Summer Project Report

A study on Human Strategic Reasoning from the perspective of Game Theorists, Logicians and Cognitive Scientists

Under the guidance of Dr. Sujata Ghosh

ISI Chennai

Rishabh Bhonsle
IISER Bhopal
Roll No: 17217

July 24, 2019

Acknowledgements

I avail the opportunity to express my deep sense of gratitude to my mentor Dr. Sujata Ghosh of the Computer Science department, Indian Statistical Institute, Chennai for her guidance and teachings throughout my two month long summer project. I would also like to thank my project colleague Prasanna Bramhe, then sophomore at IISER Bhopal for his collaboration on this project. I would further like to thank my intern colleagues Tanvi Telang and Shashank Pathak from IISER Bhopal for providing insights on Propositional and First Order Logic which provided a better understanding for the topics within this project.

The atmosphere created at ISI Chennai was truly inspiring. From attending guest lectures to giving presentations, from studying in classrooms to discussions over coffee, every day felt like a unique mathematical journey as we unravelled, however so small, a part of its beauty. I am grateful for Dr. Sujata Ghosh for providing me with this unique opportunity.

Contents

1 Introduction **1**

2 Describing Game Trees and Strategies **2**
2.1 Extensive form . . . 2
2.2 Game Trees . . . 2
2.3 Strategies . . . 2
2.4 Partial Strategies . . . 3
2.5 Syntax for Extensive Form Game Trees . . . 3

3 Strategy Specifications **4**
3.1 Syntax . . . 4
3.2 Semantics . . . 4

4 Strategies and Types of Reasoning **6**
4.1 Formal Definitions . . . 6
4.2 Describing Types of Strategic Reasoning . . . 6
4.3 Further Study on Types of Strategic Reasoning . . . 7
4.3.1 Formalising Typologies - Cooperative and Competitive Strategies . . . 8
4.3.2 Formalising Typologies - Risk Taking and Risk Averse Strategies . . . 9
4.3.3 Conclusion . . . 9

5 Cognitive Modelling with PRIMS **10**
5.1 PRIMS Modules . . . 10
5.2 Visual Representation in PRIMS . . . 10
5.3 Translating the myopic and own-payoff models . . . 10
5.3.1 Model Definition . . . 10
5.3.2 Initial Memory Chunks . . . 10
5.3.3 Task Script . . . 10
5.3.4 Goals and Operators . . . 10
5.4 A General Translation Method . . . 11
5.4.1 Representing Games . . . 11
5.4.2 Representing Strategies . . . 11
5.4.3 Translation . . . 11
5.5 Further Study on Building Cognitive Models through the General Translation Method . . . 11
5.5.1 Belief Operator . . . 11
5.5.2 Negation Operator . . . 12

1 Introduction

Human strategic reasoning in turn-taking games has been extensively investigated by game theorists, logicians, cognitive scientists, and psychologists. Whereas game theorists and logicians use formal methods to formalise strategic behaviour, cognitive scientists use cognitive models of the human mind to predict and simulate human behaviour. In the present body of work, we first learn the game theorist's and logician's perspective of formalising strategic behaviour in these turn taking games and then we move on to see how a translation system which, starting from a strategy represented in formal logic, automatically generates a computational model in the PRIMs cognitive architecture. We model these strategic reasonings with the help of Dynamic perfect-information games. In dynamic games, both players take turns choosing an action, and both players can see which actions the other player has chosen in the past before they have to choose their next action. Whereas, in perfect-information games, both players know everything there is to know about the game - all possible actions and all possible outcomes. There are no hidden elements, and there are no chance elements. We represent these Dynamic perfect-information games by game trees.

2 Describing Game Trees and Strategies

The following definitions are as seen in the paper Strategic Reasoning: Building Cognitive Models from Logical Formulas by Sujata Ghosh, Ben Meijering and Rineke Verbrugge [1]

2.1 Extensive form

Extensive form games are a natural model for representing finite games in an explicit manner. In this model, the game is represented as a finite tree where the nodes of the tree correspond to the game positions and edges correspond to moves of players. For this logical study, we will focus on game forms, and not on the games themselves, which come equipped with players' payoffs at the leaf nodes of the games.We present the formal definition below. Let N denote the set of players; we use i to range over this set. For the time being, we restrict our attention to two player games, and we take $N = \{C, P\}$. We often use the notation i and ı to denote the players, where $C = P'$ and $P = C'$. Let Σ be a finite set of action symbols representing moves of players; we let a, b range over Σ. For a set X and a finite sequence$\rho = x_1x_2...x_m \in X$, let $\text{last}(\rho) = x_m$ denote the last element in this sequence.

