Creating and Using Matrix Representations of Social Interaction

Alan R. Wagner
Georgia Institute of Technology
85 Fifth Street NW
Atlanta, GA 30308
1-404-894-9311
alan.wagner@cc.gatech.edu

ABSTRACT
This paper explores the use of an outcome matrix as a computational representation of social interaction suitable for implementation on a robot. An outcome matrix expresses the reward afforded to each interacting individual with respect to pairs of potential behaviors. We detail the use of the outcome matrix as a representation of interaction in social psychology and game theory, discuss the need for modeling the robot's interactive partner, and contribute an algorithm for creating outcome matrices from perceptual information. Experimental results explore the use of the algorithm with different types of partners and in different environments.

Categories and Subject Descriptors
I.2.9 [**Artificial Intelligence**]: Robotics – *autonomous vehicles*, *operator interfaces*

General Terms
Algorithms, Human Factors.

Keywords
Mental model, interaction, Interdependence theory.

1. INTRODUCTION
Many scientists have recently come to recognize the social aspects of intelligence [1]. In contrast to purely cognitive intelligence, which is most often described by problem solving ability and/or declarative knowledge acquisition and usage, social intellect revolves around an individual's ability to effectively understand and respond in social situations [2]. Neuroscientific evidence is beginning to emerge supporting theories of social intelligence [3]. From a roboticist's perspective, it then becomes natural to ask how this form of intelligence could play a role in the development of an artificially intelligent robot. As an initial step, one must first consider which concepts are most important to social intelligence.

Social interaction is one fundamental concept [4]. Social psychologists define *social interaction* as influence—verbal, physical, or emotional—by one individual on another [5]. If a goal of artificial intelligence is to understand, imitate, and interact with humans then researchers must develop computational representations for interaction that will allow an artificial system to: (1) use perceptual information to generate its representation for interaction; (2) represent its interactions with a variety of human partners in numerous different social environments; and (3) afford the robot guidance in selecting interactive actions.

This paper presents a representation that allows a robot to manage these challenges. A general, established, computational representation for social interaction that is not tied to specific social environments or paradigms is presented [4]. Moreover, we contribute a preliminary algorithm that allows a robot to create representations of its social interactions from direct verbal communication. Simulation results demonstrate our algorithm in several different domains and with numerous different types of partners. The purpose of this paper is to introduce the outcome matrix as an important potential representation of social interaction in artificial systems and demonstrate a method for generating outcome matrices. This paper begins by first summarizing relevant research.

HRI'09, March 11–13, 2009, La Jolla, CA, USA.

2. RELATED WORK
Representations for interaction have a long history in social psychology and game theory [4, 6]. Interdependence theory, a type of social exchange theory, is a psychological theory developed as a means for understanding and analyzing interpersonal situations and interaction [4]. The term interdependence specifies the extent to which one individual of a dyad influences the other. Interdependence theory is based on the claim that people adjust their interactive behavior in response to their perception of a social situation's pattern of rewards and costs. Thus, each choice of interactive behavior by an individual offers the possibility of specific rewards and costs—also known as outcomes—after the interaction. Interdependence theory represents interaction and social situations computationally as an outcome matrix (figure 1). An outcome matrix represents an interaction by expressing the outcomes afforded to each interacting individual with respect each pair of potential behaviors chosen by the individuals.

Game theory also explores interaction. Moreover, game theory has been described as "a bag of analytical tools" to aid one's understanding of strategic interaction [6]. As a branch of applied mathematics, game theory thus focuses on the formal consideration of strategic interactions, such as the existence of equilibriums and economic applications [6]. Game theory and

interdependence theory both use the outcome matrix to represent interaction [4, 6]. Game theory, however, is limited by several assumptions, namely: both individuals are assumed to be outcome maximizing; to have complete knowledge of the game including the numbers and types of individuals and each individual's payoffs; and each individual's payoffs are assumed to be fixed throughout the game. Because it assumes that individuals are outcome maximizing, game theory can be used to determine which actions are optimal and will result in an equilibrium of outcome. Interdependence theory does not make these assumptions and does not lend itself to analysis by equilibrium of outcomes. Numerous researchers have used game theory to control the behavior of artificial agents in multi-agent environments (e.g. [7]). We do not know of any that have used interdependence theory. The use of interdependence theory is a crucial difference between this work and previous investigations by other researchers using game theory to control the social behavior of an agent.

Example Outcome Matrices

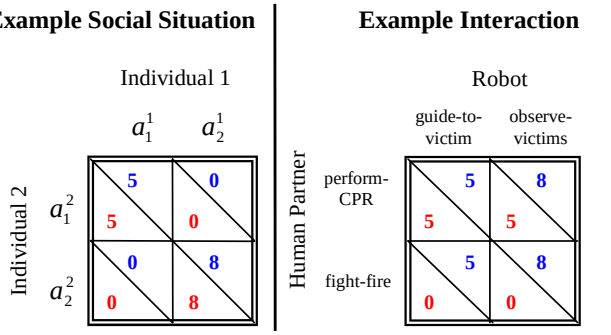

Figure 1. Example outcome matrices are depicted above. The right hand side depicts an outcome matrix representing an actual interaction encountered by the robot in the experiments. The left hand side depicts a social situation. Social situations abstractly represent interactions. A dependent situation is depicted on the left and an independent situation is depicted on the right.

