Modeling Human-Like Decision Making using Probabilistic Programming

Maciej Olejnik[1[0000−0002−6996−230X]] and Marta Kwiatkowska[1[0000−0002−3496−4735]]

Univesity of Oxford, Oxford, UK
{maciej.olejnik,marta.kwiatkowska}@cs.ox.ac.uk

Abstract. Effective collaborations between humans and machines necessitate the modelling of human cognitive processes and complex social attitudes such as trust, guilt or shame. To that end, we propose cognitive stochastic multiplayer games, a novel parametric framework for multi-agent decision making, which aims to capture human motivation through mental, as well as physical, goals. The framework enables expression of cognitive notions such as trust in terms of beliefs, whose dynamics is affected by agent's observation of interactions and own preferences. We generalise utility functions to incorporate an agent's, as well as other agents', emotions and take into account their preference over different goals. Agents are modelled as soft expected utility maximisers, which allows us to capture the full range of rationality, from imperfections characteristic of human rationality, to fully rational robots. Our framework supports Bayesian learning, simulation and prediction, and has been implemented in the probabilistic programming language WebPPL. We evaluate the framework's prediction on a number of synthetic case studies, demonstrating that mental reasoning can explain experimentally-observed human behaviour that standard, equilibria-based analysis often overlooks.

Keywords: Human decision-making · Cognitive Modelling · Trust · probabilistic programming

1 Introduction

The current century has seen technological progress on a scale never experienced before. Watches pack more computing power than many desktop computers from two decades ago, cars operate autonomously and are safer than human drivers, robots perform backflips and know how to dance and AI agents write articles and beat us easily in most games. It now seems a matter of when, not if, autonomous robots become fully-fledged members of our society. Allowing technology into our personal lives promises a great deal of benefits in almost every aspect of it (and plenty of dangers too, but we focus on the desirable aspects). However, fruitful cooperation between humans and robots will only be possible if the machines behave as we expect them to; for that, it is necessary that they understand social norms and human motivations. Therefore, a computational model of human cognition and decision making is needed that would allow one to formally define unobservable social concepts such as trust, guilt, regret, and many others. Those inspired by recent successes of deep learning may argue that robots can teach themselves how to navigate the social context and an explicit formulation is not needed. However, AI systems have so far only been successful in a particular subset of applications and dependent on availability of large data sets. Others will claim millions of years of evolution cannot be bypassed as easily and a model of human cognition should be designed by experts in the field.

We argue for a combined approach, in which knowledge of area experts is utilised to construct a parameterised model, which can subsequently be refined based on available behavioural data, using Bayesian methods. To this end, we propose cognitive stochastic multiplayer games, a novel parametric framework for multi-agent decision making. Unlike robots and other artificial agents, every human has a unique personality, preferences and desires. Therefore, an accurate model of human-decision making must allow for agent variability. Our formalism achieves it by equipping each player with unique characteristics, which can, along with belief, be automatically inferred from their behaviour. We draw inspiration from theory of mind [14] to truthfully represent what an agent knows, and what they do not know. The model reflects research findings from decision-making theory and is based on expected utility maximisation principle. This may seem controversial, as numerous experiments have shown apparent deviations from this postulate. However, a key contribution of our framework is an alternative way of computing utility of agents, one that captures their feelings as well as physical rewards. Our framework supports Bayesian learning, simulation and prediction, and is implemented in the probabilistic programming language WebPPL [8]. That allows us to quantitatively evaluate it on a number of case studies and show that it accounts for experimentally-observed human behaviour that standard game-theoretic analysis fails to predict.

2 Related Work

Traditionally, game theory has studied interactions of rational decision-makers. However, many standard assumptions, such as rationality, common knowledge and complete information, do not apply to humans. As a result, many authors have tried to adapt game-theoretic techniques to capture intricacies of homo sapiens. The most prominent example are *psychological games* [7], which extend a standard concept of a game by assuming that utility of agents depends not only on outcomes, but also on beliefs that agents hold about the future behaviour of their opponents. This allows one to model how emotions of players influence their behaviour, with a restriction that such emotions must be expressible in terms of one's initial (possibly nested) belief. However, psychological games, while admitting solution methods based on standard game-based techniques such as backward induction and equilibria computation, assume fixed payoff structure, do not support inference of beliefs from data or belief updating, and have no associated software implementation.

Dynamic psychological games [1] address some of the limitations of the original proposal by including a more robust belief structure and supporting non-equilibrium analysis. However, a hierarchy of deeply nested beliefs is problematic from the computational point of view. Moreover, the model ignores uncertainty and imperfect information that humans face when making choices, and it does not support learning. We approach the problem with implementation in mind and choose simple, efficiently computable heuristics in place of complex data structures. We recognise that human personalities fall on a wide spectrum and include adjustable parameters in our framework to reflect that variability. Our model also benefits from powerful inference mechanisms which allow us to refine it based on data.

A stochastic game-based framework for verifying trust properties expressed in temporal logic was introduced in [11]. Our framework is instead implemented using probabilistic programming where trust (and other mental states) are latent variables. Appropriateness of probabilistic programming to modelling emotions has recently been put forward by Ong et al [13]. Goodman et al [9] show how generative models of human cognition can be implemented in WebPPL, but do not consider mental states such as trust. Evans et al [4] use similar techniques to model rational agents interacting in stochastic environments, but they focus on strategic reasoning and ignore varying preferences and characteristics of agents.

3 Background

Before diving into the details of our proposed framework, we review standard notions from game theory and decision theory which the model builds upon.

3.1 Stochastic Multiplayer Games

For a set S, let $\mathcal{D}(S)$ denote the set of probability distributions on S and $\mathcal{P}(S)$ the set of subsets of S. A stochastic multiplayer game (SMG) is a tuple $(\mathrm{Ags}, \mathrm{S}, \mathrm{Act}, \mathrm{T}, \mathrm{R})$, where Ags is a set of agents, S is a finite set of states, $\mathrm{Act} = \langle \mathrm{Act}_{\mathrm{A}} \rangle_{A \in \mathrm{Ags}}$ is a finite set of global actions, $\mathrm{T} : \mathrm{S} \times \mathrm{Act} \to \mathcal{D}(\mathrm{S})$ is the transition function and $\mathrm{R} = \bigcup_{i=1}^{k} \{\mathrm{R_i}\}$ is a set of reward structures (for some $k \in \mathbb{N}$), with $\mathrm{R_i} = \{\mathrm{r_{i,A}}\}_{A \in \mathrm{Ags}}$ and $\mathrm{r_{i,A}} = (\mathrm{r}^{\mathbf{a}}_{\mathrm{i,A}}, \mathrm{r}^{\mathbf{s}}_{\mathrm{i,A}})$, $\mathrm{r}^{\mathbf{a}}_{\mathrm{i,A}} : \mathrm{S} \times \mathrm{Act} \to \mathbb{R}$ (action rewards), $\mathrm{r}^{\mathbf{s}}_{\mathrm{i,A}} : \mathrm{S} \to \mathbb{R}$ (state rewards). Each global action a is a vector of local actions of each agent; we use a_A to denote the component of a corresponding to A's local action. If an agent does not perform an action at a given state, we set $a_A = \bot$. A game $\mathcal{G}$ is *turn-based* if, for all $s \in \mathrm{S}$, there exists $A \in \mathrm{Ags}$ such that if $\mathrm{T}(s, a)(s') > 0$ for some a and s', then $a_B = \bot$ for all agents $B \neq A$. In what follows, we restrict our attention to two-player games (i.e., we assume $|\mathrm{Ags}| = 2$); this simplifies notation significantly, without compromising the generality of our approach (all the constructions extend easily to the case of more than two agents). For convenience, we introduce a function $\textsc{actions} : \mathrm{S} \to \mathcal{P}(\mathrm{Act})$, which retrieves actions available in a given state, and, for turn-based games, a function $\textsc{owns} : \mathrm{S} \to \mathrm{Ags}$, which associates a state s to an agent that takes an action in s.

Paths Let FPath denote the set of finite paths, which we take to be sequences of states interleaved with actions taken at those states, i.e., sequences of the form $s_0 a_0 s_1 \dots s_n$ such that $\mathrm{T}(s_i, a_i)(s_{i+1}) > 0$ for all $0 \leq i < n$. We assume all paths start and end in a state. For a path $\rho \in \mathrm{FPath}$, we use $\mathrm{last}(\rho)$ to refer to the last state of ρ, $\mathrm{first}(\rho)$ to refer to its first state and $\mathrm{length}(\rho)$ is its length, defined as one less than the number of states in ρ. Moreover, if $\rho = s_0 a_0 s_1 \dots a_{n-1} s_n$, then given $0 \leq i < j \leq n$, $\rho[i \dots j]$ denotes a fragment $s_i a_i \dots s_j$ of ρ. If $\delta \in \mathrm{FPath}$ is another finite path that satisfies $\delta[0 \dots n] = \rho$, we say δ *extends* ρ or that ρ is a prefix of δ. We also use ρ^s to denote the set of states in ρ and ρ^a to refer to the set of actions taken along ρ. Finally, for $a \in \rho^a$, $s(a) \in \rho^s$ is the state at which a was taken (i.e., $s(a_i) = s_i$). We allow single-state paths of length 0.

Utility It is customary to assume that preferences of agents are captured by a utility function. In particular, for $A \in \mathrm{Ags}$, cumulative utility function $\mathrm{u_A} : \mathrm{FPath} \to \mathbb{R}$ ranks possible paths according to agent A's preference, so that given ρ and ρ', $\mathrm{u_A}(\rho) > \mathrm{u_A}(\rho')$ if and only if A prefers ρ to ρ'. To reason about agent behaviour, it is helpful to assume a certain structure on the utility function; for example, taking it to be a linear combination of rewards is referred to as a *linear utility.* Then, given a set of coefficients $\{\lambda_i^A\}_{1 \leq i \leq k}$, utility gained by A in a state s is given by $\mathrm{u}^{\mathbf{s}}_{\mathrm{A}}(s) = \sum_{i=1}^{k} \lambda_i^A \mathrm{r}^{\mathbf{s}}_{\mathrm{i,A}}(s)$, while utility obtained when taking action a at s is expressed as $\mathrm{u}^{\mathbf{a}}_{\mathrm{A}}(s,a) = \sum_{i=1}^{k} \lambda_i^A \mathrm{r}^{\mathbf{a}}_{\mathrm{i,A}}(s,a)$. With that, cumulative utility gained by A along a path $\rho \in \mathrm{FPath}$ is given by $\mathrm{u_A}(\rho) = \sum_{s \in \rho^s} \mathrm{u}^{\mathbf{s}}_{\mathrm{A}}(s) + \sum_{a \in \rho^a} \mathrm{u}^{\mathbf{a}}_{\mathrm{A}}(s(a), a)$.

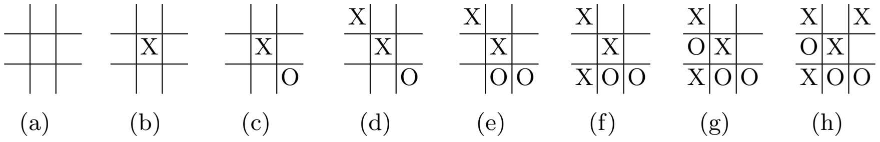
Softmax Choice A benefit of numerical utility is that it allows us to define decision making in the face of uncertainty. In particular, we assume that agents are *soft expected utility maximisers.* To clarify what that means, suppose that the system is in a state s and it's agent A's turn to take an action, selected from a set $A_0 = \textsc{actions}(s)$. For $a \in A_0$, let U_a denote expected utility of A when they take action a. Then the probability of agent A taking some action $a_0 \in A_0$ is $\frac{\exp(\alpha_A U_{a_0})}{\sum_{a \in A_0} \exp(\alpha_A U_a)}$. Therefore, agents aim to maximise their expected utility, i.e., select an action that yields such a maximum, but do that with a certain amount of noise, measured by their rationality parameter α_A. The key aspect of the above formula is the way expected utility is computed – it constitutes one of the main contributions of our work, detailed in Section 4.2.

