2019

Rationality in Bargaining by Finite Automata

Jim Bell
bell8410@mylaurier.ca

Rationality in Bargaining by Finite Automata

BY

James Bell

Bachelor of Applied Sciences
in Mechanical Engineering
University of British Columbia, 2018

THESIS

Submitted to the Department of Mathematics
Faculty of Science
in partial fulfillment of the requirements for the
degree of
Master of Science in Mathematics
Wilfrid Laurier University
2019

Abstract

Aspects of behavioral decision-making can be integrated into game-theoretic models of two-player bargaining using finite automata which can represent bargaining strategies in combination with various behavioral traits. The automata are used as bargaining agents who must jointly agree upon a fixed allocation of transferable utility in an infinite-horizon Rubinstein bargaining game. At each turn, the automata are given the opportunity to accept a proposed portion of the transferable utility, or to reject the proposal and make a counter-offer of their own. A round-robin tournament and ecological simulations were run to explore strategic dominance under different conditions. Principles of bargaining strategy were discussed and future fields of research explored.

Acknowledgements

First, I would like to express my gratitude towards Dr. Marc Kilgour for his guidance and support throughout this project. His encouragement fueled my passion to explore and experiment with new ideas, giving me a much broader perspective of the field.

I would also like to thank Dr. Maria Gallego and Dr. Ross Cressman for their feedback and direction through the project for helping me stay succinct and on-path. Your patience and advice was much appreciated.

I would also like to acknowledge the Wilfrid Laurier Mathematics Department for creating a welcoming and supportive culture, encouraging to the discovery and growth of new ideas.

Finally, I would like to thank the innumerable peers and mentors that I have encountered along my journey who have motivated me to be a better person every day. Although I do not have the space to thank you all individually here, please know that your impact on me and my work has been substantial and integral to my growth and well-being as a person.

Contents

List of Figures

Chapter 1

Introduction

This section will introduce the motivation of the project and the basic vocabulary and concepts to be employed throughout this thesis.

1.1 Motivation

When seeking to solve a problem through a game-theoretic lens, it is important to establish which game can be used to accurately model the problem, and also to understand how the analysis of this game translates to the problem. Numerous classic games exist, some of which will be considered in this text, whose structures often can be adequately mapped to real-world situations. Although the analysis of these games may provide insight to real-world decision makers, each game will almost always have some kind of blind-spot or difficulty in mapping its analysis from model-world to the real world. As such, the development of novel games and analytical methods is critical for the continued expansion and evolution of the field of game theory.

Game theory and similar fields of analysis have typically focused on the normative or descriptive modes of modelling, concerned with modelling "optimal" decisions, or decisions that are actually made in the real world. Consequently, less emphasis has been placed on the study of *how* decisions are made, which is particularly important when facing decision problems with imperfect information and other such uncertainties. A common thread through this document will be the focus on procedural rationality in building a bottom-up model which sheds light on the poorly understood dynamics of complex and uncertain decision problems.

To this end, this project seeks to introduce a novel analysis of the Rubinstein Bargaining Game [22] in the hope of developing a new understanding of the dynamics and strategies surrounding bargaining-type interactions. Many different events can fit into this class of interaction, whether in a direct sense or in a more general sense where the general principles of the interaction (or closely related decisions) allow for a carryover of intuition and understanding. For example, haggling over the price of a single resource closely mirrors the mechanics of

the simple bargaining model produced by Rubinstein [23], whereas negotiations over a complex array of items may follow a similar enough pattern of initial stances followed by patterns of concessions that the principles of bargaining can be applied effectively.

The end goal of this project will be to improve intuition surrounding strategy selection in real-world bargaining-type encounters and to better understand the big picture effects of different approaches to strategy selection.

1.2 Basic Concepts in Games

The field of Game Theory is primarily concerned with the detailed analysis of games so as to understand the optimal decision-making of its players. Games can range from simple to complex, from competitive to cooperative, stochastic to deterministic, and in many other ways. In each game, every Decision Maker (DM) will be faced with at least two choices of action. Analysis of the game will revolve around what choices a DM is expected to make, and the resulting impact of his decision(s). Each major class of game will provide unique insight into rational choice given a particular set of circumstances. The following section will discuss some fundamental concepts in the analysis of games, the structure of certain classic games, and the insight to be drawn from their analysis.

1.2.1 Preferences and Utility

For a DM to make coherent and rational decisions, a clear goal or objective must first be established. Furthermore, to properly evaluate the eventual decisions that they will encounter, a DM must have a thorough understanding of how each of these decisions will affect their progress towards their goal. The concepts of Preference and Utility help to provide a framework for understanding how a DM will approach decision problems and how they evaluate its outcomes.

A DM's preference over the possible outcomes of a decision problem sheds light on their framework for evaluation. To determine a DM's preference ordering, they must first be able to identify the relationship between each of the possible outcomes of a decision.