This work differs from much of current human-robot interaction research in that our work investigates theoretical aspects of human-robot interaction. Typically, HRI research explores the mechanisms for interaction, such as gaze following, smooth pursuit, face detection, and affect characterization [8].

3. REPRESENTING INTERACTION

The outcome matrix is a standard computational representation for interaction [4]. It is composed of information about the individuals interacting, including their identity, the interactive actions they are deliberating over, and scalar outcome values representing the reward minus the cost, or the outcomes, for each individual. Thus, an outcome matrix explicitly represents information that is critical to interaction. Typically, the identity of the interacting individuals is listed along the dimensions of the matrix. Figure 1 depicts an interaction involving two individuals. In this article the term individual is used to indicate either a human or a social robot or agent. We will focus on interaction involving two individuals—dyadic interaction. An outcome matrix can, however, represent interaction involving more than two individuals. The rows and columns of the matrix consist of a list of actions available to each individual during the interaction. Finally, a scalar outcome is associated with each action pair for each individual. Outcomes represent unitless changes in the robot, agent, or human's utility. Thus, for example, an outcome of zero reflects the fact that no change in the individual's utility will result from the mutual selection of that action pair.

Because outcome matrices are computational representations, it is possible to describe them formally. Doing so allows for powerful and general descriptions of interaction. The notation presented here draws heavily from game theory [6]. A representation of interaction consists of 1) a finite set N of interacting individuals; 2) for each individual $i \in N$ a nonempty set A^i of actions; 3) the utility obtained by each individual for each combination of actions that could have been selected [4]. Let $a_j^i \in A^i$ be an arbitrary action j from individual i's set of actions. Let $\left(a_j^1, \ldots, a_k^N\right)$ denote a combination of actions, one for each individual, and let u^i denote individual i's utility function: $u^i\left(a_j^1, \ldots, a_k^N\right) \rightarrow \Re$ is the utility received by individual i if the individuals choose the actions $\left(a_j^1, \ldots, a_k^N\right)$. The term O is used to denote an outcome matrix. The superscript $-i$ is used to express individual i's partner. Thus, for example, A^i denotes the action set of individual i and A^{-i} denotes the action set of individual i's interactive partner.

3.1 Representing Social Situations

The term interaction describes a discrete event in which two or more individuals select interactive behaviors as part of a social situation or social environment. Interaction has been defined as influence—verbal, physical, or emotional—by one individual on another [5]. The term situation has several definitions. The most apropos for this work is "a particular set of circumstances existing in a particular place or at a particular time [9]." A social situation, then, characterizes the environmental factors, outside of the individuals themselves, which influence interactive behavior. A social situation is abstract, describing the general pattern of outcome values in an interaction. An interaction, on the other hand, is concrete with respect to the two or more individuals and the social actions available to each individual. For example, the prisoner's dilemma describes a particular type of social situation. As such, it can, and has been, instantiated in numerous different particular social environments ranging from bank robberies to the trenches of World War I [10]. Interdependence theorists state that interaction is a function of the individuals interacting and of the social situation [4]. A dependent situation, for example, is a social situation in which each partner's outcome depends on the other partner's action (Figure 1 left). An independent situation, on the other hand, is a social situation in which each partner's outcome does not depend on the partner's action (Figure 1 right). Although a social situation may not afford interaction, all interactions occur within some social situation. Interdependence theory represents social situations involving interpersonal interaction as outcome matrices (see figure 1 for a graphical depiction of the difference).

In previous work, we presented a situation analysis algorithm that calculated characteristics of the social situation or interaction (such as interdependence) when presented with an outcome matrix [11]. The interdependence space is a four dimensional space which maps the location of all interpersonal social situations [4]. A matrix's location in interdependence space provides important information relating to the interaction. Information such as the level of interdependence can indicate to the robot the sensitivity of the partner's outcomes to the robot's actions. In assistive therapy domains, for example, the outcomes of the patient may rely on the action selection of the robot. Our results showed that by analyzing the interaction, the robot could better select interactive actions. Thus, using an outcome matrix as a representation of interaction can benefit the robot in terms of selecting the best action.

A computational representation for interaction should afford the robot guidance in selecting interactive actions. Outcome matrices afford several simple action selection strategies. The most obvious method is to choose the action that maximizes the robot's outcome. This strategy is termed *max_own*. An individual's use of the *max_own* strategy results in egoistic interactive behavior. Alternatively, the robot may select the action that maximizes its partner's outcome, a strategy termed *max_other*. An individual's use of the *max_other* strategy results in altruistic behavior. Yet another action selection strategy is for the robot to select the action that maximizes the sum of its and its partner's outcome. The use of this strategy results in a cooperative style of behavior. Outcome matrices afford many other simple action selection strategies (see [11] for other examples). Previous work in robotics and planning which employ a similar representation for controlling a robot or agent generally focus on a *max_own*, game theoretic action selection strategy. To the best of our knowledge action selection strategies such as *max_other* have not been investigated.