4 Cognitive Stochastic Multiplayer Games

We begin by formally introducing our model of human decision making, which aims to capture cognitive processes that underlie agent behaviour. The framework is grounded on standard assumptions from game theory, such as (soft) expected utility maximisation and discounting, but enriches them with a complex mechanism of reasoning. We postulate that agents, in order to select the best course of action, quantify the likelihood of future paths (of limited length), which involves nested reasoning and maintaining belief about their opponent(s). To ensure efficient execution of our model without compromising its integrity, we define rich data structures and mechanisms that approximate cognitive processes in humans. The model is predominantly expert-driven, as some of its components require insights into human psychology, but supports powerful inference mechanisms that allow its parameters to be learned from data.

Example. *We illustrate our construction by showing how it can be applied to analyse plays of tic-tac-toe. It is a two-player pen and paper game played on a 3x3 grid in which players take turns to insert an 'X' (the cross player) or an 'O' (the nought player) into an empty square. A goal of each player is to fill a full row, column or a diagonal with their mark ('X' or 'O'). Figure 1 shows a possible play of tic-tac-toe; note that the cross player always starts.*

There exist many strategies for either player which guarantee at least a draw, and they are not hard to follow. Hence, a game of tic-tac-toe between two players following optimal strategies will always end in a draw. In reality, however, this is not always the case. Children, especially the younger ones, often find tic-tac-toe challenging, as their cognitive limitations prevent them from playing perfectly. In a matchup between a parent and their kid, the adult should be a clear favourite. However, as some readers may know from experience, the outcome of such a game is far from decided and in fact the child wins more often than not. In what follows, we use our framework to explain this apparent paradox.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 1: An example game of tic-tac-toe, developing left to right

4.1 Model Syntax

A cognitive stochastic multiplayer game (CSMG) is a tuple $(\mathrm{Ags}, \mathrm{S}, \mathrm{Act}, \mathrm{T}, \mathrm{R}, \mathrm{P}, \{\vec{\lambda}^A\}_{A \in \mathrm{Ags}}, \{\mathrm{be}^0_{\mathrm{A}}\}_{A \in \mathrm{Ags}}, \{\theta_A\}_{A \in \mathrm{Ags}}, \{\mathrm{est}^0_{\mathrm{A}}\}_{A \in \mathrm{Ags}})$, where Ags, S, Act and T, R are standard components, introduced for SMG in Section 3.

Example. *Tic-tac-toe between a parent and their child is modelled by setting* $\mathrm{Ags} = \{\mathit{kid}, \mathit{parent}\}$ *(abbreviated to k and p in subscripts); set of actions* $\mathrm{Act}_{\mathrm{A}} = \{(x, y) \mid 0 \leq x, y \leq 2\}$ *of each player* $(A \in \mathrm{Ags})$ *consists of nine elements, each* *identifying a single position in the grid (e.g., Figure 1f adds an 'X' at position* $\langle 2, 0\rangle$*); each state records the history of execution (e.g., state from Figure 1d is* $[(1,1),(2,2),(0,0)]$*); the transition function encodes the informal description of the game (we assume kid goes first); there is one physical reward structure* R_1 *that assigns no action rewards, but allocates non-zero rewards at states in which the game has ended:* $+1$ *for the winner and* -1 *for the loser. Note that tic-tac-toe is a turn-based game.*

Below, we specify in detail the novel elements of the above definition and motivate them by referring to the running example.

Example. *A standard approach to modelling players' motivations in a game like tic-tac-toe is to assign a positive reward for winning the game and a negative reward for losing. However, it is clear that when parent plays against their child, something else is at stake. One may be tempted to modify the reward structure by changing parent's rewards to equal child's rewards, so that the parent prefers losing to winning. However, this would be too simplistic and lead to unrealistic behaviour, whereby parent never defends against obvious threats and plays unnatural moves. Instead, we postulate that, rather than trying to increase the probability of child winning, the parent wants to increase child's* satisfaction *from playing the game. Unlike the outcome of the game, satisfaction constitutes private information of each player. Therefore, it cannot be captured by the standard reward structure* R *which deterministically assigns rewards at each state. Instead, we treat satisfaction, and other* mental states*, differently.*

Mental Rewards In general, our observation is that humans are motivated not only by physical, easily quantifiable quantities, such as money or time, but they are driven also by *mental goals*. Examples include high level desires such as developing relationships with people, making others happy or causing an enemy to suffer, as well as emotional considerations: avoiding guilt or shame, maximising trust and pride. Distinctive features of these *mental states* are that they are difficult to measure and quantify and typically not observable to others – which complicates things further, given that an agent might be motivated by *someone else's* mental state (e.g., I may care how much you trust me). To represent mental goals, we equip each agent with a set of *mental variables* that are latent in a sense that other agents do not know their values. Each mental variable represents a mental state or an emotion, such as anger, joy, regret or pride. An agent knows how they feel, and so they can evaluate their mental state at any point – such evaluation might take agent's belief and personality into account. On the other hand, mental states of others are unknown to an agent, which necessitates *estimation* of their values. The intuition behind mental rewards is to assign a numeric value to those feelings experienced by the agents so that they can be included in agent's utility function.

Before diving into detail, a note about nomenclature: *mental state* is the emotion we are considering (trust, guilt, shame etc.); *mental goal* is closely related, but captures what an agent is trying to achieve with respect to some mental state (e.g., minimize guilt or increase trust); *mental variable* is purely syntactic – it identifies a mental state of an agent; *mental reward* is numerical, as explained above.

Formally, along with a set of reward structures R which model physical rewards, we introduce a set of mental reward structures $\mathrm{P} = \bigcup_{i=1}^{l}\{\mathrm{P_i}\}$, one for each *mental state* we wish to model (we assume there are l such). Each mental reward structure $\mathrm{P_i}$ is a tuple $(\{\eta_i^A\}_{A\in\mathrm{Ags}}, \sigma_i, \omega_i)$, where η_i^A is the i^{th} mental variable of an agent $A \in \mathrm{Ags}$, while σ_i and ω_i are used to compute mental rewards.

In particular, $\sigma_i : \mathrm{FPath} \times \mathrm{Act} \times \mathrm{Ags} \times \mathrm{Ags} \rightarrow ([-1,1] \rightarrow [-1,1])$ is a mental state dynamics model, which encodes agents' appraisal of mental states of others. Given a path $\rho \in \mathrm{FPath}$, $a \in \mathrm{Act}$ and $A, B \in \mathrm{Ags}$, $\sigma_i(\rho, a, A, B)$ is a dynamics function that encodes a heuristic according to which A estimates how i^{th} mental state of B changes when action a is taken in last state of ρ. For instance, Bob may hypothesize that being late for his weekly meeting will deteriorate his relationship with his supervisor, unless that relationship is already at a low point, in which case it may get better as he at least showed up. In some cases, the dynamics model is constructed using common sense, while at other times research findings from fields of psychology and cognitive sciences are utilised. Dynamics functions are typically continuous (but we do not enforce it) and may be partial, as some mental states only take nonnegative values (trust, for example).

On the other hand, mental state evaluation function, $\omega_i : \mathrm{FPath} \times \mathrm{Ags} \rightarrow [-1,1]$, captures the mechanism through which an agent experiences their own mental state. For example, it may be that the supervisor has already given up on Bob due to his past behaviour and the relationship cannot be recovered in her eyes. This may be because the supervisor is unforgiving, which Bob might not be aware of. In general, evaluation function ω_i may take into account personality and beliefs of an agent and its definition will often be informed by insights from relevant disciplines.

Example. *Coming back to our tic-tac-toe example, satisfaction is the only mental state we model, represented by mental variables η_1^k (child's satisfaction) and η_1^p (satisfaction of the parent). The mechanism through which the parent makes inferences about the value of η_1^k is the satisfaction dynamics function σ_1. It encodes a heuristic that dictates how satisfaction changes as the game develops and it is specified in terms of* forks *and* blunders*, which are certain types of moves in tic-tac-toe, easily explained with an example. For a fork, see Figure 1f – the cross player now has two winning lines (left column and a diagonal), which guarantees victory in the next move (assuming optimal play). When it comes to blunders, there are two types: (i) not defending against an obvious threat (any move other than $\langle 3, 1\rangle$ in the state from Figure 1e is a blunder of that type), (ii) not taking an obvious winning opportunity (e.g., any move other than $\langle 1, 3\rangle$ in the state from Figure 1g). With that, the heuristics captured by satisfaction dynamics postulates that winning the game increases satisfaction, especially if the game was won as a consequence of creating a fork; however, satisfaction is* *lower if opponent blundered. On the other hand, losing the game brings about a decline in satisfaction and the reduction is much steeper if losing is a result of own blunder. For a precise specification, see Section A.1 in the Appendix. We omit specification of satisfaction evaluation function, as it does not feature in agents' decision making (which is explained when utility functions of agents are introduced). However, Section 5.1 introduces an evaluation function for trust.*

Agent Characteristics Each agent in our framework is equipped with a vector of *goal coefficients*, which provides subjective weights of various reward structures. This vector, referred to as *agent characteristics* and denoted $\vec{\lambda}^A$ (for agent A), gives rise to the $\{\vec{\lambda}^A\}_{A\in\mathrm{Ags}}$ component in our definition. Goal coefficients encode players' preference over various rewards. Without loss of generality, we assume that entries of $\vec{\lambda}^A$ sum to 1.

Belief A vector of goal coefficients, which reflects personality of a player, constitutes private information of an agent. A discrepancy between actual characteristics of an agent and opponents' perception of these characteristics is captured by the notion of *belief.* To express it, we introduce a probability space on the set of possible goal coefficient vectors, $\Lambda = \{\langle\lambda_i\rangle | \sum_i \lambda_i = 1\}$, of agents and, for each $A \in \mathrm{Ags}$ and a path $\rho \in \mathrm{FPath}$, a belief function $\mathrm{be}^{\rho}_{\mathrm{A}} : \Lambda \to \mathbb{R}$ which acts as a density function on Λ. Then, a probability space of agent A after executing ρ is a triple $(\Lambda, \mathcal{F}_A, \mathrm{Pr}^{\rho}_{\mathrm{A}})$, where the set of events $\mathcal{F}_A$ is a standard Borel σ-algebra generated by open balls in Λ and the probability measure $\mathrm{Pr}^{\rho}_{\mathrm{A}} : \mathcal{F}_A \to [0,1]$ is given in terms of belief by $\mathrm{Pr}^{\rho}_{\mathrm{A}}(X) = \int_X \mathrm{be}^{\rho}_{\mathrm{A}}(\vec{\lambda})d\vec{\lambda}$. We assume agents update their belief in a Bayesian way every time their opponent takes an action. That requires initial, i.e., *prior*, belief of every agent to be specified, represented by the $\{\mathrm{be}^0_{\mathrm{A}}\}_{A\in\mathrm{Ags}}$ component of a tuple defining CSMG. A formal specification of belief update is available on request, but the intuition is that observing an action taken by an opponent provides an agent with an insight on what values of opponent's goal coefficients are more likely than others. This insight is incorporated into agent's belief to produce an updated belief. Finally, note that to enable efficient learning of beliefs (see Section 4.3) we support a representation of belief as a Dirichlet distribution, with custom-defined belief update.

