listen for any more questions from the human.	**G** [Answering ∧ humanUnavailable ∧ end→ ¬ (**F** Answering∧ begin)]

Farewell Expectations

Category	*Description*	*LTL Form*
Task/Social	The interaction should eventually end.	**F** deadlock
Task/Social	When the interaction ends, it must end with a farewell.	**G** [deadlock → Farewell]

Turn-Taking Expectations

Category	*Description*	*LTL Form*
Social	There should not be instances in which the robot speaks twice in a row.	**G** [(robotSpeaking ∧¬ humanSpeaking) → (**X** humanSpeaksFirst)]* (translates to *if within a group the robot speaks and the human stays silent, then in the next group the robot will allow the human to speak first*)
Social	There should not be instances in which the human is prompted to speak twice in a row.	**G** [(humanSpeaking ∧¬ robotSpeaking) → (**X** robotSpeaksFirst)]* (translates to *if a group involves the human speaking and the robot staying silent, then in the next group the robot will speak first*)

fix phase, participants had access to the printed image of their designs and annotations. After the experiment, participants filled out several questionnaires that assessed their experience, including SUS [10] scores that measured RoVer's usability.

Participants

A total of nine students recruited from the University of Wisconsin–Madison campus took part in the study. All participants were native English speakers between the ages of 18 and 22 (M = 19.7, SD = 1.4) and had completed at least one semester of programming courses.

Measures & Analysis

To assess the effects of verification-aided feedback on the design process, we captured several measures of design quality, performance, and experience. Design performance was measured using the time participants spent in the interpretation and fix phases. We corrected time spent interpreting errors by the complexity of participants' initial designs, calculated as the sum of groups, microinteractions, and transitions between groups. Similarly, we corrected time spent fixing errors by the number of perceived errors recorded by participants during the interpretation phase. We also calculated error discrepancy for each participant, expressed as the difference between the recorded number of perceived errors versus the number of actual errors present at the end of the design phase. We corrected this difference for design complexity. To capture design experience, we measured designer task load using the NASA Task Load Index (TLX) [17] and designer experience using a questionnaire with items to measure the participants' perceived ease of finding (four items, Cronbach's $\alpha = 0.89$), understanding (four items, Cronbach's $\alpha = 0.82$), and fixing (three items, Cronbach's $\alpha = 0.79$) errors. Finally, to measure design quality, we captured the number of actual errors present at the end of the fix phase (corrected for design complexity), offset by the number of errors present after the design phase (also corrected for design complexity). Errors included

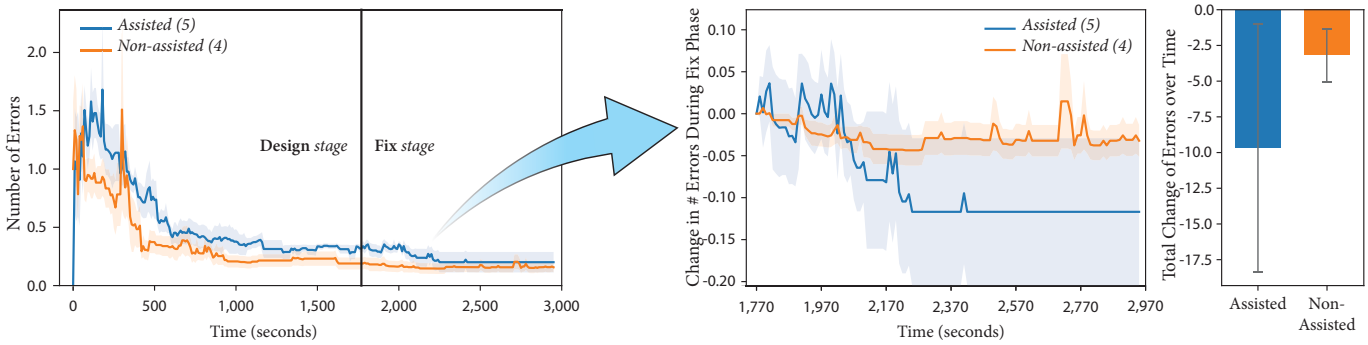

Figure 6. Left: Errors tracked for each participant in the design and fix stages, corrected for design complexity. Right: Change in the corrected number of errors during the fix process (center-right), and the summed change over time during the fix process (right).

property violations, type-checking errors, and errors related to incorrect conditional logic on transitions between groups.