3.2 Partner Modeling

Several researchers have explored how humans develop mental models of robots (e.g. [12]). A mental model is a term used to describe a person's concept of how something in the world works [13]. We use the term partner model (denoted m^{-i}) to describe a robot's mental model of its interactive human partner. We use the term self model (denoted m^{i}) to describe the robot's mental model of itself. Again, the superscript *-i* is used to express individual *i*'s partner [6].

An exploration of how a robot could model its human partner should begin by considering what information will be collected in this model. Our partner model contains three types of information: 1) a set of partner features $\left(f_1^{-i},\ldots,f_n^{-i}\right)$; 2) an action model, A^{-i}; and 3) a utility function u^{-i}. We use the notation $m^{-i}.A^{-i}$ and $m^{-i}.u^{-i}$ to denote the action model and utility function within a partner model.

Partner features are used for partner recognition. Partner features allow the robot to recognize the partner in subsequent interactions. The partner's action model contains a list of actions available to that individual. The partner's utility function includes information about the outcomes obtained by the partner when the robot and the partner select a pair of actions. The information encompassed within our partner models does not represent the final word on what types of information should be included in such models. Information about the partner's beliefs, knowledge, personality, etc. could conceivable be included in these models.

Outcome Matrix Creation Algorithm

Input: Self Model m^{i}, Partner Model m^{-i}.

Output: Outcome matrix *O*.

1. Create empty outcome matrix *O*
2. **Set** $O.partner = x\left(m^{-i}.features\right)$, $O.robot$ = "`robot`", $O.columns = m^{i}.A^{i}$, $O.rows = m^{-i}.A^{-i}$
2. **For** each pair $\left(a_j^{i},a_k^{-i}\right)$ in all rows and columns
3. $\quad O^{i}\left(a_j^{i},a_k^{-i}\right) \leftarrow m^{i}.u^{i}\left(a_j^{i},a_k^{-i}\right)$,
4. $\quad O^{-i}\left(a_j^{i},a_k^{-i}\right) \leftarrow m^{-i}.u^{-i}\left(a_j^{i},a_k^{-i}\right)$
5. **Return** *O*

Interact-and-update Algorithm

Input: partner features $f_1^{-i},\ldots,f_n^{-i}$, situation features $e_1,\ldots,e_n$

Pre-interaction

1. **Set** $m^{i} = y(e_1,\ldots,e_n)$, $m^{-i} = z\left(f_1^{-i},\ldots,f_n^{-i}\right)$
2. *OutcomeMatrixCreationAlgorithm* $\left(m^{i},m^{-i}\right)$
3. **Set** $a^{i} = \texttt{max\_own}(O^{i})$, ${}^{*}o^{i} = O^{i}\left(a^{i},a^{-i}\right)$, ${}^{*}a^{-i} = \texttt{max\_own}(O^{-i})$, ${}^{*}o^{-i} = O^{-i}\left(a^{i},a^{-i}\right)$

Interact

4. **Perform** a^{i}

Update

5. **Perceive** value a^{-i}, o^{i}, o^{-i}
6. **If** $a^{-i} \neq {}^{*}a^{-i}$
7. $\quad$ **update** $m^{-i}.A^{-i} = a^{-i}$, $m^{-i}.u\left(a^{-i}\right) = o^{-i}$
8. **else if** $o^{-i} \neq {}^{*}o^{-i}$
9. $\quad$ **update** $m^{-i}.u\left(a^{i},a^{-i}\right) = o^{-i}$
10. **If** $o^{i} \neq {}^{*}o^{i}$ **then update** $m^{i}.u\left(a^{-i},a^{i}\right) = o^{i}$
12. **for** all a^{-i} in m^{-i}
13. $\quad$ **if** $p\left(a^{-i}\right) < k$ **then** delete a^{-i}

Figure 2. Algorithms for creating and using outcome matrices. The algorithm successively updates the partner models achieving greater outcome matrix creation accuracy. The function *x* maps partner features to a partner ID, *y* maps situation features to the robot's self model, and *z* maps partner features to a partner model.

The self model also contains an action model and a utility function. The action model contains a list of actions available to the robot. Similarly the robot's utility function includes information about the robot's outcomes.

3.3 From Interaction to Outcome Matrix

The proposed representation has the following elements which must be filled in: the identity of the individuals interacting, the actions for each individual, and the outcomes available for each pair of actions and each individual. The robot must fill in this information in the order listed because, for example, the identity of the robot's partner could influence which actions are available to the partner.