Example. *In tic-tac-toe, the parent has two sources of motivation (each represented by a reward structure): winning the game and maximising their child's satisfaction. Therefore, parent's characteristics is specified by a two element vector $\langle\lambda^p_1, \lambda^p_2\rangle$. We assume that they assign significantly more importance to the latter incentive, i.e., $\lambda^p_1 < \lambda^p_2$.*

However, the child is not aware of what their parent is up to – if they were, the game would not be entertaining for them. Hence, the child will think that $\lambda^p_1 = 1$, which can be formally captured using Dirac distribution.

Meta-parameters Apart from a vector of goal coefficients, each agent is characterised also by a set of meta-parameters, so called because they control agent's decision-making process on a higher level. One such is lookahead $\beta \in \mathbb{N}$, which determines how far into the future a player looks when computing their action. Other meta-parameters include *rationality* $\alpha \in [0, \infty]$, which comes into play as a parameter of the softmax choice formula introduced in Section 3, and *discount factor* $\gamma \in (0, 1]$, which defines how much an agent discounts future rewards. The set of all meta-parameters is denoted Θ.

Example. *Lookahead is especially important in tic-tac-toe and popular board games such as chess or go. One of the distinctive characteristics of advanced players is their ability to simulate the execution of the game multiple moves into the future, allowing them to evaluate a given choice better. Computers do it even better and can beat humans in most games. In tic-tac-toe, we would expect the child to have a lookahead no bigger than three, while for a parent a value of six or seven would not be uncommon. Rationality would also differ significantly; we expect the kid to err in their computations more often than their parent, which we could model by setting* $\alpha_k = 5$ *and* $\alpha_p = 20$*. Finally, discounting is useful to reflect the fact that players typically prefer to win the game sooner rather than later – any value of* γ *smaller than one does the trick.*

In our framework, nested reasoning of agents necessitates that they quantify the likelihood of different configurations of meta-parameters of their opponent, captured by the $\{\theta_A\}_{A \in \mathrm{Ags}}$ component in the definition. In particular, for each agent $A \in \mathrm{Ags}$ and their opponent $B \in \mathrm{Ags}$, we take $\Theta_B = [0, \infty] \times \mathbb{N} \times (0, 1]$ to be the set of possible meta-parameter tuples of B; then $\theta_A : \Theta_B \to \mathbb{R}$ describes A's estimation of B's meta-parameters.

Mental State Estimation Recall that a mental variable of an agent, say η_1^k, identifies that agent's mental state (in this case, satisfaction of the child). At any point of execution, the value of agent A's ith mental state will be denoted as $[\![\eta_i^A]\!]^\rho$ (for current path ρ) and given by $\omega_i(\rho, A)$.

However, an agent cannot directly compute the value of their opponent's mental state. Instead, they start with an estimation (which takes a form of a probability distribution) and they update it according to the dynamics model as the execution progresses. To formalise this, for a set S, we let $\mathcal{D}^f(S)$ be the set of probability distributions on S such that elements that are assigned a positive probability form a countable subset of S. Also, for a function $f : X \to Y$ and an element $y \in Y$, $f^{-1}(y)$ denotes the preimage of y, i.e., the set $\{x \in X \mid f(x) = y\}$. Also, for $B \in \mathrm{Ags}$, $\mathrm{H_B} = \bigcup_{1 \le i \le l} \{\eta_i^B\}$ is a set of mental variables of B. Then, for $A, B \in \mathrm{Ags}$ and $\rho \in \mathrm{FPath}$, the *mental state estimation function* $\mathrm{est}_\mathrm{A}^\rho : \mathrm{H_B} \to \mathcal{D}^f([-1, 1])$ gives A's estimation, expressed as a probability distribution, of each mental state η_i^B of B, upon executing path ρ. Letting $\rho = s_0 a_0 \ldots s_n a_n s_{n+1}$ and fixing $\eta_i^B \in \mathrm{H_B}$, $\mathrm{est}_\mathrm{A}^\rho(\eta_i^B)$ is computed recursively by A, starting from their initial estimations $\mathrm{est}_\mathrm{A}^0 : \mathrm{H_B} \to \mathcal{D}^f([-1, 1])$ and using the dynamics function σ_i, as follows:

$$\mathrm{est}_\mathrm{A}^\rho(\eta_i^B)(x) = \begin{cases} \mathrm{est}_\mathrm{A}^0(\eta_i^B)(x) & \text{if } n+1 = 0, \\ \sum_{y \in \sigma_i(\rho[0\ldots n], a_n, A, B)^{-1}(x)} \mathrm{est}_\mathrm{A}^{\mathrm{s_0 \ldots s_n}}(\eta_i^B)(y) & \text{otherwise.} \end{cases}$$

With that, agent A's expectation of the value of ith mental state of B is computed with respect to A's estimation function as $\mathbb{E}^{\rho}_{A}[\eta^{B}_{i}] = \sum_{x \in [-1,1]} x \cdot \mathrm{est}^{\rho}_{\mathrm{A}}(\eta^{B}_{i})(x)$. A continuous case of the above construction, which requires placing restrictions on the dynamics functions σ_i and involves some notational inconveniences, is omitted for space reasons.

4.2 Model Semantics

Having introduced the components of our model, we now describe how they combine to give its semantics, which boils down to a formal specification of agents' decision-making process. The first step in that direction is the definition of a *cognitive utility function*. Then, a method of computing own and opponent's expected utility is given. However, due to multiple sources of partial observability which differ between agents, special care has to be taken to truthfully capture their cognitive processes.

Utility Function Cognitive utility function extends the standard one, defined in Section 3, with mental rewards. Note that, due to the nature of mental rewards, we record their values in states, but do not associate any mental rewards with actions. Therefore, action utility remains the same in the cognitive model. However, for an agent $A \in \mathrm{Ags}$, their opponent $B \in \mathrm{Ags}$ and a path $\rho \in \mathrm{FPath}$ with $\mathrm{last}(\rho) = s$, *cognitive state utility function* $\mathtt{cu}^{\mathbf{s}}_{\mathrm{A}} : \mathrm{FPath} \to \mathbb{R}$ describes utility gained by A in s as follows:

$$\mathtt{cu}^{\mathbf{s}}_{\mathrm{A}}(\rho) = \sum_{i=1}^{k} \lambda^{A}_{i} \mathrm{r}^{\mathbf{s}}_{\mathrm{i,A}}(s) + \sum_{i=1}^{l} \lambda^{A,A}_{i} [\![\eta^{A}_{i}]\!]^{\rho} + \sum_{i=1}^{l} \lambda^{A,B}_{i} \mathbb{E}^{\rho}_{A}[\eta^{B}_{i}].$$

Hence, utility obtained by an agent at a state is a weighted sum of their physical rewards and mental rewards, where the latter are defined in terms of values of agent's own mental states and expected values of their opponent's mental states. Typically, some, if not most, of the mental goal coefficients $\lambda^{A,A}_{i}$ and $\lambda^{A,B}_{i}$ will be equal to 0 and the corresponding component of the utility function may be omitted, as the example below illustrates.

Example. *In the tic-tac-toe example, parent's cognitive state utility at a state s where $s = \mathrm{last}(\rho)$ is*

$$\mathtt{cu}^{\mathbf{s}}_{\mathrm{p}}(\rho) = \lambda^{p}_{1} \mathrm{r}^{\mathbf{s}}_{\mathrm{p}}(s) + \lambda^{p,k}_{1} \mathbb{E}^{\rho}_{p}[\eta^{k}_{1}],$$

where p is a shorthand for parent, *η^{k}_{1} is kid's satisfaction and state reward $\mathrm{r}^{\mathbf{s}}_{\mathrm{p}}(s)$ equals 1 if parent wins the game and -1 if they lose. On the other hand, child's utility lacks the cognitive component as they only care about winning the game:*

$$\mathtt{cu}^{\mathbf{s}}_{\mathrm{k}}(\rho) = \lambda^{k}_{1} \mathrm{r}^{\mathbf{s}}_{\mathrm{k}}(s).$$

It may seem counter-intuitive that satisfaction does not feature in child's utility function, but our reasoning is that, during the game, the kid only thinks about *winning, while satisfaction comes later, without the child fully realising the origin of that sensation (due to cognitive limitations). It illustrates the difference between parent's long-term vs child's short-term thinking.*

Things get a little more complicated when agent B attempts to compute their opponent's utility, i.e., the value of $\mathtt{cu}^{\mathbf{s}}_{\mathrm{A}}(\rho)$. First of all, B does not know the value of A's goal coefficients, maintaining a belief (see Section 4.1) over their values instead. Second, B does not know the values of A's mental states and A's estimations of B's mental states. Using the expectation operator liberally, we may express the expected value of A's utility in the last state of some path $\delta \in \mathrm{FPath}$, computed by B after executing path ρ as

$$\mathbb{E}^{\rho}_{B}[\mathtt{cu}^{\mathbf{s}}_{\mathrm{A}}(\delta)] = \sum_{i=1}^{k} \mathbb{E}^{\rho}_{B}[\lambda^{A}_{i}]\mathrm{r}^{\mathbf{s}}_{\mathrm{i,A}}(s) + \sum_{i=1}^{l} \mathbb{E}^{\rho}_{B}[\lambda^{A,A}_{i}]\mathbb{E}^{\delta}_{B}[\eta^{A}_{i}] + \sum_{i=1}^{l} \mathbb{E}^{\rho}_{B}[\lambda^{A,B}_{i}]\mathbb{E}^{\rho}_{B}[\mathbb{E}^{\delta}_{A}[\eta^{B}_{i}]].$$

Note two different paths present in the above expression – here we require that δ extends ρ (i.e., ρ is a prefix of δ), reflecting the fact that B computes *future* utility of A as part of their decision-making process (see below).

While B's expectations of A's goal coefficients are easily computed with respect to B's belief and $\mathbb{E}^{\delta}_{B}[\eta^{A}_{i}]$ is computed with respect to B's mental state estimation function, nested expectation is more difficult to resolve in principle. In fact, rather than introducing a complex structure such as nested estimation, we assume agents approximate nested expectation, such as $\mathbb{E}^{\rho}_{B}[\mathbb{E}^{\delta}_{A}[\eta^{B}_{i}]]$, by taking the true value of their mental state at ρ, $[\![\eta^{B}_{i}]\!]^{\rho}$ and using mental state dynamics from there. Formally, letting $\delta = s_0 a_0 \dots s_n a_n s_{n+1}$, nested expectation is resolved as follows:

$$\mathbb{E}^{\rho}_{B}[\mathbb{E}^{\delta}_{A}[\eta^{B}_{i}]] = \begin{cases} [\![\eta^{B}_{i}]\!]^{\rho} & \text{if } \rho = \delta, \\ \sigma_i(\delta[0\dots n], a_n, A, B)(\mathbb{E}^{\rho}_{B}[\mathbb{E}^{\delta[0\dots n]}_{A}[\eta^{B}_{i}]]) & \text{otherwise.} \end{cases}$$

Decision-Making Process Recall that, in order to select an action according to the softmax choice rule, an agent computes expected future utility corresponding to each available action. This computation, in general, is recursive in nature, where the depth of the recursion is given by agent's lookahead parameter β. The recursion explores the game tree, computing expected utilities at each visited state, including states where agent's opponent takes actions. To predict actions of other player, an agent computes their expectation of their opponent's future utility, which, combined with softmax choice rule, allows an agent to quantify the likelihood of different future paths.