2.2 Game Trees

Let $\mathbb{T} = (S, \Rightarrow, s_0)$ be a tree rooted at s0 on the set of vertices S and let $\Rightarrow$: $(S \times \Sigma) \rightarrow S$ be a partial function specifying the edges of the tree. The tree T is said to be finite if S is a finite set. For a node $s \in S$, let $\vec{s} = \{s' \in S | s \stackrel{a}{\Rightarrow} s' for some a \in \Sigma\}$. A node s is called a leaf node (or terminal node) if $s = \varnothing$.

An extensive form game tree is a pair $T = (\mathbb{T}, \widehat{\lambda})$ where $\mathbb{T} = (S, \Rightarrow, s_0)$ is a tree. The set S denotes the set of game positions with s_0 being the initial game position. The edge function $\Rightarrow$ specifies the moves enabled at a game position and the turn function $\widehat{\lambda} : S \rightarrow N$ associates each game position with a player. Technically, we need player labelling only at the non-leaf nodes. However, for the sake of uniform presentation, we do not distinguish between leaf nodes and non-leaf nodes as far as player labelling is concerned. An extensive form game tree $T = (\mathbb{T}, \widehat{\lambda})$ is said to be finite if T is finite. For $i \in N$, let $S^i = \{s | \widehat{\lambda}(s) = i\}$ and let $frontier(\mathbb{T})$ denote the set of all leaf nodes of T.

A play in the game T starts by placing a token on s_0 and proceeds as follows: at any stage, if the token is at a position s and $\widehat{\lambda}(s) = i$, then player i picks an action a which is enabled for her at s, and the token is moved to s' where $s \stackrel{a}{\Rightarrow} s'$. Formally a play in T is simply a path $\rho : s_0a_0s_1...$ in T such that for all $j > 0, s_{j-1} \stackrel{a_{j-1}}{\Rightarrow} s_j$. Let $Plays(T)$ denote the set of all plays in the game tree T.

2.3 Strategies

A strategy for player i is a function μ^i which specifies a move at every game position of the player, i.e. $\mu^i : S^i$. For $i \in N$, we use the notation μ to denote strategies of player i and τ to denote strategies of player $\bar{\imath}$. A strategy μ can also be viewed as a subtree of T where for each node belonging to player i , there is a unique outgoing edge and for nodes belonging to player $\bar{\imath}$, every enabled move is included. Formally we define the strategy tree as follows: For $i \in N$ and a player i's strategy $\mu : S^i \rightarrow \Sigma$, the strategy tree $T_\mu = (S_\mu, \underset{\mu}{\Rightarrow}, s_0, \widehat{\lambda}_\mu)$ associated with μ is the least subtree of T satisfying the following property:

- $s_0 \in S_\mu$
- For any non-leaf node $s \in S_\mu$
- if $\widehat{\lambda}(s) = i$ then there exists a unique $s' \in S_\mu$ and action a such that $s_\mu \underset{\mu}{\Rightarrow} s'$, where $\mu(s) = a$ and $s \stackrel{a}{\Rightarrow} s'$
- if $\widehat{\lambda}(s) \neq i$ then for all s' such that $s \stackrel{a}{\Rightarrow} s'$, we have $s_\mu \underset{\mu}{\Rightarrow} s'$.

– $\widehat{\lambda}_\mu = \widehat{\lambda}|S_\mu$

Let $\Omega^i(T)$ denote the set of all strategies for player i in the extensive form game tree T.

2.4 Partial Strategies

A partial strategy for player i is a partial function σ^i which specifies a move at some (but not necessarily all) game positions of the player, i.e. $\sigma^i : S^i \to \Sigma$.Let $\mathfrak{D}_{\sigma^i}$ denote the domain of the partial function σ^i. For $i \in N$, we use the notation σ to denote partial strategies of player i and π to denote partial strategies of player $\bar{i}$. A partial strategy σ can also be viewed as a subtree of T where for some nodes belonging to player i , there is a unique outgoing edge and for other nodes belonging to player i as well as nodes belonging to player $\bar{i}$, every enabled move is included.