We have thus sketched the outline of an algorithm for creating outcome matrices from the robot's model of itself and its partner. Figure 2 (top) depicts the algorithm. The algorithm takes as input the self model and the partner model and produces an outcome matrix as output. The first step of the algorithm creates an empty outcome matrix. The second step of the algorithm sets the partner's ID and both the robot's and the partner's actions. This step uses the function *x* to map perceptual features to a unique label or ID. ID creation provides a means of attaching the perception of an individual to what is learned from interacting with that individual. In theory, any method that provides a unique ID from perceptual features should work in this algorithm. We have not, however, explored this claim experimentally. Finally, for each pair of actions in the action models, we use each individual's utility function (u^i and u^{-i}) to assign an outcome for the pair of actions.

It should be apparent that the Outcome Matrix Creation algorithm simply fills in the matrix with missing information. Moreover, the accuracy of the outcome matrices created by the algorithm depends entirely on the accuracy of the information contained in the self and partner models. This begs the question, where does the information for the models come from? The interact-and-update algorithm serves this purpose.

The Interact-and-update algorithm uses information learned during an interaction to revise its partner and robot models. Norman notes that humans continually revise their mental models with additional interaction [13]. Our algorithm employs a similar strategy, updating its representation of its human partner with each additional interaction. The algorithm works by first predicting the action the partner will select and the outcomes the robot and the partner will obtain. Then, in the update phase, the algorithm adjusts the partner model.

Figure 2 (bottom) depicts the algorithm. For clarity, the algorithm is divided into three phases: pre-interaction, interact, and update. During the pre-interaction phase the robot selects models, calls the Outcome Matrix Creation algorithm constructing the matrix, selects an action and sets its predictions for the interaction. During the interact phase the robot performs the action. Finally, in the update phase, the robot adjusts its partner model to account for the actual outcome obtained and actions performed.

The interact-and-update algorithm takes as input the partner features and situation features. Partner features are used to recognize and/or characterize the robot's interactive partner. Similarly, situation features are perceptual features used to characterize the environment. The algorithm begins by using the situation features to retrieve a self model. The function *y* maps situation features to subsets of the robot's action set and utility values. Thus the robot's model of itself depends on the type of environment in which it is interacting. The partner's features are used to retrieve a model of the partner. The function *z* selects the partner model from a database of partner models with the greatest number of equivalent features. During initialization, the partner model database is seeded with a model of the robot. Thus the database always contains at least one model. During the interaction phase the robot performs the action. During the update phase of the algorithm, the robot first perceives the action performed by its partner and the outcome both it and the partner obtain. Next, if the partner action does not match the prediction, then the action is added to the model if it did not exist and the outcome for the action pair is updated. If, on the other hand, the robot predicted the correct action but did not predict the correct outcome then the outcome is updated in the partner model. Next, if the outcome the robot obtained differed from the robot's prediction then the robot updates its own model to reflect the received outcome. Finally actions and associated outcome values which have less than *k* probability of usage based on previous experience are removed. This prevents the model from becoming filled with rarely used actions. Successive matrices can be created by looping to line 3.

Line 7 updates the outcome value to match the perceived outcome value when an unexpected action is encountered. If the action is unknown, then robot does not yet have information about the outcome values of all action pairs. In this case it must make an assumption as to their value. As currently presented the algorithm assigns a single outcome value to all action pairs irrespective of the robot's action. This assignment results in what we call an **action independence assumption**. The robot is assuming that, for the unknown action pairs, the partner receives the same outcome regardless of the robot's choice of action. Alternatively, we could have assumed that for unknown action pairs the human receives the same outcome as the robot. Either of these assumptions is equally valid as the values simply serve as placeholders and allude to the robot's current ignorance of the human's action preference.

Intuitively the algorithm directly updates the outcome values and actions. Hence the algorithm is susceptible to noise. Machine learning algorithms could be used to reduce this susceptibility. Ng, for example, describes inverse reinforcement learning as the problem of learning a task's reward function. He has also developed techniques for learning from a teacher [14]. We have begun to explore the use of clustering techniques to aid in partner model learning and our development of methods for matrix creation is ongoing research. Numerous game theoretic methods, such as Bayesian games, also exist for handling uncertainty [6]. Unfortunately space limitations prevent a detailed examination of error reduction techniques.

3.4 Determining Model Accuracy

The preceding discussion raises an important question: how do we measure partner model accuracy? For example, given a particular human partner with action set $m^{-i}.A^{-i}$ and utility function $m^{-i}.u^{-i}$, how close is the robot's partner model m^{-i} to the actual model ${}^{*}m^{-i}$? We address this problem by viewing action models and utility functions as sets. The action model is a set of

actions and a utility function is a set of triplets contains the action of each individual and a utility value. We can then do set comparisons to determine the accuracy of the robot's partner model m^{-i}.