To formally express this expected future utility for agent A, we define a family of mutually recursive random variables $U^{\rho,\delta,h}_{A,A}$, $U^{\rho,\delta,h}_{A,B}$, $U^{\rho,\delta,h,a}_{A,A}$, $U^{\rho,\delta,h,a}_{A,B}$ and $P^{\rho,\delta,h,a}_{A,B}$ on the space Θ_B of B's meta-parameters. The first four of those random variables express different variants of expected (with respect to B's mental characteristics) utility of A or B (depending on the second subscript), accumulated over the next h steps starting in the last state of δ, computed

MK: not sure what 'typically' means - I thought you need to make sure that δ follows ρ, which means that they have the last and first state in common?

MO: by 'typically', i meant that this is the relationship that holds when this expression is used as part of the decision making process. in principle, any other path δ could be substituted in this expression and it shouldnt be nonsensical. but maybe that's just confusing - please see my edit (in fact, I also updated the next paragraph too as I realised it was out of date!). also, the relationship between δ and ρ is not that δ follows ρ but rather it extends it, which means that ρ is a prefix of δ (normally, both start at the initial say) - maybe that's nonstandard and should be stressed?

MK: I think this is fine, and implies other assumptions on δ, but you may want to say 'here we require' and add a clarification in brackets after 'extends', (i.e., ρ is a prefix of δ), which clarifies things. I expect there may be things that I missed, so please by A (as denoted by the first subscript) at a point of execution where current path is ρ. If an action a is specified in the superscript, the random variable denotes expected utility assuming a is taken after executing δ; otherwise it is the expected utility in a state reached after executing path δ. The last random variable, $P^{\rho,\delta,h,a}_{A,B}$, expresses the probability that agent B takes an action a in the last state of δ, computed by A after executing ρ.

We present the definitions of those random variables below. We call this set of equations *decision making equations*. To improve readability, we fix an element $\langle \alpha_B, \beta_B, \gamma_B \rangle \in \Theta_B$ and implicitly assume that all occurrences of random variable U are passed this element as argument. We also assume that the current path is ρ and δ is a future extension of ρ, with $\mathrm{last}(\delta) = s$. The definitions are as follows:

$$U^{\rho,\delta,h}_{A,A} = \begin{cases} \mathtt{cu}^{\mathtt{s}}_{\mathrm{A}}(\delta) & \text{if } h = 0, \\ \mathtt{cu}^{\mathtt{s}}_{\mathrm{A}}(\delta) + \sum_a P^{\rho,\delta,h,a}_{A,\textsc{owns}(s)} U^{\rho,\delta,h,a}_{A,A} & \text{if } h > 0. \end{cases} \tag{1}$$

$$U^{\rho,\delta,h}_{A,B} = \begin{cases} \mathbb{E}^{\rho}_{A}[\mathtt{cu}^{\mathtt{s}}_{\mathrm{B}}(\delta)] & \text{if } h = 0, \\ \mathbb{E}^{\rho}_{A}[\mathtt{cu}^{\mathtt{s}}_{\mathrm{B}}(\delta)] + \sum_a P^{\rho,\delta,h,a}_{A,\textsc{owns}(s)} U^{\rho,\delta,h,a}_{A,B} & \text{if } h > 0. \end{cases} \tag{2}$$

$$U^{\rho,\delta,h,a}_{A,A} = \mathtt{cu}^{\mathtt{a}}_{\mathrm{A}}(\delta, a) + \gamma_A \sum_{s'} \mathrm{T}(s,a)(s') U^{\rho,\delta a s',h-1}_{A,A} \tag{3}$$

$$U^{\rho,\delta,h,a}_{A,B} = \mathbb{E}^{\rho}_{A}[\mathtt{cu}^{\mathtt{a}}_{\mathrm{B}}(\delta, a)] + \gamma_B \sum_{s'} \mathrm{T}(s,a)(s') U^{\rho,\delta a s',h-1}_{A,B} \tag{4}$$

$$P^{\rho,\delta,h,a}_{A,B} = \frac{\exp(\alpha_B U^{\rho,\delta,h,a}_{A,B})}{\sum_{a' \in \textsc{actions}(s)} \exp(\alpha_B U^{\rho,\delta,h,a'}_{A,B})} \tag{5}$$

Hence, looking first at Equations 1 and 2, expected future utility computed by an agent depends on the time horizon. When it is equal to 0, utility may be computed directly; in Equation 1 it is simply agent's own cognitive state utility, while in Equation 2 it is an expectation, computed with respect to one's own belief, of opponent's cognitive state utility. Otherwise, expected utility accumulated within next h steps is equal to cognitive state utility (or, again, an expectation of opponent's utility) gained at the current state plus a sum of expected utilities corresponding to different actions available to an agent at the current state (whichever agent "owns" that state), weighted by probabilities of taking those actions. That probability, in turn, is computed according to Equation 5, where the expected utilities are expressed by random variables U.

Next, looking at Equations 3 and 4, expected utility corresponding to an action a taken in state s (where $s = \mathrm{last}(\delta)$) is simply action utility (in Equation 3, or an expectation of opponent's action utility in Equation 4), plus a discounted sum of expected utilities in states reachable from s, weighted by transition probabilities.

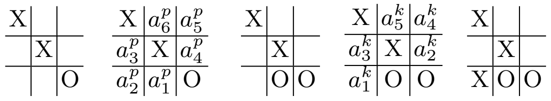
Finally, the probability of B taking action a (Equation 5) is based on the softmax choice formula, adapted to the current notation. Note that we allow A and B to refer to the same agent.

With that, we can express the probability of agent A taking action a when current path is ρ, computed by the system's modeller, as $Prob^{\rho}(a) = \mathbb{E}^{\rho}[P_{A,A}^{\rho,\rho,\beta_A,a}]$, where $A = \text{OWNS}(\text{last}(\rho))$. Similarly, probability of agent B taking action a at the last state of δ, computed by agent A after executing path ρ, is simply an expectation (computed with respect to A's meta-parameters estimations of B) of the value of random variable $P_{B,A}^{\rho,\delta,\beta_B,a}$, formally expressed as $Prob_A^{\rho}(a) = \mathbb{E}_A^{\rho}[P_{B,A}^{\rho,\delta,\beta_B,a}]$.

Example. *We briefly overview how the parent computes their action in state* $s_0 = [(1,1)]$ *from Figure 2a. A full explanation is available in the Appendix. For simplicity, we assume* $\beta_p = 3$ *(short lookahead),* $\alpha_p \to \infty$ *(perfect rationality) and* $\gamma_p = 1$ *(no discounting); moreover, we assume the parent knows child's meta-parameters:* $\beta_k = 2$, $\alpha_k = 5$ *and* $\gamma_k = 0.7$. *We set parent's goal coefficients to* $\langle 0.1, 0.9\rangle$ *– child's satisfaction is parent's priority.*

To decide on an action, an agent generally computes expected utilities corresponding to each available action (Equation 5) and chooses probabilistically - the higher the expected utility, the higher the chances of an action being picked. In the special case of perfectly rational, only actions that yield maximal expected utility are considered (usually there is only one such). The six actions available to the parent in s_0 *are depicted in Figure 2b; Table 1b shows expected utilities corresponding to each action computed according to decision making equations. We focus on* a_1^p*; in particular, letting* ρ *be the path taken to reach* s_0*, we analyse how* $U_{p,p}^{\rho,\rho,3,a_1^p}$ *is computed.*

In absence of action rewards, expected utility for a_1^p *reduces to the expected utility at a state reached after taking* a_1^p *(by Equation 3, given lack of discounting) – call it* s_1 *and the path taken to it* δ_1*. Now, at* s_1 *it is the child who takes an action, so computing* $U_{p,p}^{\rho,\delta_1,2}$ *requires the parent to quantify the likelihood of the child taking different actions (by Equation 1). Possible actions for the kid at* s_1 *are depicted in Figure 2d; to compute probabilities of the child taking each of them (*$P_{p,k}^{\rho,\delta_1,2,\cdot}$*), parent computes expected utilities of the kid corresponding to each action according to Equation 4 (it involves using belief and meta-parameter estimations). It turns out that* a_1^k *is deemed most likely by the parent (*$P_{p,k}^{\rho,\delta_1,2,a_1^k} \approx 0.74$ *(recall that kid is not perfectly rational, so they choose their action noisily), which is reassuring, since any other move loses the game for the kid (assuming optimal play on parent's side). All that remains for the parent is computing their expected utility corresponding to each action of the child – the values are summarised in Table 1a. They are not hard to compute using Equation 3 – in absence of action rewards, each* $U_{p,p}^{\rho,\delta,2,a_i^k}$ *is simply a sum of rewards in the next two states, first of which is given by the transition function, while the second is easily determined by the parent. With that, the value of* $U_{p,p}^{\rho,\rho,3,a_1^p} = U_{p,p}^{\rho,\delta_1,2}$ *(1.05) is given (according to Equation 2) by the sum of expected utilities weighted by probabilities from Table 1a.*



(a) s_0 (b) (c) s_1 (d) (e) s_2

Fig. 2: Reference for illustration of decision-making process

Table 1: Expected utilities and action likelihoods computed by the parent

(a)

	a^k_1	a^k_2	a^k_3	a^k_4	a^k_5
$U^{\rho,\delta_1,2,\cdot}_{p,p}$	1.26	-0.17	-0.17	1.09	1.09
$P^{\rho,\delta_1,2,\cdot}_{p,k}$	0.74	0.06	0.06	0.06	0.06

(b)

	a^p_1	a^p_2	a^p_3	a^p_4	a^p_5	a^p_6
$U^{\rho,\rho,3,\cdot}_{p,p}$	1.05	0.04	1.05	0.5	0.04	0.5

4.3 Implementation

Along with the theoretical framework, we provide a computational tool that automates the reasoning presented above. As we mentioned above, probabilistic programming has recently been recognised as highly appropriate for expressing cognitive reasoning of humans. Powerful inference mechanisms allowed us to implement agent decision-making succinctly and facilitated inference from data. WebPPL has been chosen as implementation language due to its high accessibility and existence of related code base [4,9].

We now give an overview of the tool[1], which serves also as a summary of the operation of our framework. Recall that our main contribution is a novel formalisation of agent decision making. Its most basic application is an ability to *simulate* the execution of a turn-based, stochastic game, which constitutes a primary feature of our tool. Formally speaking, the tool computes posterior predictive distribution over future actions of agents. It requires several inputs: mechanics of the game (states, actions, transition function etc), specification of the mental component (mental states and their dynamics) and each agent's utility, parameters and initial state. The simulation may then be run for any number of iterations of the game and it generates a sequence of actions taken at each step. Note that this is a probabilistic process and the computed trace might differ between executions; what remains the same, however, are the probability distributions according to which agents choose their actions. As the execution progresses, players update their beliefs in a Bayesian fashion, assimilating the information gained by observing their opponents' actions.

The main obstacle encountered when simulating execution of a game is the need to provide parameters and initial states of agents. These are often not known; even if some information about the agents is at hand, it may not be easy to translate it to the required format. However, behavioural data, i.e., a record

[1] Available at https://github.com/maciekolejnik/cognitive-modelling

of actions taken by agents, is likely to be much more readily available. Hence, the second functionality of our tool involves *learning* preferences, beliefs and estimations of agents from data. The inference is again Bayesian in nature and requires a prior, which may encode population-wide statistics, our belief about characteristics of a particular agent or be set to uniform if no information is available. The process is graphically summarised in Figure 10 available in the Appendix.