A partial strategy can be viewed as a set of total strategies. Given a partial strategy tree $T_\sigma = (S_\sigma, \underset{\sigma}{\Rightarrow}, s_0, \widehat{\lambda}_\sigma)$ for a partial strategy σ for player i, a set of trees $\widehat{T}_\sigma$ of total strategies can be defined as follows. A tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ if and only if

– if $s \in S$ then for all $s' \in \vec{s}$, $s' \in S$ implies $s' \in S_\sigma$

– if $\widehat{\lambda}(s) = i$ then there exists a unique $s' \in S$ and action a such that $s \overset{a}{\Rightarrow} s'$.

Note that $\widehat{T}_\sigma$ is the set of all total strategy trees for player i that are subtrees of the partial strategy tree T_σ for i. Any total strategy can also be viewed as a partial strategy, where the corresponding set of total strategies becomes a singleton set.

2.5 Syntax for Extensive Form Game Trees

The syntax for specifying finite extensive form game trees is given by:

$$\mathbb{G}(Nodes) ::= (i, x) | \Sigma_{a_m \in J}((i, x), a_m, t_{a_m})$$

where $i \in N$, $x \in Nodes$, $J(finite) \subset \Sigma$, and $t_{a_m} \in \mathbb{G}(Nodes)$.

Given $h \in \mathbb{G}(Nodes)$, we define the tree T_h generated by h inductively as follows:

Define $T_h = (S_h, \underset{h}{\Rightarrow}, \widehat{\lambda}_h, s_x)$ where

- $S_h = \{s_x\} \cup S_{T_1} \cup \ldots S_{T_k}$
- $\widehat{\lambda_h}(s_x) = i$ and for all j , for all $s \in S_{T_j}, \widehat{\lambda_h}(s) = \widehat{\lambda_j}(s)$
- $\underset{h}{\Rightarrow} \bigcup_{j:1 \leqslant j \leqslant k}(\{(s_x, a_j, s_{j,0})\} \cup \underset{a}{\Rightarrow})$.

Given $h \in \mathbb{G}(Nodes)$, let $Nodes(h)$ denote the set of distinct pairs (i, x) that occur in the expression of h.

3 Strategy Specifications

For any countable set X, let $BPF(X)$ (the boolean, past and future combinations of the members of X) be sets of formulas given by the following syntax:

$$BPF(X) ::= x \in X | \neg\Psi | \Psi 1 \vee \Psi 2 | \langle a^+ \rangle \Psi | \langle a^- \rangle \Psi$$

where $a \in \Sigma$, a countable set of actions.

The formula $\langle a^+ \rangle \Psi$ (respectively, $\langle a^- \rangle \Psi$) refers to one step in the future (respectively, past). It asserts the existence of an edge a after (respectively, before) which Ψ holds. The 'time free' fragment of $BPF(X)$ is formed by the boolean formulas over X. We denote this fragment by $Bool(X)$.

For each $h \in \mathbb{G}(Nodes)$ and $(i, x) \in Nodes(h)$, we now add a new operator $\mathbb{B}_h^{(i,x)}$ to the syntax of $BPF(X)$ to form the set of formulas $BPF_b(X)$. The formula $\mathbb{B}_h^{(i,x)}\Psi$ can be read as "in the game tree h, player i believes at node x that Ψ holds".

3.1 Syntax

Let $P_i = \{p_0^i, p_1^i, ...\}$ be a countable set of observables for $i \in N$ and $P = \bigcup_{i \in N} P^i$. To this set of observables we add two kinds of propositional variables $(u_i = q_i)$ to denote 'player i 's utility (or payoff) is q' and $(r \leqslant q)$ to denote that 'the rational number r is less than or equal to the rational number q'. The syntax of strategy specifications is given by:

$$Strat^i(P^i) ::= [\Psi \mapsto a]^i | \eta 1 + \eta 2 | \eta 1 \cdot \eta 2$$

where $\Psi \in BPF_b(P^i)$.