Two types of error are possible. Type I error (false positive) occurs if an action or utility is added to the robot's partner model (m^{-i}) which is not in the actual model ($^{*}m^{-i}$). Type II error (false negative) occurs if an action or utility in the actual model ($^{*}m^{-i}$) is not included in robot's partner model (m^{-i}). Both of these types of error must be included in a measure of action model or utility function accuracy. Moreover, a utility function value was not considered present in the model if the value differed from the actual value by an amount greater than one. To determine Type I error we calculate the number of actions or utilities in m^{-i} which are not in $^{*}m^{-i}$ as a percent of the number of actions or utilities in m^{-i}. Thus, $\frac{\left|m^{-i} - {}^{*}m^{-i}\right|}{\left|m^{-i}\right|}$, is the number of actions in the robot's model that are not in the actual model divided by the number of actions in the robot's model. Model accuracy, as oppose to inaccuracy, is calculated as $1 - \frac{\left|m^{-i} - {}^{*}m^{-i}\right|}{\left|m^{-i}\right|}$. Type II error can be calculated as the number of actions or utilities in both m^{-i} and $^{*}m^{-i}$ as a percent of $^{*}m^{-i}$. Thus, $\frac{\left|{}^{*}m^{-i} \cap m^{-i}\right|}{\left|{}^{*}m^{-i}\right|}$, is the number of actions in both models divided by the number of actions in the actual model. Finally, the two types of errors are averaged in the equation,

$$d = 0.5\left(1 - \frac{\left|m^{-i} - {}^{*}m^{-i}\right|}{\left|m^{-i}\right|}\right) + 0.5\left(\frac{\left|{}^{*}m^{-i} \cap m^{-i}\right|}{\left|{}^{*}m^{-i}\right|}\right) \quad (1)$$

to create *d*, an overall measure of model accuracy for either an action model (d^{a}) or a utility function (d^{u}). To determine overall model accuracy we average the error from both components of the partner model,

$$d^{-i} = \frac{d^{a} + d^{u}}{2} \quad (2)$$

4. EXPERIMENTAL METHODOLOGY

Three components make up a human robot interaction—the robot, the human, and the environment. This research explores the robot's social behavior. Thus, we must control for the behavior of the human and the environment. In the section that follows we present a method to control for the human's behavior.

4.1 Controlling Human Behavior

Evaluating the robot's ability to model its interactive partner requires control over its interactive partner's behavior. In essence, we need the human partner to act in a predefined manner. Laboratory experiments involving controlled human behavior are standard in many psychology experiments [5]. These experiments typically require that the experimenter's confederate follow a predefined script often acting the part of a fellow subject. This script explains how the person should behave in all of the situations he or she will face in the experiment. In much the same way, our evaluation of the robot's ability to model its partner requires that the human partner act in a scripted manner. We use the term *actor script* to describe a predefined set of interactive instructions that the human will follow when interacting with the robot. Actor scripts are used in the experiments presented in this work.

An actor script is created by first delineating the situations that the human-robot dyad will encounter. Once the situations have been determined, the human's actions can be dictated in several different ways. One method is to assign the human a social character and to then select actions in accordance with the assigned character. For example, if the human is assigned the social character of altruist then the human will select the outcome matrix action that most favors the robot's outcomes. To complete the actor script, actions are determined for each interaction, possibly being contingent on the robot's prior behavior, and a list or flowchart is created that the human follows when interacting with the robot.

In our experiments, the robot's human partner was assigned a predetermined list of perceptual features that were used by the robot for identification or as evidence of the partner's type. Moreover, the human's actions were scripted. In other words, the human selected a predefined series of actions that were contingent on the robot's prior actions and the experimental condition. Because the experiments controlled for the human's features and actions, all experiments could be conducted by a single human partner. Still, a pilot study involving three different humans (a 20 year old American woman, a 20 year old Indian-American woman, and a 33 year old American male) was conducted to rule out the possibility of experimenter bias. The study compared the outcome matrices created by the robot when interacting in the same environment with each different human. No difference was found.

4.2 Controlling the Environment

The use of predefined social situations as a method for exploring human interactive behavior has become a common methodological tool for psychologists, economists and neuroscientists [15]. We control the environment by using predefined, arbitrary social situations. These social situations determined the pattern of outcome values received by the robot and the human. Each partner type was assigned an arbitrary preference over his/her actions. These actions were assigned utilities in a top down fashion. For instance, the police officer in the search and rescue environment might be assigned a utility function of `limit-access`=1, `direct-traffic`=0, `search-for-victim`=-1. Keep in mind that our goal is to determine if, and how well the robot can learn a model of its partner. Hence it suffices to create an arbitrary utility function for

the robot's partner to gauge if the robot can learn the model. We tested also the algorithm on dependent and independent situations (figure 1).

4.3 Experimental Setup

We conducted both simulation experiments and real world experiments to test the proposed algorithm. Our simulation experiments utilized USARSim, a collection of robot models, tools, and environments for developing and testing search and rescue algorithms in high-fidelity simulations.

We created five different environments in USASim to test the generality of our algorithm. The household environment modeled a small studio apartment and contained couches, a bed, a television, etc. (Figure 3 top left). The museum environment modeled a small art and sculpture gallery and contained paintings, statues, and exhibits (Figure 3 top right). The assistive environment modeled a rehabilitation suite (Figure 3 bottom left). The prison environment modeled a small prison and contained weapons, visiting areas, and a guard station (Figure 3 bottom right). The search and rescue environment (not shown) modeled a disaster area and contained debris fields, small fires, victims, and a triage area.