5 Experiments

To validate the proposed formalism and showcase the functionality of our probabilistic programming tool, we now use it to analyse three scenarios of human interactions where mental motivations play a role. We perform two types of experiments, reflecting capabilities of our software: (i) simulations, where we initialise agents in various ways and use the tool to predict their behaviour in the game, and (ii) learning and prediction, where we use the tool to infer agents' parameters from data and make predictions about their future behaviour based on what we've learned. Note that the predictions generated by the tool always take form of posterior predictive distributions over available actions; however, if agents are assumed to be (almost) perfectly rational ($\alpha \to \infty$), such distribution will effectively be a Dirac distribution. In all the games below, we assume states encode execution histories, which allows us to use states where paths are required, thereby simplifying notation.

5.1 Trust Game

The first case study we consider is a well-known Trust Game [2] (aka Investment Game). It involves two agents, Alice and Bob, who are given an opportunity to earn some money. In particular, Alice is endowed with \$10 which she can keep to herself (*withhold*) or share an integral fraction of it with Bob. In the latter case, a skilled investor which he is, Bob doubles the amount received and can share any (integral) part of it with Alice. Figure 3 shows a graphical representation of the game where initial endowment of Alice is \$4 (rather than \$10) to help readability. Note that states are subscripted with the sequence of actions taken so far in the game.

Standard game-theoretic analysis based on equilibria predicts that no cooperation will arise in this scenario. A crucial assumption needed to arrive at that conclusion is that players only care about money; then Bob has no incentive to share any of his profits and Alice, anticipating Bob's behaviour, has no reason to invest with him. This reasoning may be applied iteratively, which means that no cooperation is anticipated even in a repeated version of the Trust Game. However, human experiments show that non-zero transfers occur in majority of interactions [12], contradicting the theoretical predictions. Evidently, human decision-making is not as simple as maximising monetary payoffs.

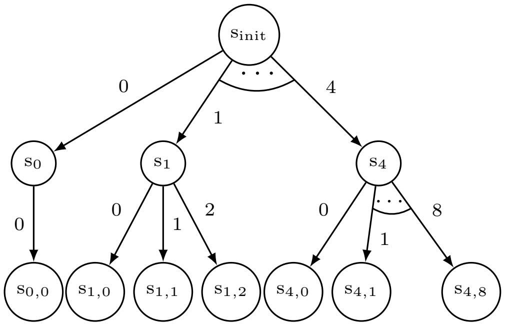


Fig. 3: One iteration of Trust Game

As hinted in the name of this scenario, our hypothesis is that trust between agents should be included in the analysis. Humans are social animals (an observation that goes back to Aristotle) guided by social norms, such as reciprocity, and driven by a desire to develop new, and preserve existing, interpersonal relationships. For the purposes of our example, inspired by social and cognitive science research [6,5,10], we will assume that those motivations are captured by trust, roughly defined as a *subjective belief of a trustor over trustee's willingness to go out of one's way to help trustor achieve their goal.* Trust will be modelled as a mental state using our formalism, allowing us to capture how trust considerations shape decision making of agents.

Game Setup The standard components of the model are fairly clear given the informal description; a single physical reward structure R reflects monetary incomes and outcomes occurring in the game, with state rewards that model Alice's endowments and Bob's investment, and action rewards that represent transfers.

Mental Rewards The most important novel component of our cognitive model is a mental reward structure that captures how agents perceive trust. It consists of mental variables η_1^{Alice} and η_1^{Bob}, representing Alice's trust towards Bob and Bob's trust towards Alice, respectively. For the sake of readability, we will write those mental variables as $\texttt{trust}_{\text{Alice,Bob}}$ and $\texttt{trust}_{\text{Bob,Alice}}$, respectively, which also underlines that trust is a binary relation where order matters.

Before describing the mental reward structure in detail, we give the utility function for each agent. The idea is that, besides money, an agent cares about the trust that *their opponent* places in them.

$$\begin{aligned}\texttt{cu}^{\mathbf{s}}_{\text{Alice}}(s) &= \lambda_m^{\text{Alice}} \mathrm{r}^{\mathbf{s}}_{\text{Alice}}(s) + \lambda_\tau^{\text{Alice}} \mathbb{E}^s_{\text{Alice}}[\texttt{trust}_{\text{Bob,Alice}}],\\ \texttt{cu}^{\mathbf{s}}_{\text{Bob}}(s) &= \lambda_m^{\text{Bob}} \mathrm{r}^{\mathbf{s}}_{\text{Bob}}(s) + \lambda_\tau^{\text{Bob}} \mathbb{E}^s_{\text{Bob}}[\texttt{trust}_{\text{Alice,Bob}}].\end{aligned}$$

Recall that states encode execution histories, which explains why s is passed as argument to cognitive utility functions. We now describe how agents compute their expectations and also give a formal definition of trust.

Trust Dynamics The heuristic agents use to estimate how much trust their opponent places in them is expressed by the trust dynamics function $\sigma_1 : \mathrm{FPath} \rightarrow \mathrm{Act} \rightarrow \mathrm{Ags} \rightarrow \mathrm{Ags} \rightarrow ([0,1] \rightarrow [0,1])$ that describes how one agent's estimation of their opponent's trust changes after some action is taken in some state. A full specification of σ is given in the Appendix , but a crucial insight is that trust increases faster (as a response to trustworthy actions by an opponent) when its value is low and decreases faster when its value is high. The formulation of σ_1 is fairly complex and we invite readers to consult the codebase for full detail, but a crucial insight is that trust increases faster (as a response to trustworthy actions by an opponent) when its value is low and decreases faster when its value is high. This dynamics is captured by exponential and logarithm functions, displayed in Figure 5. Assuming current trust is t, it will increase to $\exp(t-1)$ if opponent chooses a trustworthy action and decrease to $\ln(t+1)$ otherwise. The question remains which actions should be considered trustworthy, which is complicated by the fact that agents have a range of actions to choose from, each being trustworthy to some extent. We therefore scale the exponential and logarithmic functions appropriately to accommodate those variations. What complicates matters further is that higher transfers are not necessarily more trustworthy. For instance, following a transfer of \$4 by Alice, a return of \$8 by Bob might be considered suspiciously high, and hence not trustworthy. On the other hand, a return of \$2 after a \$1 investment (which is also the maximal possible return) might well be interpreted as an encouraging sign of willingness to cooperate (due to lower stakes), especially if trust is initially low. This dynamics is captured by a function that starts as exponential, but turns into a logarithm once it crosses the $y = x$ line (two examples are displayed in Figure 5). The crossing point is chosen according to the proportion of the (doubled) investment returned by Bob (higher proportion means lower crossing point). Another consideration is which actions should lead to a change in whose trust; Bob's return is a sign of his trustworthiness and should cause a revision of Alice's trust, but Alice's investment has a different nature - it reflects the trust she has towards Bob, but not necessarily her own trustworthiness. We mention these complexities to give the reader an idea of what is involved in designing a dynamics model, but refer to the code base to see how exactly we tackled those challenges.

MK: is it given in the Appendix?

MO: not yet... I will remove this for now and hopefully add later if I put that in appendix. or maybe i should just say that those algorithms are available in the codebase?

MK: if there is no time then just say 'omitted for space reasons but can be inferred from the code'

Trust Evaluation The second important component of the mental reward structure is the trust evaluation function $\omega_1 : \mathrm{FPath} \rightarrow \mathrm{Ags} \rightarrow [0,1]$, which expresses how much an agent *actually* trusts their opponent. The difficulty in formally defining trust is that, even on the informal level, no single, widely-accepted formulation exists. Moreover, there are various types of trust and its interpretation often depends on the context in which it is used. In this work, we focus on social trust and draw inspiration from the work of Russell Hardin [10], who formulates trust as encapsulated interest, noting that people are incentivised to maintain relationships which are beneficial to them in the long-term. To show commitment, one often has to go out of one's way to help another achieve their goal, hoping to strengthen, or at least preserve, the relationship. In the context of our example, the mental component of the utility function represents agent's dedication to reinforcing that relationship and this commitment is measured by a goal coefficient λ_τ^A (for $A \in \text{Ags}$), which effectively defines *trustworthiness* τ of an agent (so $\tau_A = \lambda_\tau^A$). With that, trust (of A towards B) is simply (A's) expectation of (B's) trustworthiness, given by $\xi_{A,B} = \mathbb{E}_A[\tau_B]$, where the expectation is computed with respect to A's belief at some point of a game. Given a path $\rho \in \text{FPath}$, A's trust towards B is captured by $\omega_1(\rho, A) = \mathbb{E}_A^\rho[\tau_B]$

Experiments To validate our proposed setup, we use the tool to predict the behaviour of a collection of agents to see whether it matches our expectations. We then use synthetic data to learn agents' parameters and predict their future behaviour. We take a following list of hypotheses to guide our experiments:

(H1) There exists a configuration of agent parameters under which cooperation is predicted by the framework
(H2) As initial trust between agents increases, so does the social welfare of the game
(H3) Untrustworthy individuals who try to trick their opponent into trusting them are punished

The first hypothesis (H1) serves as a sanity check for the framework - it must hold or our approach is deemed to fail. The second hypothesis (H2) hopes to confirm a widely held conviction that high trust is beneficial to society - in this case, measured by average size of a transfer in Trust Game. Finally, third hypothesis asserts that fair, trustworthy agents can detect when their opponent attempts to take advantage of their good intentions to make monetary profit and they do not cooperate with such con men.

In the experiments described below, we focus on the way agents' characteristics and initial beliefs influence their behaviour. Therefore, we fix agents' meta-parameters by assuming they're almost perfectly rational ($\alpha = 100$), have a limited lookahead ($\beta = 2$) and discount future rounds ($\gamma = 0.8$). We furthermore assume that agents estimate their opponent's meta-parameters accurately. We also reduce initial mental estimations (i.e., estimations of opponent's trust) to their expectations.

First of all, H1 holds; it suffices to make both agents trusting (i.e., their beliefs should be biased towards λ_τ) and trustworthy (i.e., $\lambda_\tau > \lambda_m$) for cooperation to arise. This is clear given the analysis below.

To verify H2, we fix initial beliefs of agents while varying their characteristics and initial mental estimations and simulate execution of ten rounds of Trust Game. We use five different values of initial beliefs which give rise to five different values of initial trust (0.2, 0.4, 0.5, 0.6 and 0.8). For each, we perform twenty simulations, each with randomly chosen goal coefficients and trust estimations of both agents. We are interested in mean investment and mean return corresponding to each value of initial trust. The results confirm our hypothesis and are plotted in Figure 4. We note a steep increase in the plot between trust values 0.5 and 0.6, suggesting that if full trust is not possible, one should at least strive to keep its value above 0.6 to benefit the society. This principle could be used by robots and other autonomous agents to guide their interactions with humans.