3.2 Semantics

Let $M = (T, \rightarrow_x^i, V)$ with $T = (S, \Rightarrow, s_0, \widehat{\lambda}, \mathcal{U})$, where $(S, \Rightarrow, s_0, \widehat{\lambda})$ is an extensive form game tree, $\mathcal{U} : frontier(T) \times N \rightarrow Q$ is a utility function. Here, $frontier(T)$ denotes the set of leaf nodes of the tree T. For each $s_x \in S$ with $\widehat{\lambda}(s_x) = i$, we have a binary relation $\rightarrow_x^i$ over S. Finally, $V : S \rightarrow 2^P$ is a valuation function. The truth value of a formula $\Psi \in BPF_b(P)$ at the state s, denoted $M, s \models \Psi$, is defined as follows:

- $M, s \models p$ iff $p \in V(s)$.
- $M, s \models \neg\Psi$ iff $M, snot \models \Psi$.
- $M, s \models \Psi 1 \vee \Psi 2$ iff $M, s \models \Psi 1$ or $M, s \models \Psi 1$.
- $M, s \models \langle a^+ \rangle \Psi$ iff there exists an s' such that $s \stackrel{a}{\Rightarrow} s'$ and $M, s' \models \neg\Psi$.
- $M, s \models \langle a^- \rangle \Psi$ iff there exists an s' such that $s' \stackrel{a}{\Rightarrow} s$ and $M, s' \models \neg\Psi$.
- $M, s \models \mathbb{B}_h^{(i,x)}$ iff the underlying game tree of T_M is the same as T_h and for all s' such that $s \rightarrow_x^i s', M, s \models \Psi$.

The truth definitions for the new propositions are as follows:

- $M, s \models (u_i = q_i)$ iff $\mathcal{U}(s, i) = q_i$.
- $M, s \models (r \leqslant q)$ iff $r \leqslant q$, where r, q are rational numbers.

Strategy specifications are interpreted on strategy trees of T. We also assume the presence of two special propositions $turn1$ and $turn2$ that specify which player's turn it is to move, i.e. the valuation function satisfies the property

- for all $i \in N$, $turni \in V(s)$ iff $\widehat{\lambda}(s) = i$.

One more special proposition $root$ is assumed to indicate the root of the game tree, that is the starting node of the game. The valuation function satisfies the property

– $root \in V(s)$ iff $s = s_0$.

The semantics of the strategy specifications are given as follows. Given a model M and a partial strategy specification $\eta \in Strati(P^i)$, we define a semantic function $[\![\cdot]\!]_M : Strat_i(P^i) \to 2^{\Omega^i(T_M)}$, where each partial strategy specification is associated with a set of total strategy trees and $\Omega^i(T)$ denotes the set of all player i strategies in the game tree T. For any $\eta \in Strat^i(P^i)$, the semantic function $[\![\cdot]\!]_M$ is defined inductively:

– $[\![[\Psi \mapsto a]^i]\!]_M = \Upsilon \in 2^{\Omega^i(T_M)}$ satisfying: $\Upsilon \in \mu$ iff μ satisfies the condition that, if $s \in S_\mu$ is a player i node then $M, s \models \Psi$ implies $out_\mu(s) = a$.

– $[\![\eta_1 + \eta_2]\!]_M = [\![\eta_1]\!]_M \cup [\![\eta_2]\!]_M$

– $[\![\eta_1 \cdot \eta_2]\!]_M = [\![\eta_1]\!]_M \cap [\![\eta_2]\!]_M$

Above, $out_\mu(s)$ is the unique outgoing edge in μ at s.

4 Strategies and Types of Reasoning

We now study the formal definitions as seen in Studying strategies and types of players: experiments, logics and cognitive models by Sujata Ghosh and Rineke Verbrugge [2]

4.1 Formal Definitions

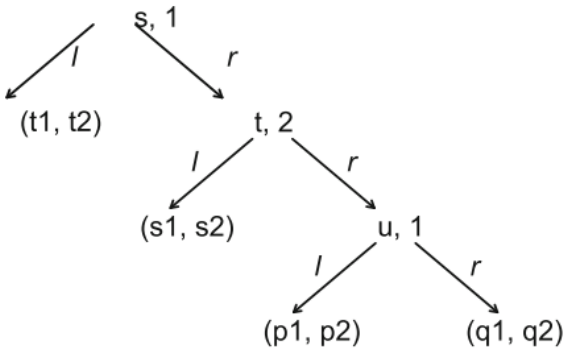

Figure 1: Visualising Payoffs

Let us assume that actions are part of the observables, that is, $\Sigma \subset P$. The semantics for the actions can be defined appropriately. Let $n1, ..., n4$ denote the four decision nodes of Game 1 of Fig. 2, with C playing at n1 and n3, and P playing at the remaining two nodes n2 and n4. We have four belief operators for this game, namely two per player. We abbreviate some formulas that describe the payoff structure of the game:

$\alpha := [\![d]\!][\![f]\!][\![h]\!]((u_C = p_C) \wedge (u_P = p_P))$ (from the current node, a d move followed by an f move followed by an h move lead to the payoff (p_C, p_P))

$\beta := [\![d]\!][\![f]\!][\![g]\!]((u_C = q_C) \wedge (u_P = q_P))$ (from the current node, a d move followed by an f move followed by a g move lead to the payoff (q_C, q_P))

$\gamma := [\![d]\!][\![e]\!]((u_C = r_C) \wedge (u_P = r_P))$ (from the current node, a d move followed by an e move lead to the payoff $(r_C, r\mathrm{P})$)

$\delta := [\![c]\!]((u_C = s_C) \wedge (u_P = s_P))$ (from the current node, a c move leads to the payoff (s_C, s_P))

$\chi := [\![b^-]\!][\![e]\!]((u_C = t_C) \wedge (u_P = t_P))$ (the current node can be accessed from another node by a b move from where an a move leads to the payoff (t_C, t_P))

Now we can define the conjunction of these five descriptions:

$$\Phi := \alpha \wedge \beta \wedge \gamma \wedge \delta \wedge \chi$$

Let Ψ_i denote the conjunction of all the order relations of the rational payoffs for player i ($\in \{P, C\}$) given in Fig 1.

4.2 Describing Types of Strategic Reasoning

We now analyse different formulas corresponding to the type of Strategic Reasoning based on the analysis of Forward Induction, Backward Induction, Myopic Player, Own-Payoff Player, Latent Class Analysis and Theory of Mind. These formulas can be found in [2]. These formulae are derived from Games $1, 2, 3, 4, 1'$ and $3'$ of figure 2 and would be frequently used.

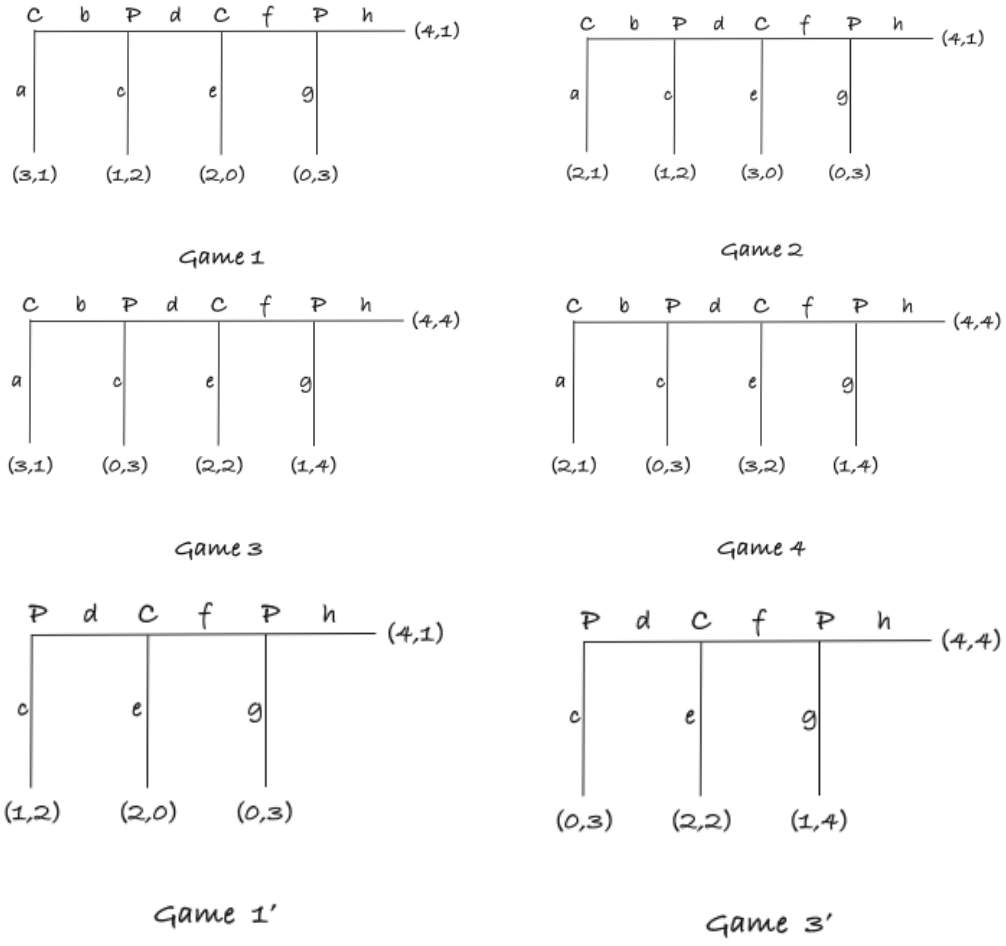

Figure 2: Games

We state the formulas for the Myopic ($\mathcal{K}$) and Own-Payoff Model ($\mathcal{X}$) for Games $1'$ and $3'$ below as examples and would also be referenced in the upcoming sections.