Our analysis of the data from the above measures involved one-tailed Student's t-tests. We used α levels of .05 and .10 for significant and marginal effects, respectively.

Results

The analysis of our data showed that feedback from formal verification increased designers' ability to accurately find errors in their designs and the ease of finding and understanding these errors. The paragraphs below provide descriptive statistics and test details for these effects.

The average corrected time interpreting errors for assisted and nonassisted participants were 23.8 ($SD = 4.51$) and 19.7 ($SD = 8.26$), respectively. Average corrected time fixing errors was 66.0 ($SD = 32.8$) and 80.0 ($SD = 42.0$) for assisted and nonassisted participants, respectively (Figure 7). Due to technical issues, we discarded data from one nonassisted participant for time fixing errors. Error discrepancy in the assisted condition ($M = 0.13$, $SD = 0.05$) was significantly lower than in the nonassisted condition ($M = 0.31$, $SD = 0.19$), $t(7) = 2.08$, $p = 0.04$ (Figure 7).

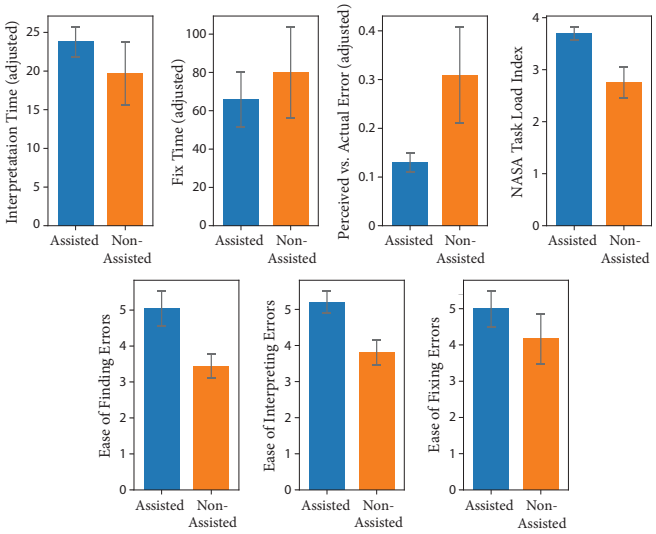

Figure 7. Results from the user study show that verification assistance decreases error discrepancy (top-center-right) and increases ease of finding and interpreting errors (bottom-left, bottom-center). No improvement is shown in time interpreting or fixing errors (top-left, top-center-left), effort expended (top-right), or ease of finding errors (bottom-right).

Contrary to our prediction, the average effort expended by participants in designing interactions, measured using the NASA TLX, in the assisted condition ($M = 3.70$, $SD = 0.32$) was not less than the effort expended in the nonassisted condition ($M = 2.75$, $SD = 0.62$) (Figure 7).

Assisted participants found finding errors to be significantly easier ($M = 5.05$, $SD = 1.11$) than did nonassisted participants ($M = 3.44$, $SD = 0.72$), $t(7) = 2.50$, $p = 0.02$. Additionally, assisted participants ($M = 5.20$, $SD = 0.74$) found understanding errors in their designs to be significantly easier than did nonassisted participants ($M = 3.81$, $SD = 0.72$), $t(7) = 2.84$, $p = 0.01$. Assisted participants did not find it significantly easier to fix errors ($M = 5.0$, $SD = 1.13$) than did nonassisted participants ($M = 4.17$, $SD = 1.40$) (Figure 7). The change in the amount of errors in the fix phase was not significantly lower in the assisted condition ($M = -0.12$, $SD = 0.20$) than in the nonassisted condition ($M = -0.03$, $SD = 0.04$) (Figure 6, center-right). For one participant, RoVer exhibited an inconsistency in setting the starting states of the interaction, requiring this participant's error data to be manually corrected.

Lastly, assisted participants reported an average SUS score of 77 ($SD = 14.5$), while nonassisted participants reported an average SUS score of 77.5 ($SD = 11.7$). We believe the similarity between scores result from the conditions only differing in the interpretation phase in which most functionality of the system was deactivated and modifications to designs was disallowed.

These results provide partial support for our central hypothesis, showing improvements in (1) the ability to identify and contextualize design errors and (2) the ease of finding and interpreting errors. On the other hand, we found no improvements in (2) the ease of fixing errors, (3) the amount of time spent finding and fixing errors, (4) the overall effort required by participants, and (5) the quality of interaction designs.