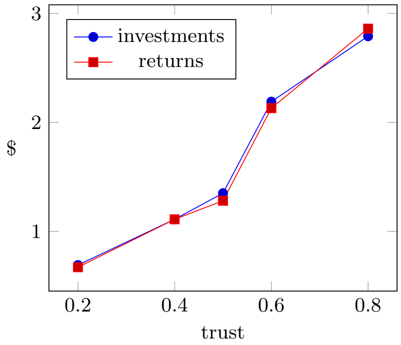


Fig. 4: Investments, returns

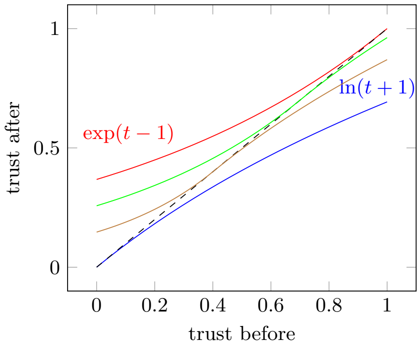


Fig. 5: Trust dynamics functions

Finally, to verify H3, we note that an untrustworthy individual is naturally modelled in the framework by setting $\lambda_m = 1$. However, a true con man (playing as Bob) might try to trick their opponent into trusting them by initially cooperating (particularly if initial investments are low) and then defecting (i.e., keeping all the profits for themselves) once investments become high enough. However, such strategy can only profit the con man if he times his defection to coincide with the end of the game. To enable that, we assume he knows how many rounds the game will last. Moreover, we equip the con man with a more simplistic trust dynamics model – he assumes that the higher his return, the greater the increase in Alice's trust towards him. Finally, the con man assumes that Alice's behaviour is driven by her trust – the more she trusts him, the more she will be willing to invest.

To see what happens when a cognitive agent faces a con man, we randomise Alice's goal coefficients, her belief and her initial estimation of Bob's trust and equip Bob with a high lookahead (β_5) to model his strategic thinking. We then simulate execution of the trust game for three rounds. We repeat this twenty times, ignoring the runs where no transfers arise (when Alice's trust is too low for her to invest). Bob's strategy is indeed as expected, with very cooperative behaviour in the first two rounds and defection in the last move. However, the data shows that in majority of cases Alice does not let herself be manipulated by Bob, as she finishes the game with more money than him seven times (he comes out on top twice). She earns on average \$9.63, compared with Bob's \$5.9. Our hypothesis is therefore partially confirmed – a variety of behaviours is observed, most of which are favourable for the cooperative agent.

MK: expand on H3

5.2 Tipping

The next case study we analyse has to do with tipping (i.e., gratuity). In particular, our aim is to formalise the cognitive process that underlies a decision of how much to tip. This, in turn, allows us to infer various statistics, such as tipping norms in different countries, based on behavioural data.

Before presenting the details of the model, we make the following observations. First of all, tipping customs differ between cultures - what would be a very generous tip in Europe may qualify as an insult in America. We hypothesise that each country/region/community can be assigned a baseline amount which characterises the social norm that dictates how much to tip having received a service of standard quality. Second, by tipping less than imposed by the local custom, an agent exposes themselves to feelings of guilt, which features as a mental attitude in our model. Finally, each agent's attachment to money differs and affects how much they're willing to tip.

Game Setup Tipping is modelled as a two-person game between a service provider (we'll call them Ben) and service receiver (called Abi). Ben provides a service, be it waiting at a restaurant, a taxi ride or valeting, which we assume can be of *low*, *medium* or *high* quality. This is followed by Abi giving a tip, which ranges between 5% and 30% (we only consider multiples of 5 for simplicity). Figure 6 gives a graphical representation of one iteration of the tipping game; it is easy to deduce standard components of the model, such as the transition function T and set of actions Act. As usual, states encode execution histories (which allows us to use states where paths are expected) so, for example:

$$s_1 = ([med], []),$$
$$s_6 = ([low], [20]).$$

Also, if Ben provides service of high quality from state s_8 then the game would transition into state $([med, high], [5])$. A sole physical reward structure $\mathrm{R_m}$ models the monetary exchange associated with tipping with a state reward function $\mathrm{r}^{\mathbf{s}}$ that assigns $-a$ to Abi and a to Ben in a state where Abi has just given a tip a. Note that an action reward would be more natural here, but mental rewards are collected at states and dealing with only one type of rewards simplifies analysis.

Mental Reward As mentioned above, we hypothesise that guilt features as a factor when tipping. Note that, unlike Trust Game, where an agent is driven by their opponent's trust, guilt is experienced by the agent themselves. Also, while in the Trust Game the behaviour of both agents is of interest, here we mostly concern ourselves with modelling Abi's behaviour; we assume Ben always does his best, but quality of service provided varies due to randomness. Therefore, our main task is to capture how much guilt, denoted η^{g}_{Abi}, Abi experiences depending on how high her tip is. We assume that Abi's guilt is primarily influenced by two factors: (i) her guilt proneness, measured according to GASP (Guilt And Shame Proneness) scale [3] as a number between 1 and 7, and (ii) her estimation

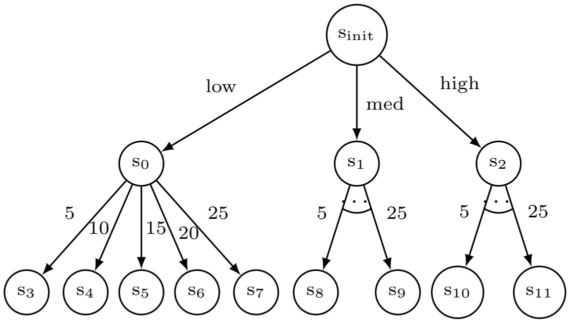


Fig. 6: Tipping model

of Ben's expectation of the tip, which we assume is a function of service quality. In particular, based on perceived quality of service received and a tipping norm, Abi estimates how high her tip should be. Tipping less than that reference value causes Abi to feel guilty ($[\![\eta^g_{\text{Abi}}]\!] < 0$) and the smaller the tip, the more guilt she experiences (where the pace of guilt increase is exponential, but the rate is dependent on Abi's GASP score). Conversely, tipping higher than the reference value will invoke "negative guilt", i.e., pride ($[\![\eta^g_{\text{Abi}}]\!] > 0$), but the absolute values will be smaller than for guilt. With that, Abi's utility function captures her attachment to money and the guilt she would experience after tipping less than expected:

$$\mathtt{cu}^{\mathbf{s}}_{\text{Abi}}(s) = \lambda^{\text{Abi}}_m \mathrm{r}^{\mathbf{s}}_{\text{Abi}}(s) + \lambda^{\text{Abi}}_g [\![\eta^g_{\text{Abi}}]\!]^s$$

Experiments We now describe how using our tool with simulated tipping data can provide insights about population-wide patterns. The idea is that human actions are driven primarily by their personalities, beliefs and desires, but societal norms play a role too. For example, a successful home assistance robot must be able to adapt its behaviour not only to the preferences of its owner(s) but also to the environment it operates in. In the context of our example, tipping norm constitutes such a societal norm.

In all the experiments, we are only interested in learning Abi's parameters (and the tipping norm) – Ben's behaviour is assumed to be random.

For the first experiment, we use three batches of synthetic data, each consisting of ten rounds of Tipping Game. Tips in the first file are generally low, tips is the second file are mostly medium and tips in the third file are generally high, as judged by a human. We perform inference on each of the data files, in which we learn Abi's goal coefficients, Abi's GASP score and the tipping norm. We take a (discretised) uniform distribution as prior, separate for each parameter we are inferring (with around five elements in the support of each prior). Note that computing disjoint posteriors is less accurate as it does not capture correlations between parameters, but it is easier to present and conceptualise. Figure 7 visualises inferences of Abi's goal coefficients; note that goal coefficients sum to 1, which explains why only points on one of the diagonals have positive probabilities. Reassuringly, the higher the tips, the more biased the posterior is towards λ_g^{Abi}. In other words, low tips are explained by agent's attachment to money, while high tippers are inferred to place more importance on their feelings. Figure 8 confirms that tipping is higher when tipping norm increases and that low tippers are less prone to experience guilt. Finally, when we performed the same experiment with fixed goal coefficients ($\vec{\lambda}^{\text{Abi}} = [0.5, 0.5]$) and GASP score (6), the inferred posterior over tipping norm was characterised by less uncertainty. We also note that the tipping norm should not be inferred solely based on the data, as one might be tempted to assume, because parameters of an agent play a role too. To illustrate it, we use the data characterised by medium tips, fix Abi's goal coefficients and her GASP score and infer the tipping norm. To improve accuracy we use a more informed prior – still uniform, but its support is restricted to the set $\{5, 7, 9, 11, 15\}$. The inferred posterior varies widely depending on the values of goal coefficients and GASP score that we fix. For example, when Abi is more money-oriented ($\vec{\lambda}^{\text{Abi}} = [0.9, 0.1]$) and not prone to experience guilt (GASP score equal to 1), the posterior is biased toward smaller tipping norm (5 being most likely), but with little certainty. With a more centralised GASP score (equal to 4) and lack of strong preference between guilt and money ($\vec{\lambda}^{\text{Abi}} = [0.5, 0.5]$), high tipping norm (15) is deemed most likely and the confidence of the prediction is higher. In-between values of tipping norm also feature as likely for other values of Abi's parameters.

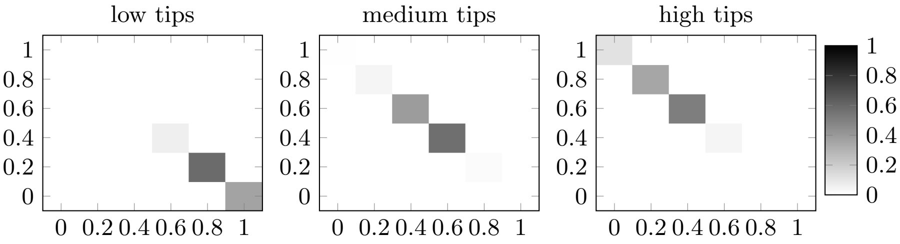


Fig. 7: Posterior predictive distribution over Abi's goal coefficients for three different batches of data; λ_m^{Abi} on x-axis, λ_g^{Abi} on y-axis

The other type of experiment we perform involves learning from data generated (through simulations) by the tool itself. For that, we randomly generate Abi's goal coefficients, her GASP score and tipping norm and simulate the execution for ten rounds. We then attempt to learn the randomly chosen parameters starting from a uniform (discretised) prior, the same as before. We repeat this process ten times and evaluate each posterior predictive distribution by computing its mean squared prediction error (MSPE), which we then aggregate (over ten runs) via mean and median averages.

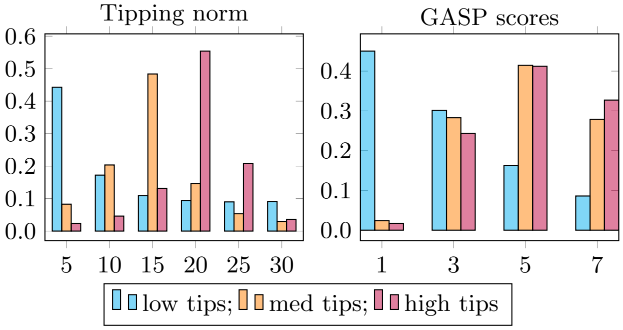


Fig. 8: Posterior predictive distribution over Abi's GASP score and tipping norm

Additionally, to study how the amount of data affects the performance of the inference, we repeat the above on data generated by simulating five and fifteen rounds of Tipping Game. The results are displayed in Table 3, separately for each inferred parameter. The leftmost column indicates the number of rounds that data used for inference consisted of. For reference, the first row of each table shows prediction errors computed from the prior, before any learning occurs. The results show that learning performs especially well for goal coefficients and relatively well for GASP score and tipping norm, especially in presence of more data. GASP score seems to be especially hard to learn, judging by the high variance of the results. Interestingly, ten rounds of data offer little or no advantage for learning compared to five rounds, suggesting one should aim to use at least fifteen rounds to maximise chances of effective inference.