$\mathcal{K}_P^{1'} : [(\delta_{P,1'} \wedge \gamma_{P,1'} \wedge (1 \leqslant 2) \wedge root) \mapsto c]^P$

$\mathcal{K}_P^{3'} : [(\delta_{P,3'} \wedge \gamma_{P,3'} \wedge (1 \leqslant 2) \wedge root) \mapsto c]^P$

$\mathcal{X}_P^{1'} : [(\alpha P, 1' \wedge \beta P, 1' \wedge \delta_{P,1'} \wedge \gamma_{P,1'} \wedge (1 \leqslant 2) \wedge (2 \leqslant 4) \wedge (2 \leqslant 3) \wedge root) \mapsto d]^P$

$\mathcal{X}_P^{3'} : [(\alpha P, 3' \wedge \beta P, 3' \wedge \delta_{P,3'} \wedge \gamma_{P,3'} \wedge (1 \leqslant 2) \wedge (2 \leqslant 4) \wedge root) \mapsto d]^P$

4.3 Further Study on Types of Strategic Reasoning

Based on the paper Cross-cultural differences in playing centipede-like games with surprising opponents by Sujata Ghosh, Rineke Verbrugge, Harmen de Weerd and Aviad Heifetz [3], we further extend our study to develop formulae for four more types of Strategic Reasoning based on the degree of risk-taking and cooperativeness. To gain a perspective on the complexity involved in modelling risk, refer chapters 1 and 2 of

The Psychology of Risk Taking Behaviour [4]. We thus classify our players as Risk Taker, Risk Averser, Cooperative and Competitive.

Experiments were conducted with three sets of 50 students each set being of different cultural background and different nationality. The participants played the turn based games described above through a graphical interface on the computer screen similar to Figure 3. The computer $(C, blue)$ controls the blue trapdoors and acquires blue marbles (represented as dark grey in a black and white print) as pay-offs, while the participant $(P, orange)$ controls the orange trapdoors and acquires orange marbles (light grey in a black and white print) as pay-offs. The participant's goal was that the marble should drop into the bin with as many orange marbles as possible. The computer's goal was that the marble should drop into the bin with as many blue marbles as possible. The participant's were asked about the move they would play at node 2 and reasoning behind the move. The responses were recorded and then formalised into strategy specifications based on the reasoning given for their strategies.

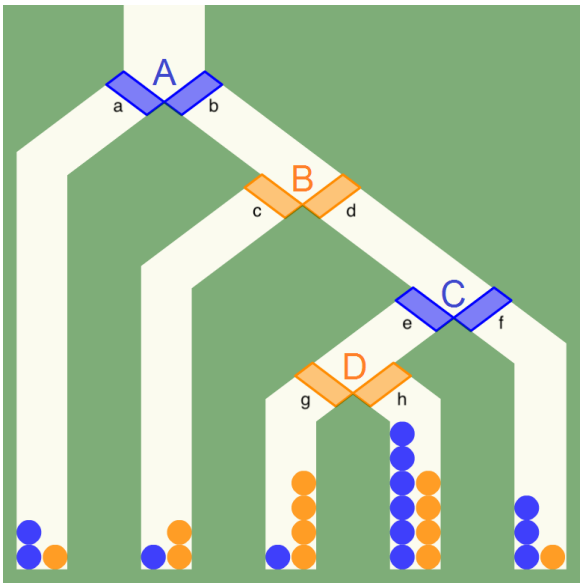

Figure 3: Games

Typologies of the players were decided according to game edges chosen by the player and were classified as follows- The edge that gave the same payoff to the player and higher to the computer was considered a cooperative choice and vice versa was considered competitive choice. The choice to stop the game at node two was considered a risk averse choice and to continue was considered a risk takers choice breaking down of risk awareness into further categories was done based on the answers given by participants about their reasoning.