DISCUSSION

Implications for Interaction Design

The demonstration of RoVer has shown that our design and verification framework is capable of verifying a wide range of social norms in temporal logic and of expressing complex

interactions with only a small set of microinteractions. The use of verification in designing human-robot interactions has the potential to prevent errors introduced at the design stage, aiding in the development of more effective applications of social robots for a variety of day-to-day scenarios of use.

Verification is a powerful technique to aid designers in reducing errors in human-robot interactions. The results from our user study confirm that the use of verification in RoVer helps designers better understand where errors are in their designs and that designers are aware that the process is easier when using verification. The ability to identify errors is critical to developing interactions that are unlikely to encounter breakdowns, better facilitating deployment of these robots. Designer's subjective understanding that addressing potential design breakdowns is made easier with verification is particularly important given the increased cognitive load that verification bears on designers, helping them feel that the tradeoff of the cognitive burden for ease of use is justified.

Limitations and Future Work

Despite the wide range of interactions that various combinations of a small set of microinteractions affords, the size of the microinteraction library and limited ability to parameterize microinteractions are some of the improvements that users wanted to see in RoVer. Thus, a logical next step in the development of our design and verification framework is to enable users to create their own microinteractions to add to the library. User-created microinteractions could then be checked with verification to determine whether each microinteraction satisfies design constraints required by RoVer.

In addition to user-created microinteractions, designers might also benefit from the ability to author their own social norms and task expectations against which they can check their designs, affording them more control over the interactions they wish to create. Determining the amount of control the user should have, however, is not straightforward. More control is favorable when the interaction is novel or the designer has the background and expertise necessary to define "correctness" for the designed behavior. Too much control, however, is problematic when the correct interaction is not clear to the user. Further research is needed to determine the most effective means for users to author LTL properties.

Limitations of the modeling framework also limit the social-norm properties that can be expressed in LTL, specifically the expression of probabilistic properties and properties that reason about the ordering of events among concurrently executing microinteractions. RoVer analyzes interactions nondeterministically, meaning that it must consider every *possible* interaction event without any regard as to whether certain events are likely or unlikely. Consequently, false positives often arise when RoVer searches for expectation violations. For instance, in the group-level property that states that the robot should never interrupt the human during the execution of two concurrent microinteractions, the model checker outputs any *possible* combination of states between the microinteractions that results in a violation. In future work, we plan to incorporate *probability* into the execution of microinteractions to better reason about the likelihood of property violations. We can also use probability to more accurately reason about the actions of the human in each microinteraction. With probabilistic information on human actions, verification can estimate and inform the designer about the chances that an expectation will be violated, rather than labeling a property as violated even if the likelihood of violation is small.

We would also like to explore methods for *automated repair* of property violations within interaction designs. That is, instead of simply informing designers of the violated social norms, we can automatically modify their designs to make them satisfy the given set of social norms. For example, by automating the repair of common errors, we can enable designers to focus on the higher-level, unique aspects of their designs.

Lastly, we believe it is necessary to perform a more comprehensive evaluation of RoVer, including a more careful calculation of interaction design quality. Counting the number of errors at any given point in time, as we did in the current evaluation, disregards imbalances in error severity, highly correlated errors, and the fact that removing errors often requires temporarily introducing others, as seen in the assisted condition for errors tracked during the fix phase in Figure 6 (right). Additionally, we observed the number of errors in the assisted condition to be higher than the non-assisted condition during the design phase (Figure 6, left), despite conditions being equal during this phase. This phenomenon may be a product of our small sample size and of the limitations of our error metric. We would also like to extend our user study to include experimental conditions in which users have access to simulation and verification, simulation without verification, and verification without simulation to achieve a more comprehensive and ecologically valid assessment.

CONCLUSION

In this paper, we describe a novel approach to human-robot interaction design, including visual authoring of interactions, formalizing interaction-related properties in temporal logic, and verifying that the interactions adhere to a set of social norms and task expectations. We implemented these methods in an authoring environment, RoVer, and demonstrated its capabilities and limitations in an illustrative "delivery robot" scenario. In a preliminary user study with nine participants, we assessed user experience within a single iteration of a design-interpret-fix pipeline, in which users designed interactions, interpreted social and task errors, and then addressed these errors. The study results showed that having feedback from formal verification increases designers' ability to accurately find errors within their designs and the ease of finding and understanding these errors.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation (NSF) award 1651129, an NSF Graduate Research Fellowship, and a University of Wisconsin–Madison, College of Letters & Science, Community of Graduate Research Scholars (C-GRS) fellowship. We would like to thank Dylan Glas and Takayuki Kanda for sharing interaction models from their work and Christopher Little for help in conducting the evaluation study.