Figure 3. Four of the USARSim environments used in the simulation experiments.

The USARSim model of the Pioneer DX robot was used in all experiments. The robot had both a camera and a laser range finder. The robot used speech synthesis to communicate questions and information to the human partner. Speech recognition translated the spoken information provided by the human. Microsoft's Speech SDK provided the speech synthesis and recognition capabilities.

Table 1. Partner features and feature values

Feature Name	Values
Gender	<man,woman>
Height	<tall,medium,short>
Age	<young,middling,old>
Weight	<heavy,average,thin>
Hair color	<blonde,black,brown,red>
Eye color	<blue,green,brown>
Tool 1	<axe,gun,stethoscope,baseball-cap>
Tool 2	<oxygen-mask,badge,medical-kit,backpack>

The real world environment was 5x5 meter maze in the shape of a cross (Figure 6). One branch of the cross contained victims from a notional disaster (babies) and the other branch contained hazard items such as a biohazard.

5. EXPERIMENTS

Simulation experiments were conducted to gather accuracy data. Real robot experiments were conducted to demonstrate the feasibility of this approach on real-world situated embodied systems. We will describe the simulation experiments first.

We hypothesized that continued interaction would result in improved partner model accuracy—both accuracy of the partner's action model and of the partner's utility function. Two simulation experiments were conducted to test this hypothesis. The first experiment examined interaction with a single partner type (doctor) in each different environment. Each environment resulted in a different action model and utility function for the robot. In the search and rescue environment, for example, the robot helped to locate trapped victims. In the museum environment, on the other hand, the robot acted as a security guard patrolling the museum. A second simulation experiment explored interaction in a single environment (search and rescue) with four different types of individuals: policeman, firefighter, doctor, and citizen. Each different type of partner had a unique action model and utility function. For example, the firefighter preferred to search for hazards and the doctor preferred to save victims.

Table 2. A list of actions for each type of partner

Partner Type	Actions
Robot	`SearchFor-x, Observe-x, Light-x, GuideTo-x`
Policeman	`limit-access, direct-traffic, search-for-victim`
Firefighter	`remove-toxic-material, fight-fire, rescue-victim, move-debris`
Doctor	`startIV, intobate, peformCPR`
Citizen	`run, cry, scream`
Random	Any of the above non-robot actions.

Each human partner type possessed particular values for all of the partner features (Table 1 and Table 2). The values for Tool 1 and Tool 2 were type specific the other values were selected at random. For example, a firefighter might have the following partner features: woman, short, young, thin, red, green, axe, oxygen-mask. Action models consisted of three or four actions and were also type specific. Similarly, utility values were action and type specific. Table 2 depicts the action models for each type of partner and the robot. Ground truth consisted of predefined sets of actions and outcome values for a specific partner type. For example, a citizen partner was produced by 1) randomly selecting the values for the partner features (except tools which are set to baseball-cap and backpack for this type) 2) setting the action model to that from Table 2 for citizen creating and 3) creating arbitrary utility values for the utility function.

Both simulation experiments involved 20 interactions with a partner. Prior to interacting, the robot used OpenCV to detect objects in the environment and create the situation features. Next the robot used synthesized speech and speech recognition to query the partner for their features. Once the robot had gathered the information necessary to run the Interact-and-update algorithm, the algorithm was run, creating an outcome matrix and then selecting an action from the matrix. Both the robot and its human partner performed actions in simulation. The action performed by the human was dictated by the actor script. If the situation was independent then the robot received the outcome regardless of the

action selected by the human. If the situation was dependent, the robot's outcome depended on the human's action. Finally the robot queries the human for the action it performed and outcome received.

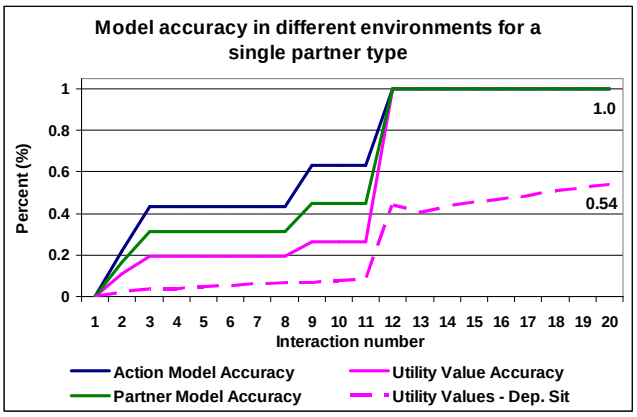

Figure 4. The graph depicts the results from the first simulation experiment involving different environments. The results show that model accuracy increases with continued interaction, eventually matching the ground truth.