4.3.1 Formalising Typologies - Cooperative and Competitive Strategies

Cooperative Strategy: At Node D

$$\mathcal{L}_{n_4} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge (1 \leqslant 6)) \mapsto h\}]^P$$

Competitive Strategy: At Node D

$$\mathcal{L}_{n_4} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge (1 \leqslant 6)) \mapsto g\}]^P$$

4.3.2 Formalising Typologies - Risk Taking and Risk Averse Strategies

We define the following functions for our purpose.

$HP(n_k)$: For any current node (n_k), HP of that node gives the highest pay-off of the player amongst all the nodes following that node.
$LP(n_k)$: For any current node (n_k), LP of that node gives the lowest pay-off of the player amongst all the nodes following that node.
$CP(n_k)$: Gives the payoff of the current node (n_k).
$DP(n_k)$: For any current node (n_k), DP of that node gives the ratio of the differences between CP and LP and current CP and HP.
$H/L(n_k)$: Ratio of the number of nodes with higher payoff than current node (n_k) to that of payoffs lower than the current node.

Type 1: HP Type risk taker
$\mathcal{N}_{n_2} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge HP(n_2) \geqslant CP(n_2) \wedge \mathbb{B}_{g4}^{(n3,c)} \langle d \rangle e) \mapsto d\}]^P$

Type 2: DP Type risk taker
$\mathcal{N}_{n_2} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge DP(n_2) \geqslant 1 \wedge \mathbb{B}_{g4}^{(n3,c)} \langle d \rangle e) \mapsto d\}]^P$

Type 3: Node Counter Type Risk Taker
$\mathcal{N}_{n_2} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge H/L(n_2) \geqslant 1 \wedge \mathbb{B}_{g4}^{(n3,c)} \langle d \rangle e) \mapsto d\}]^P$

Type 4: Risk Averser
$\mathcal{N}_{n_2} : [(\Phi \wedge \Psi_P \wedge \Psi_C \wedge CP(n_2) \geqslant LP(n_2)) \mapsto c\}]^P$

4.3.3 Conclusion

The number of participants subscribing to a particular risk strategy has been tabulated below:

Strategy	Participants
Competitive	16
Cooperative	20
DP type	5
HP type	20
Node counter type	5
EFR type risk taker	7

The number of people who were unable to decide at the last node were far fewer than those who chose either competitive or cooperative which had similar frequencies.

The HP type risk takers were the highest amongst the risk takers and had a significant number of them thought of the computer being a coin toss opponent.

Very few individual chose a risk averse option. This can be due to the specificity of the relative payoffs or loss gain ratios and hence it cannot be considered a general trend without analysing further.

5 Cognitive Modelling with PRIMS

PRIMs, short for primitive information processing elements, also known as ACTransfer, is a cognitive architecture. Broadly speaking, PRIMs is a system that models the human mind as it performs experimental tasks. PRIMs can be used to predict reaction times, decisions, and neural activity.

5.1 PRIMS Modules

Specific areas in the human brain have specific functions, as well as corresponding input and output. Because PRIMs models the human mind, it also has a set of modules with specific functions: a visual module, a manual module, a working memory module, a declarative memory module and a task control module. Different modules can work in parallel, but within a single module only one thing can be done at a time. Modules communicate with each other through their buffers. Each buffer has a number of slots (temporary storage locations that can hold a single piece of information, such as a number or a word) that can be used to exchange information with other buffers. All the buffers together are the system's global workspace. Game specific information about the same can be found in Jordi Top's Thesis on Bridging the gap between logic and cognition: a translation method for centipede games [5]. Given below is an overview of this cognitive architecture.

5.2 Visual Representation in PRIMS

A PRIMs file contains a model, which is comprised of goals and operators, and a script, which runs the experiment and sends visual input, such as feedback, to the visual module of the PRIMs model. Visual input is sent to a PRIMs model using the function screen.

5.3 Translating the myopic and own-payoff models

The myopic and own-payoff models implemented in PRIMs in Ghosh & Verbrugge (online first) [1] are composed of four components: the model definition, the initial chunks in declarative memory, the script that runs the task and the set of goals and operators. The model definition and the initial chunks in declarative memory are the same for both models. The script that runs the task and the specific operators differ between both models.

5.3.1 Model Definition

Each PRIMs model has a model definition which is used to define which goals are initially in the goal buffer, which task constants exist, and what values each PRIMs parameter has. If a PRIMs parameter is not set within the model definition, it is set to the PRIMs default value, if a default exists.