Table 2: Prediction errors, aggregated over 10 runs

(a) goal coeffs

#	MSPE	
	mean	median
0	0.209	0.174
5	0.045	0.035
10	0.057	0.041
15	0.022	0.015

(b) GASP score

#	MSPE	
	mean	median
0	10	10
5	5.38	4.91
10	7.37	5.61
15	4.01	2.25

(c) tipping norm

#	MSPE	
	mean	median
0	107.5	112.5
5	105.33	102.82
10	69.18	68.55
15	53.06	37.39

5.3 Bravery Game

The next case study we analyse is inspired by an example from Geanakopolos et al [7] and serves as a comparison between psychological games and our model.

Table 3: Prediction errors, aggregated over 10 runs

(a) goal coefficients

#	$MSPE$		%
	mean	median	
5	0.045	0.035	6
10	0.057	0.041	4
15	0.022	0.015	4

(b) GASP score

#	$MSPE$		%
	mean	median	
5	5.38	4.91	2
10	7.37	5.61	4
15	4.01	2.25	5

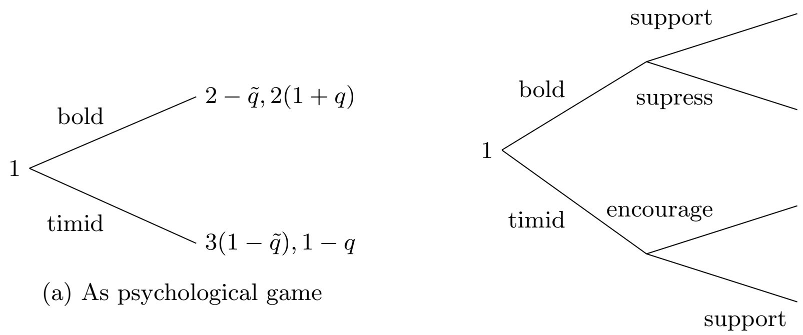
(c) tipping norm

#	$MSPE$		%
	mean	median	
5	105.33	102.82	4
10	69.18	68.55	5
15	53.06	37.39	6

The original scenario involves an agent (player 1) who makes a decision in front of his friends (player 2). He prefers to be *timid*, but he does not want to disappoint his friends, who may expect him to act *boldly*. Beliefs of players are captured by letting p be the probability of player 1 taking action *bold*, q be player 2's expectation of p (first-order belief) and $\tilde{q}$ be player 1's expectation of q (second-order belief). The game and agents' preferences are summarised in Figure 9a (in the style of the original paper); note that utility functions are defined in terms of beliefs, a crucial feature of psychological games framework. For example, the more player 1 thinks his friends expect him to be bold (reflected by a value of $\tilde{q}$ close to 1), the more likely he is to act boldly. Note that player 2 not only prefers their friend to act boldly - they moreover prefer to think of him as bold. While we do not explain the workings of psychological games framework in detail here (we refer interested readers to the original publication), we note that three distinct equilibria are predicted for this game: one in which player 1 acts boldly ($p = 1$), one in which he acts timidly ($p = 0$) and one where he randomises ($p = 0.5$). We note also that a crucial assumption that gives rise to these three equilibria is that beliefs of players correspond to reality in equilibrium, i.e., $p = q = \tilde{q}$ in each of the above solutions.

We propose to use cognitive stochastic multiplayer game framework to conduct complementary analysis. We are interested in the expected behaviour of agents when their beliefs are *not* accurate and we wish to find out under what circumstances an equilibrium is reached and whether it is one of the equilibria predicted by psychological games framework. We therefore consider a repeated version of bravery game and, to enable player 1 making inferences about his friends' preferences, we modify the game to allow player 2 to *react* to player 1's decision. In particular, following a bold move, player 2 may *support* or *suppress* their friend's action, whereas after a timid decision, possible reactions are *encouragement* or *support*. A graphical representation of (one iteration of) the resulting game is displayed in Figure 9b. Note that the notion of equilibrium is not defined in our framework, but we informally say that a simulation has reached an equilibrium if, from some point onward, action distribution computed by agents and their beliefs do not change from one iteration to the next.

Before giving an overview of a CSMG model of the bravery game, we outline the major differences in how mental states of agents are formalised in our



(a) As psychological game

(b) As CSMG

Fig. 9: Bravery Game

framework and in psychological games [7]. Belief plays an important role in both models but its scope differs: our belief is dynamic and computable and it quantifies uncertainty over opponent's characteristics; their belief operates on strategies of other agents and does not support updating. Our model allows formulation of rich, parameterised, belief-dependent mental states, whereas in psychological games emotions *are* beliefs. Moreover, while psychological games introduce nested beliefs of arbitrary order, our framework proposes a heuristic approach of mental estimations that simplifies beliefs of higher orders.

To represent the standard, numerical (non belief-based) component of agents' utility from Figure 9a, we define two physical reward structures: $\mathrm{R_b}$ assigns a unit reward to both players in a state reached after *bold* move, while $\mathrm{R_t}$ assigns a unit reward after a *timid* move. With that, appropriately set goal coefficients allow us to model boldness or timidness of various degree. To represent belief-dependent components of agents' utility, we introduce a mental variable $\eta = \mathit{pride}$ and are particularly interested in pride_{p_2}, pride felt by player 2 upon player 1 making his decision. Note that we allow negative values of pride, interpreting them as shame. Since player 1 cares about what his friends think of him, he is motivated by maximising their pride, which he estimates according to a pride dynamics function. While we do not go into detail of estimation heuristics here, the intuition is that pride is estimated based on player 2's reactions, with *support* hinting at positive pride and other reactions suggesting a negative value (shame).

Utility functions of agents are as follows, where we use states where paths are expected under the assumption that states encode executions histories:

$$\begin{aligned}\mathtt{cu}^{\mathbf{s}}_{\mathrm{p}_1}(s) &= \lambda_b^{p_1}\mathrm{r}^{\mathbf{s}}_{\mathrm{b,p_1}}(s) + \lambda_t^{p_1}\mathrm{r}^{\mathbf{s}}_{\mathrm{t,p_1}}(s) + \lambda_p^{p_1}\mathbb{E}^s_{p_1}[\mathit{pride}_{p_2}],\\ \mathtt{cu}^{\mathbf{s}}_{\mathrm{p}_2}(s) &= \lambda_b^{p_2}\mathrm{r}^{\mathbf{s}}_{\mathrm{b,p_2}}(s) + \lambda_t^{p_2}\mathrm{r}^{\mathbf{s}}_{\mathrm{t,p_2}}(s) + \lambda_p^{p_2}\llbracket \mathit{pride}_{p_2}\rrbracket^s.\end{aligned}$$

Intuitively, player 1 is bold when $\lambda_b^{p_1} \gg \lambda_t^{p_1}$; in fact, we define his boldness as $\xi_{p_1} = \frac{2\lambda_b^{p_1}}{2-\lambda_p^{p_1}}$ and player 2's belief of player 1's boldness at state s is then $\mathbb{E}^s_{p_2}[\xi_{p_1}]$.

With that, player 2's pride is proportional to their belief of p_1's boldness in absolute terms, but when player 1 takes a timid move, pride is negative.

Experiments For the first set of experiments, we assume player 1 ($\alpha_{p_1} = 30$) is more rational than player 2 ($\alpha_{p_2} = 10$), as we expect a group of people to make decisions more noisily than an individual. Additionally, we assume players do not discount future events ($\gamma = 1$) and are rather short-sighted ($\beta_{p_1} = 1$, $\beta_{p_2} = 2$). Moreover, as an attempt to recreate the setting from the psychological game, we fix goal coefficients of agents as follows: $\vec{\lambda}^{p_1} = [0.2, 0.35, 0.45]$, $\vec{\lambda}^{p_2} = [0.4, 0.2, 0.4]$. We then vary initial beliefs of agents and simulate the execution of the system for twenty steps. We find that, regardless of the accuracy of initial beliefs, within a few iterations, the game reaches an equilibrium-like state characterised by dominance of *bold* actions followed by reactions of *support*. Interestingly though, this uniformity is disrupted by an occasional *suppress* reaction of player 2. We interpret it as complacency of player 2 who, having observed a streak of *bold* decision by their friend, assumes that this will continue in this manner whatever they do. In any case, our analysis suggests that, provided player 2 is given an opportunity to provide feedback following their friend's decisions, out of the three psychological equilibria the one characterised by $p = 1$ is superior.

Another experiment we perform shows that looking further into the future may produce better outcomes for everyone. In particular, we set goal coefficients of player 1 to $\vec{\lambda}^{p_1} = [0.35, 0.6, 0.05]$ so that he still prefers to be timid, but he now cares less about his friends' pride. We then simulate the execution for three different values of player 1's lookahead: $\beta_{p_1} = 1$, $\beta_{p_1} = 3$ and $\beta_{p_1} = 5$, and repeat this process five times to smooth out any probabilistic noise. Our findings, summarised in Table 4, show that by looking more into the future, player 1 can increase not only his, but also player 2's rewards gained along execution. Closer inspection reveals that longer lookahead allows player 1 to visualise growing pride of player 2 which outweighs player 1's natural preference for a timid move.

Overall, our treatment of Bravery Game serves as a hands-on comparison of our approach and psychological games framework. While both models stem from a desire to enrich the domain of utility functions, the way this is achieved differs. In our framework, emotions are captured by a complex mechanism that allows parameterisation and encoding of psychological theories, making it possible to capture intricacies of human feeling; in psychological games, emotions are reduced to beliefs about behaviour of others. Moreover, the CSMG framework is well-suited to capturing continuity of human preferences (via goal coefficients vectors), limits of our knowledge and the repeated, dynamic nature of our interactions. Conversely, the psychological games framework helps predict what behaviours are sustainable in the long-term, once beliefs of agents converge to reality.

Table 4: Rewards accumulated by each player over 20 steps of Bravery Game as a function of player 1's lookahead (averaged over five runs)

β_{p_1}	p_1	p_2
1	4.63	2.11
3	5.39	6.22
5	5.43	6.71

6 Conclusion

We have presented a novel model of human-like decision making that takes into account mental motivations of agents. We have turned the theoretical ideas into a probabilistic program and tested our tool on a number of case studies, showcasing its inference and prediction capabilities. Directions for further research include learning mental dynamics models from data, performing experiments with humans, such as Mechanical Turk, scaling up the implementation and investigating notions of mental equilibria.

Acknowledgements This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115).