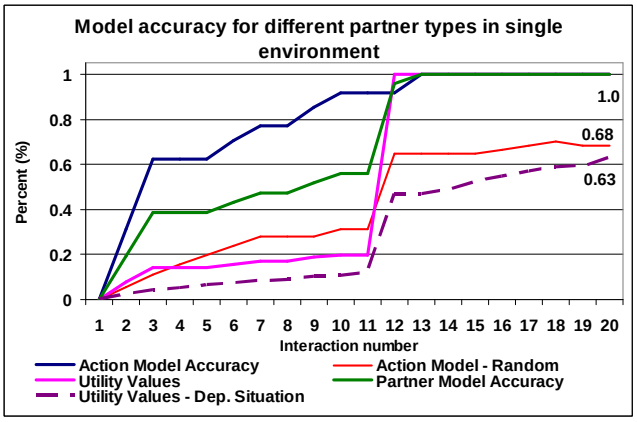

Figure 5. The graph depicts the results from the second simulation experiment involving different partner types. The results again show that model accuracy increases with continued interaction, eventually matching the ground truth.

During the experiment, we recorded the robot's model of its partner after each interaction. Equation (1) from section 3.2 was used to calculate the accuracy with respect to the individual components of the partner model. Equation (2) was used to calculate the overall accuracy of the partner model. The independent variable in the first experiment was the type of environment the robot encountered. The independent variable in the second experiment was the type of partner the robot encountered. The dependent variable for both simulation experiments was model accuracy. Figure 4 depicts the results for the first simulation experiment. The graph shows that with continued interaction the accuracy of the action model, utility function, and partner model increase, eventually matching the ground truth. After the eleventh interaction, the accuracy of all models increases dramatically. This is because the algorithm purges the models of seldom used actions and utilities reducing Type I error. In the dependent situation we see that accuracy of the utility values only reaches 64% after 20 interactions. This is because the dependent situation violates the action independence assumption discussed in section 3.3. Although less accurate, the partner model in this case still contains all of the information experienced during interaction with partner. We have found that machine learning techniques can improve this result.

Figure 5 depicts the results for the second simulation. Again the graph shows that the accuracy of all models increases with continued interaction, eventually matching the ground truth. Violating the action independence assumption again results in decreased utility accuracy (63 percent). A random partner type is also included for comparison. The random partner selected any action available to any partner type at random. The graph only depicts action model accuracy for the random partner type. An accuracy of 68 percent is achieved for the random partner type.

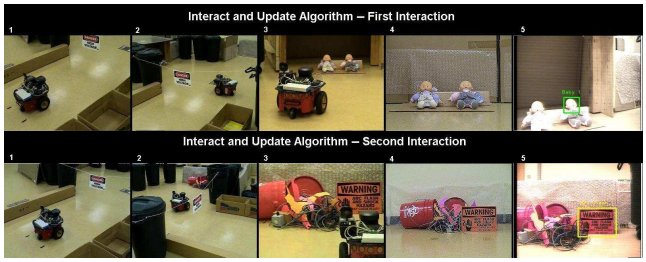

Figure 6. Photos from the robot experiment. The robot initially moves to observe the victim. After learning the model of its partner the robot moves to observe the hazard. The leftmost four photos depict the robot as it moves through the maze and selects actions. The fifth photo depicts video that the robot sends to its human partner.

A demonstration experiment involving a real robot was also conducted. In this experiment the robot was tasked with assisting in a notional search and rescue environment. The robot was capable of two actions: 1) moving to and observing a victim and 2) moving to and observing a hazard. The robot interacted with a person acting as a firefighter. The firefighter was also capable of two actions: containing the hazard or rescuing the victims. The firefighter arbitrarily preferred to contain hazards. The robot received more outcome if the victims survived. The victims survived only if the robot and the firefighter work together observing and containing the hazard or rescuing the victims (Figure 1 left side depicts the outcome matrix for this situation). Initially the robot has no knowledge of the utility functions or a model of its partner. The robot therefore sets its partner model to the robot's self model. In other words the robot assumes that its unknown partner has the same actions and preferences as it does. During the first interaction the robot moves to observe the victims. After the interaction, the robot receives feedback relating to the partner's choice of action and outcomes. It updates its partner model accordingly and during the next interaction it correctly moves to observe the hazard. The results demonstrate the potential feasibility of this approach on a robotic platform. In depth experimentation on a fielded system is an area of future work. Figure 6 shows the robot moving to observe the victim in the first interaction and the hazard in the second interaction.

6. SUMMARY AND CONCLUSIONS

This paper has introduced a computational representation for interactions and social situations suitable for implementation on a robot or software agent. We have discussed the composition of

this representation, its ability to represent both interactions and social situations, and formal operations related to the representation. Moreover, we have presented a preliminary algorithm for the creation of the representation.