5.3.2 Initial Memory Chunks

The initial chunks in declarative memory are also the same for the myopic and own-payoff models. A PRIMs model can have two scripts: a main script that runs the task, and an initialisation script, which is run once at the start of a model run. The initial chunks in declarative memory are specified in the initialisation script.

5.3.3 Task Script

A PRIMs model always contains a script that is used to determine what should be displayed onscreen for the task at hand, and how the task should respond to the model's actions.

5.3.4 Goals and Operators

The Goals and Operators defined corresponding to the strategy specification for a PRIMS model performs various actions as mentioned.

5.4 A General Translation Method

This general translation system takes logical formulae from the logic presented in Ghosh & Verbrugge (online first) [1] and automatically generate PRIMs models from them. The logical formulae, specify a list of conditions that should all hold if some action is to be played. If at least one of them does not hold, the formula allows for any action. The PRIMs models we create have the same properties as the logical formulae.

5.4.1 Representing Games

We create our system in Java 1.8. We use six Java classes to create game trees. They are the following: Game, Treeobject, Edge, Node, Nonleaf and Leaf. Each of these classes has name as an instance variable, which is used to identify the object. Instance variables are properties of the objects created using this class.

5.4.2 Representing Strategies

We have Java Classes for each of it's propositions. Their class names are, respectively, Root, Turn, Utility, Comparison, and Action.

5.4.3 Translation

This process remains the same as above and is divided into four components: the model definition, the initial chunks in declarative memory, the script that runs the task and the set of goals and operators. Along with these the propositions are also sorted in order to check the truth value of each efficiently.

5.5 Further Study on Building Cognitive Models through the General Translation Method

All the operators used in the Myopic and Own-Payoff Model have been defined in the Appendix D of Jordi Top's Thesis on Bridging the gap between logic and cognition: a translation method for centipede games [5].

We further attempt to build the operators for belief and negation to expand the scope of modelling strategies. We use the same Game (Game 1) and Code as defined in Appendix D of Jordi Top's Thesis on Bridging the gap between logic and cognition: a translation method for centipede games [5] to build these operators.

5.5.1 Belief Operator

To test whether a belief is true, a model simply has to retrieve such a chunk, corresponding to the situation described in the belief, from declarative memory, and verify whether the future situation the retrieved belief describes corresponds to the belief as prescribed by the strategy formula. Let's take an example of a formula with the belief operator incorporated - $\mathbb{B}_{g1}^{(n2,p)}\langle d\rangle f$ and thus the player choosing to play right. It states that in Game 1 (see Figure 2 on page 7), at node n2, which is player P's first decision node, after a d move, player C will play f. Because this belief involves a player C move, the model retrieves a strategy chunk for player C's strategies. Because the belief describes a situation where player C has already played b, which is right, the model retrieves a strategy chunk where the first move of player C is right. It can then compare player C's second move in the chunk it retrieves to the move prescribed by the strategy, which is f, which is also right. If these are the same, the proposition is true. If it is not, the proposition is not true.

WM1, WM2, WM3 and WM4 are used for retrieving memory chunks corresponding to comparisons. We thus use six slots, WM5 to WM10, for retrieval of strategies. Thus we define the belief operator as:

define goal belief-one {

operator belief-one-start-retieval {
WM1 = one

RT1 = nil
WM8 = c
==>
op − > RT7
strat − > RT6
}

operator belief-true{
RT9 = right
RT6 = strat
==>
play − > AC1
right − > A2
}

operator belief-true{
RT9 <> right
RT6 = strat
==>
play − > AC1
down − > A2
}

}

5.5.2 Negation Operator

Negations can be implemented by just switching the next goal or action in the operators of a particular goal. Note that this can only be implemented in games with binary choices only.

References

[1] R. V. Sujata Ghosh, Ben Meijering, "Strategic reasoning: Building cognitive models from logical formulas," April 2014.

[2] R. V. Sujata Ghosh, "Studying strategies and types of players: experiments, logics and cognitive models," April 2017.

[3] H. d. W. A. H. Sujata Ghosh, Rineke Verbrugge, "Cross-cultural differences in playing centipede-like games with surprising opponents."

[4] R. M. Trimpop, "The psychology of risk taking behavior," 1994.

[5] J. Top, "Bridging the gap between logic and cognition: a translation method for centipede games," September 2016.