REFERENCES

1. Sonya Alexandrova, Zachary Tatlock, and Maya Cakmak. 2015. RoboFlow: A flow-based visual programming language for mobile manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5537–5544.
2. Rajeev Alur, Salar Moarref, and Ufuk Topcu. 2013. Counter-strategy guided refinement of GR (1) temporal logic specifications. In *Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 26–33.
3. Sean Andrist, Bilge Mutlu, and Michael Gleicher. 2013. Conversational gaze aversion for virtual agents. In *International Workshop on Intelligent Virtual Agents (IVA)*. Springer, 249–262.
4. Sean Andrist, Tomislav Pejsa, Bilge Mutlu, and Michael Gleicher. 2012. Designing effective gaze mechanisms for virtual agents. In *ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 705–714.
5. Sean Andrist, Xiang Zhi Tan, Michael Gleicher, and Bilge Mutlu. 2014. Conversational gaze aversion for humanlike robots. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. ACM, 25–32.
6. Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.
7. J-C Baillie. 2005. Urbi: Towards a universal robotic low-level programming language. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 820–825.
8. Emilia I Barakova, Jan CC Gillesen, Bibi EBM Huskens, and Tino Lourens. 2013. End-user programming architecture facilitates the uptake of robots in social therapies. *Robotics and Autonomous Systems* 61, 7 (2013), 704–713.
9. Rafael H Bordini, Michael Fisher, and Maarten Sierhuis. 2009. Formal verification of human-robot teamwork. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. ACM, 267–268.
10. John Brooke and others. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
11. Michael Jae-Yoon Chung, Justin Huang, Leila Takayama, Tessa Lau, and Maya Cakmak. 2016. Iterative Design of a System for Programming Socially Interactive Service Robots. In *International Conference on Social Robotics (ICSR)*. Springer, 919–929.
12. 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2352–2357.
13. Clare Dixon, Matt Webster, Joe Saunders, Michael Fisher, and Kerstin Dautenhahn. 2014. "The fridge door is open"— Temporal Verification of a Robotic Assistant's Behaviours. In *Conference Towards Autonomous Robotic Systems*. Springer, 97–108.
14. Bernard Espiau, Konstantinos Kapellos, and Muriel Jourdan. 1996. Formal verification in robotics: Why and how? In *Robotics Research*. Springer, 225–236.
15. Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. 2005. Temporal logic motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020–2025.
16. Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. 2016. Human-robot interaction design using interaction composer: Eight years of lessons learned. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. IEEE Press, 303–310.
17. Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.
18. Brandon Heenan, Saul Greenberg, Setareh Aghel-Manesh, and Ehud Sharlin. 2014. Designing social greetings in human robot interaction. In *ACM Conference on Designing Interactive Systems*. ACM, 855–864.
19. Jarrett Holtz, Arjun Guha, and Joydeep Biswas. 2018. Interactive Robot Transition Repair With SMT. *arXiv preprint arXiv:1802.01706* (2018).
20. Chien-Ming Huang and Bilge Mutlu. 2012. Robot behavior toolkit: Generating effective social behaviors for robots. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. ACM, 25–32.
21. Chien-Ming Huang and Bilge Mutlu. 2013. The repertoire of robot behavior: Enabling robots to achieve interaction goals through social behavior. *Journal of Human-Robot Interaction* 2, 2 (2013), 80–102.
22. Chien-Ming Huang and Bilge Mutlu. 2014. Learning-based modeling of multimodal behaviors for humanlike robots. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ACM, 57–64.
23. Xiaowei Huang and Marta Zofia Kwiatkowska. 2017. Reasoning about Cognitive Trust in Stochastic Multiagent Systems. In *AAAI*. 3768–3774.
24. Barbara Jobstmann, Andreas Griesmayer, and Roderick Bloem. 2005. Program repair as a game. In *International Conference on Computer Aided Verification (CAV)*. Springer, 226–238.
25. Peter H Kahn, Nathan G Freier, Takayuki Kanda, Hiroshi Ishiguro, Jolina H Ruckert, Rachel L Severson, and Shaun K Kane. 2008. Design patterns for sociality in human-robot interaction. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. ACM, 97–104.