The algorithm we present assumes perceptual competencies which are difficult to achieve. It assumes that the robot can perceive 1) the partner's action, 2) the partner's outcome value, and 3) the outcome obtained by the robot itself. These assumptions may limit the current applicability of the algorithm. Nonetheless, as demonstrated by the experiments, the perceptual limitations of this algorithm can be overcome. Moreover, activity recognition and affect detection are current areas of active research [16, 17]. Finally, it is important that the HRI community recognize the importance of activity recognition and state detection. This research provides a theoretical motivation for these research topics. It may well be that the challenge of recognizing how a robot's behavior has impacted the humans interacting with it is a critical question facing the HRI community.

We have also assumed that the robot knows what actions are available to it. We believe that this is a reasonable assumption. We have not assumed that the robot has accurate knowledge of the outcomes values resulting from the selection of an action pair. We have simply assigned arbitrary initial values for the outcomes and then the robot learns the true values through interactive experience with the partner.

Although our results show that interactive experience creates increasingly accurate partner models, the actions and utilities of the robot's partner were static, did not change, and contained no noise. Because the models were static they could be modeled. Alternatively, as demonstrated in the random partner type, the partner could have continually selected random actions or received random utilities. Clearly in this case less can be learned about the partner. In a sense, the robot cannot know what to expect next from its partner. In normal interpersonal interaction there are times when humans randomize their interactive actions, such as in some competitive games. This algorithm will have limited success in these situations. Noise in the form of inaccurate perception of the human's outcome values and actions is another potential challenge. Fortunately, game theory provides numerous tools for managing outcome uncertainty [6]. Moreover, our own results have demonstrated that outcome matrices degrade gracefully with increased error [18]. Future work will employ machine learning techniques to reduce overfitting.

Near-term practical applications of this work would likely focus on environments where the outcomes of the robot's partner are readily available. In assistive therapy environments, for example, the robot could ask the patient if an exercise was causing pain. An entertainment robot, on the other hand, might gauge user outcome in terms of amount of time spent interacting with the robot. Applications in areas such as autism are more difficult because the nature of the disease may limit the human's outcome expression capabilities.

Neuroscientists have shown that humans actively model their interactive partners [15]. Certainly the interpersonal mental models maintained by humans are more complex and rich than the models used here. Our purpose is not to claim that the partner models discussed here are the same as those formulated by humans, but rather to explore what minimal modeling of its interactive partner a robot must perform in order to interact successfully with the partner and to present a method for achieving this modeling. Future work may add additional complexity and richness to the partner model. We firmly believe that accurate modeling of one's interactive partner is an important component of social intelligence and hence a critical skill for a robot to successfully operate in a dynamic social environment.

7. ACKNOWLEDGMENTS

The author would like to thank Zsolt Kira for his comments.

8. REFERENCES

[1] R. W. Byrne and A. Whiten, "Machiavellian intelligence," in *Machiavellian Intelligence II: Extensions and Evaluations*, A. Whiten and R. W. Byrne, Eds. Cambridge: Cambridge University Press, 1997, pp. 1-23.

[2] N. K. Humphrey, "The social function of intellect," in *Growing Points in Ethology*, P. P. G. Bateson and R. A. Hinde, Eds., 1976, pp. 303-317.

[3] R. Bar-On, D. Tranel, N. L. Denburg, and A. Bechara, "Exploring the neurological substrate of emotional and social intelligence," *Brain*, vol. 126, pp. 1790-1800, 2003.

[4] H. H. Kelley and J. W. Thibaut, *Interpersonal Relations: A Theory of Interdependence*. New York, NY: John Wiley & Sons, 1978.

[5] D. O. Sears, L. A. Peplau, and S. E. Taylor, *Social Psychology*. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[6] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA: MIT Press., 1994.

[7] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun, "Game Theoretic Control for Robot Teams," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2005.

[8] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, pp. 143-166, 2003.

[9] Situation, in *Encarta World English Dictionary, North American Edition*, 2007.

[10] R. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.

[11] A. R. Wagner and R. C. Arkin, "Representing and analyzing social situations for human-robot interaction," *Interaction Studies*, vol. 10, 2008.

[12] A. Powers and S. Kiesler, "The advisor robot: tracing people's mental model from a robot's physical attributes," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. Salt Lake City, UT, USA, 2006.

[13] D. Norman, "Some Observations on Mental Models," in *Mental Models*, D. Gentner and A. Stevens, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[14] P. Abbeel and A. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning," in *International Conference on Machine Learning*. Banff, Canada, 2004.

[15] J. K. Rilling, A. G. Sanfey, J. A. Aronson, L. E. Nystrom, and J. D. Cohen, "The neural correlates of theory of mind within interpersonal interactions," *NeuroImage*, vol. 22, pp. 1694-1703, 2004.

[16] R. Picard, *Affective Computing*. Cambridge, MA: The MIT Press, 2000.

[17] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hähnel, "Inferring activities from interactions with objects," *IEEE Pervasive Computing*, pp. 50–57, 2004.

[18] A. R. Wagner, "A Representation for Interaction," in *Proceedings of the ICRA 2008 Workshop: Social Interaction with Intelligent Indoor Robots (SI3R)*. Pasadena, CA, USA, 2008.