References

1. Battigalli, P., Dufwenberg, M.: Dynamic psychological games. Journal of Economic Theory **144**(1), 1–35 (2009). https://doi.org/10.1016/j.jet.2008.01.004
2. Berg, J., Dickhaut, J., McCabe, K.: Trust, Reciprocity, and Social History. Games and Economic Behavior **10**(1), 122–142 (jul 1995). https://doi.org/10.1006/game.1995.1027
3. Cohen, T.R., Wolf, S.T., Panter, A.T., Insko, C.A.: Introducing the GASP Scale: A New Measure of Guilt and Shame Proneness. Journal of Personality and Social Psychology **100**(5), 947–966 (2011). https://doi.org/10.1037/a0022641
4. Evans, O., Stuhlmüller, A., Salvatier, J., Filan, D.: Modeling Agents with Probabilistic Programs. http://agentmodels.org (2017), accessed: 2021-5-2
5. Falcone, R., Castelfranchi, C.: Social Trust: A Cognitive Approach. Trust and Deception in Virtual Societies (January), 55–90 (2001). https://doi.org/10.1007/978-94-017-3614-5_3, http://link.springer.com/10.1007/978-94-017-3614-5{_}3
6. Gambetta, D.: Can We Trust Trust? In: Trust: Making and Breaking Cooperative Relations (1988). https://doi.org/10.2307/2076328
7. Geanakopolos, J., Pearce, D., Stacchetti, E.: Psychological games and sequential reciprocity. Games and Economic Behavior **1**, 60–79 (1989)
8. Goodman, N.D., Stuhlmüller, A.: The Design and Implementation of Probabilistic Programming Languages. http://dippl.org (2014), accessed: 2021-5-2

9. Goodman, N.D., Tenenbaum, J.B., Contributors, T.P.: Probabilistic Models of Cognition. http://probmods.org/v2 (2016), accessed: 2021-5-2
10. Hardin, R.: Trust and trustworthiness. Russell Sage Foundation (2002)
11. Huang, X., Kwiatkowska, M., Olejnik, M.: Reasoning about Cognitive Trust in Stochastic Multiagent Systems. ACM Transactions on Computational Logic **20**(4), 1–64 (jul 2019). https://doi.org/10.1145/3329123
12. Johnson, N.D., Mislin, A.A.: Trust games: A meta-analysis. Journal of Economic Psychology **32**(5), 865–889 (2011). https://doi.org/10.1016/j.joep.2011.05.007
13. Ong, D., Soh, H., Zaki, J., Goodman, N.: Applying Probabilistic Programming to Affective Computing. IEEE Transactions on Affective Computing pp. 1–1 (2019). https://doi.org/10.1109/taffc.2019.2905211
14. Scholl, B.J., Leslie, A.M.: Modularity, Development and 'Theory of Mind'. Mind and Language **14**(1), 131–153 (1999). https://doi.org/10.1111/1468-0017.00106

A Appendices

A.1 More Detail on Tic-tac-toe

Satisfaction dynamics Algorithm 1 outlines satisfaction dynamics model. Note that the numerical values such as 0.7 and 0.3 were chosen arbitrarily – for an informal example such as tic-tac-toe and child's satisfaction, when no research exists to guide the design, this is inevitable. For mental states that are studied extensively, such as trust, a dynamics model can be better informed, and, ideally, learned from data (whether using our framework or someone else's work).

Algorithm 1: Satisfaction dynamics

```
input  : state, action, agent
output: estimator's updated estimation of estimatee's satisfaction after
        action is taken in state
function updateSatEst(state, action, estimator, estimatee, satisfaction):
    /* only model parent's estimation of kid's satisfaction      */
    if estimator $\neq$ parent or estimatee $\neq$ child then
        return 0;
    /* determine whose turn it is at state                       */
    turn = OWNS(state);
    if turn = kid then
        if action is a fork then
            return MIN(1, satisfaction + 0.7)
        if action is a blunder then
            return MAX(1, satisfaction - 1)
        if action wins the game then
            return MIN(1, satisfaction + 0.3)
    else
        if action is a blunder then
            return MAX(-1, satisfaction - 0.5)
        if action wins the game then
            return MAX(-1, satisfaction - 0.3)
    return satisfaction;
```

Detailed Example of Decision Making Process This is a longer and somewhat more thorough explanation of how parent computes their action in a game of tic-tac-toe, illustarting the decision-making process assumed in the framework. Suppose the game is in state from Figure 2a, call it s_0, and it is parent's turn to move. Let ρ be the path taken to reach s_0. Let A_{s_0} be the set of actions available to the parent at s_0 – it contains six elements, displayed in Figure 2b. We are interested in computing $Prob^{\rho}(a)$ for each $a^p \in A_{s_0}$, i.e., the posterior predictive distribution over parent's actions. For simplicity, we assume the parent has a lookahead of 3 ($\beta_p = 3$), is perfectly rational ($\alpha_p \to \infty$) and does not discount future rewards ($\gamma_p = 1$). Further, we assume parent knows that the kid has slightly smaller lookahead ($\beta_k = 2$), bounded rationality ($\alpha_k = 5$) and substantial discounting ($\gamma_k = 0.7$), reflecting child's impatience. Taking meta-parameters of the child as fixed allows us to treat random variables from the decision making equations as values. We also assume that the parent is characterised by the following vector of goal coefficients: $\langle 0.1, 0.9 \rangle$ – child's satisfaction is their main priority. The probability we are after, for each $a^p \in A_{s_0}$, is given by the expression $P^{\rho,\rho,\beta_p,a^p}_{p,p}$, which we set out to compute below.

Given perfect rationality of the parent, softmax choice dictates that they will choose uniformly among the actions that yield maximal expected utility (usually there is only one such, but due to symmetry of tic-tac-toe, multiple actions may be equivalent). Equation 5 confirms this; hence we must find all $a^p \in A_{s_0}$ that maximise the value of $U^{\rho,\rho,3,\cdot}_{p,p}$. This generally involves evaluating this expression for each $a^p \in A_{s_0}$; in this case, one could take advantage of symmetry to reduce the size of A_{s_0} by half (e.g., we expect a^p_2 and a^p_5 to yield the same value). In any case, the process is the same for each action; we will focus on a^p_1. To compute $U^{\rho,\rho,3,a^p_1}_{p,p}$, we use Equation 3; letting $\delta_1 = \rho a^p_1 s_1$ (where s_1 is the state reached from s_0 when parent takes action a^p_1) and noting there are no action rewards in the game (so $\mathtt{cu}^{\mathbf{a}}_{\mathrm{p}}(\rho, a^p_1) = 0$), we get that $U^{\rho,\rho,3,a^p_1}_{p,p} = U^{\rho,\delta_1,2}_{p,p}$. Hence, parent's expected utility when taking action a^p_1 is equal to his expected utility in a state reached after taking that action, since there's no discounting and no action rewards.

To compute $U^{\rho,\delta_1,2}_{p,p}$, we use Equation 1, which involves computing probabilities of various actions taken by the child in state s_1. Let's call this set A_{s_1}; then our task is computing $P^{\rho,\delta_1,2,a^k}_{p,k}$ for $a^k \in A_{s_1}$. This time, rather than following decision making equations, we give an intuitive explanation of why the kid is most likely to choose action $a^k_1 = (2, 0)$, which can be considered the obvious choice as it denies a clear winning opportunity for the opponent. Indeed, child's expected utility corresponding to each available action in s_1 is a sum of their cognitive utility in the state reached after taking the action and their utility in a state reached after that, following a move by the parent. Now, the child believes that the parent wants to win the game, i.e., if a winning move is available, then the parent will take it. If no winning move is available for the parent, the child is clueless about parent's next action, as all states yield the same, zero utility (for the parent, computed by the child). Hence, according to child's calculations, every move that is not a^k_1 will be followed by a move a^k_1 by the parent. That loses the game for the child, yielding utility of -1, which, after discounting, gives $U^{\rho,\delta_1,2,a^k}_{p,k} = -0.49$ for all $a \in A_{s_1} \setminus a^k_1$. On the other hand, $U^{\rho,\delta_1,2,a^k_1}_{p,k} = 0$ as the parent has no winning moves after child plays a^k_1. Given bounded rationality of the kid, they will select the best action noisily and Equation 5 yields $P^{\rho,\delta_1,2,a^k_1}_{p,k} \approx 0.74$ and $P^{\rho,\delta_1,2,a}_{p,k} \approx 0.06$ for other actions.

Looking back at Equation 1 and noting that $\mathtt{cu}^{\mathbf{s}}_{\mathrm{p}}(\delta_1) = 0$ (kid's satisfaction does not change and game does not end at s_1), computing $U^{\rho,\delta_1,2}_{p,p}$ reduces to computing $U^{\rho,\delta_1,2,a^k}_{p,p}$ for each possible action $a^k \in A_{s_1}$ of the child (the same actions for which we have just computed probabilities, see Figure 2d).

Starting with a^k_1 and taking s_2 to be the state reached after a^k_1 (see Figure 2e) and δ_2 be the resulting path, $U^{\rho,\delta_1,2,a^k_1}_{p,p}$ simplifies to $U^{\rho,\delta_2,1}_{p,p}$ by Equation 3, which in turn is computed by adding parent's cognitive utility at s_2 and their cognitive utility at a state reached after parent takes their next action (by Equation 1). Now, the move a^k_1 creates a fork which, according to parent's heuristics of satisfaction dynamics, results in a satisfaction increase for the child. The new value is estimated as 0.7, hence $\mathtt{cu}^{\mathbf{s}}_{\mathrm{p}}(\delta_2) = 0.9 \cdot 0.7 = 0.63$. In state s_2, parent takes an action $(2,1)$ or $(1,3)$ (with equal probabilities) – formally we would use Equation 5 to compute it, but the intuition is that the parent must deny one of the winning opportunities (there are two in s_2, as a^k_1 is a fork), otherwise their move is a blunder and child's satisfaction decreases. Neither of the two actions changes the satisfaction gained by the kid, so parent's cognitive utility is the same as in the previous state. Therefore $U^{\rho,\delta_1,2,a^k_1}_{p,p} = U^{\rho,\delta_2,1}_{p,p} = 2 \cdot 0.63 = 1.26$.

Similar analysis can be performed for other actions from A_{s_1} and the results are summarised in Table 1a. Combining this with the probabilities computed before allows us to compute $U^{\rho,\rho,3,a^p_1}_{p,p} = U^{\rho,\delta_1,2}_{p,p} = \sum_{a^k \in A_{s_1}} P^{\rho,\delta_1,2,a^k}_{p,k} U^{\rho,\delta_1,2,a^k}_{p,p} \approx 1.05$. By symmetry, $U^{\rho,\rho,3,a^p_4}_{p,p} \approx 1.05$ also. The expected utilities corresponding to other actions (depicted in Table 1b are lower, although a^p_6 (and a^p_3, by symmetry) is preferable to a^p_5 (and a^p_2) – the intuitive reason is that a^p_6 gives the kid a chance to create a fork and win, while a^p_5 will lead to a draw unless one of the players blunders.

A.2 Framework Process Diagram

Figure 10 graphically represents the main components, inputs, outputs and the flow of our framework. The main functionality, *prediction*, requires specification of a game, agents' parameters (goal coefficients and meta-parameters) and their initial states (belief, mental estimations and meta-parameters estimations). With that, for any state, the framework specifies how to compute (and our tool implements this specification) a posterior predictive distribution on actions of an agent who owns that state. Typically, one starts with an initial state of the game, generates a posterior predictive distribution, samples an action from it and samples next state. The process then repeats starting from that state for some number of iterations. This is what we call *simulation* throughout the paper.

If parameters and initial states of agents are unavailable, but a record of their past interactions is at hand, the other functionality of the framework, *learning*, will come handy. It allows one to infer these unknown parameters and states given a game specification, a prior over the unknowns and a data file.

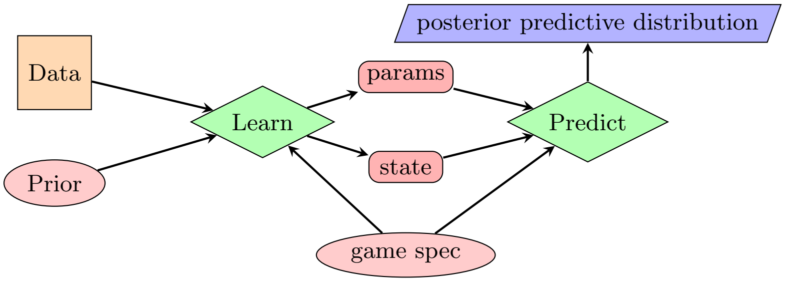


Fig. 10: Overall Process of the Framework