26. Adam Kendon. 1990. *Conducting interaction: Patterns of behavior in focused encounters*. Vol. 7. CUP Archive.
27. Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2008. Translating structured english to robot controllers. *Advanced Robotics* 22, 12 (2008), 1343–1359.
28. Marta Kwiatkowska. 2017. Cognitive Reasoning and Trust in Human-Robot Interactions. In *Annual Conference Theory and Applications of Models of Computation (TAMC)*. 3–11.
29. Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification (CAV)*. Springer, 585–591.
30. Min Kyung Lee, Sara Kiesler, Jodi Forlizzi, and Paul Rybski. 2012. Ripple effects of an embedded social agent: a field study of a social robot in the workplace. In *ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 695–704.
31. Wenchao Li, Lili Dworkin, and Sanjit A Seshia. 2011. Mining assumptions for synthesis. In *ACM/IEEE International Conference on Formal Methods and Models for Codesign*. IEEE Computer Society, 43–50.
32. Wenchao Li, Dorsa Sadigh, S Shankar Sastry, and Sanjit A Seshia. 2014. Synthesis for human-in-the-loop control systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 470–484.
33. Tino Lourens and Emilia Barakova. 2011. User-friendly robot environment for creation of social scenarios. In *International Work-Conference on the Interplay between Natural and Artificial Computation*. Springer, 212–221.
34. Alaeddine Mihoub, Gérard Bailly, Christian Wolf, and Frédéric Elisei. 2015. Learning multimodal behavioral models for face-to-face social interaction. *Journal on Multimodal User Interfaces* 9, 3 (2015), 195–210.
35. AJung Moon, Daniel M Troniak, Brian Gleeson, Matthew KXJ Pan, Minhua Zheng, Benjamin A Blumer, Karon MacLean, and Elizabeth A Croft. 2014. Meet me where I'm gazing: How shared attention gaze affects human-robot handover timing. In *ACM/IEEE International Conference on Human Robot Interaction (HRI)*. ACM, 334–341.
36. Bilge Mutlu. 2011. Designing embodied cues for dialog with robots. *AI Magazine* 32, 4 (2011), 17–30.
37. Bilge Mutlu and Jodi Forlizzi. 2008. Robots in organizations: The role of workflow, social, and environmental factors in human-robot interaction. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 287–294.
38. Bilge Mutlu, Takayuki Kanda, Jodi Forlizzi, Jessica Hodgins, and Hiroshi Ishiguro. 2012. Conversational gaze mechanisms for humanlike robots. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 1, 2 (2012), 12.
39. Julia Peltason and Britta Wrede. 2010. Modeling human-robot interaction based on generic interaction patterns. In *AAAI Fall Symposium: Dialog with Robots*.
40. Amir Pnueli. 1977. The temporal logic of programs. In *Annual Symposium on Foundations of Computer Science*. IEEE, 46–57.
41. 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.
42. Selma Sabanovic, Marek P Michalowski, and Reid Simmons. 2006. Robots in the wild: Observing human-robot social interaction outside the lab. In *IEEE International Workshop on Advanced Motion Control*. IEEE, 596–601.
43. Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language* (1974), 696–735.
44. Satoru Satake, Takayuki Kanda, Dylan F Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. 2009. How to approach humans?: strategies for social robots to initiate interaction. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. ACM, 109–116.
45. Allison Sauppé and Bilge Mutlu. 2014. Design patterns for exploring and prototyping human-robot interactions. In *ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 1439–1448.
46. Allison Sauppé and Bilge Mutlu. 2015. The Social Impact of a Robot Co-Worker in Industrial Settings. In *ACM/SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, New York, NY, USA, 3613–3622.
47. Alan Turing. 1949. Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*. Univ. Math. Lab., Cambridge, 67–69.
48. Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay, and Kerstin Dautenhahn. 2014. Formal verification of an autonomous personal robotic assistant. In *AAAI Spring Symposium: Formal Verification and Modeling in Human-Machine Systems*. 74–79.
49. Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay, Kerstin Dautenhahn, and Joan Saez-Pons. 2016. Toward reliable autonomous robotic assistants through formal verification: a case study. *IEEE Transactions on Human-Machine Systems* 46, 2 (2016), 186–196.
50. Jeannette M Wing. 1990. A specifier's introduction to formal methods. *Computer* 23, 9 (1990), 8–22.