Suppose that the set of all possible outcomes of a decision problem is W. For every $x, y \in W$, a DM must decide whether or not they find x to be at least as desirable as y. Symbolically, they must decide whether or not $x \succeq y$. Should a DM decide that $x \succeq y$ and $y \succeq x$, then he is said to be indifferent between outcomes x and y. Should he decide that $x \succeq y$, but $y \not\succeq x$, then he is said to have a strict preference of x over y, or $x \succ y$.

In this document, all preferences will be assumed to be *transitive*, meaning that if $x, y, z \in W$, and $x \succeq y$ and $y \succeq z$, then $x \succeq z$ must be true. The benefit of this assumption is that it eliminates circular preferences which can sometimes obfuscate a decision problem. Circular preferences are occasionally seen in real-life situations and are not necessarily wrong, but this topic will not be covered here.

Now that a framework has been established for comparing alternatives, we must now explore the case when a decision problem has risky outcomes. Proposed by John von Neumann and Oskar Morgenstern in 1947, von Neumann-Morgenstern utility is a value function through which a DM can evaluate their preference over risky alternatives. This utility function provides an understanding of a DM's personal preferences by unifying the preference for all possible outcomes under one function, whether they involve different units of measurement (i.e. apples or dollars), or different levels of uncertainty of the expected payoff (i.e. \$0.50 for sure or a coin flip that gives you \$1.00 if it lands heads or \$0 if it lands tails). It can be said that a DM makes decisions *as if* they are maximizing their personal utility function. The utility function becomes defined

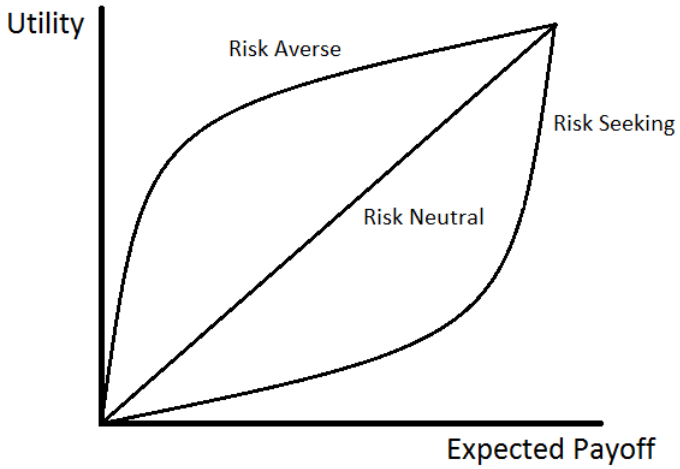

Figure 1.1: *Attitudes Toward Risk*

through the revelation of a DM's preferences, it can not be defined *a priori*.

A principal benefit of the von Neumann-Morgenstern utility function is that it illustrates a DM's *risk preference*, that is, whether or not they prefer a large expected value of a risky decision to a lower but guaranteed outcome. Figure 1.1 illustrates the basic categories of risk attitude. A DM who makes decisions based purely on the maximization of the expected value of the payoff is said to be Risk Neutral. Furthermore, a DM who prefers to accept a guaranteed payoff less than the expected value of a lottery is said to be Risk Averse, while a DM who prefers the lottery is said to be Risk Seeking. The conception of the different attitudes towards risk in the real world will be discussed in Chapter 2.

1.2.2 The Nash Equilibrium

Nash Equilibrium is among the most fundamental tools in game analysis as it is both versatile and powerful in providing insight into the how a game will be played. The formal definition of a Nash Equilibrium is as follows.

Definition 1.1. In an n-person strategic-form game $G =< N, S_1, S_2, ..., S_n; u_1, u_2, ...u_n >$,

a strategy profile $s^* \in S_1 \times S_2 \times ... \times S_n$ is a Nash Equilibrium if and only if $u_i(s^*) \geq u_i(s^*|s_i)$ for all $i \in N$ and $s_i \in S_i$, where N is the number of players in the game, S_i is player i's strategy set, any $s_i \in S_i$ strategy for player i, and u_i is the utility corresponding to strategy set i.

Essentially,this means that every player selecting a strategy will anticipate his opponents' strategies and that he will select his optimal strategy given that condition. If any of the players in the game does not select a strategy that maximizes their outcome, the game is not at a Nash Equilibrium. To make an optimal strategy selection, each player must be sufficiently knowledgeable to correctly anticipate their opponents' moves and thereby select their corresponding optimal strategy.

Nash Equilibria can exist in pure forms, where strategy selection is deterministic for all players, or mixed forms, where strategy selection can involve assigned probabilities to multiple strategies. All finite strategic form games have at least one Nash Equilibrium, making it an accessible analytic tool across a wide range of games. This method of analyzing games depends heavily on the fact that each player is able to adequately compute their best response to their opponents' strategies and will then employ that best response strategy. If each player in a game is employing a best response strategy to their opponent's strategy, they are all effectively employing a best response to each others best responses and can therefore not improve upon their outcomes. Since rational play will not deviate from these best responses, the play of the players is said to be at an equilibrium. The Nash Equilibrium is therefore most useful for analyzing simple strategic form games.

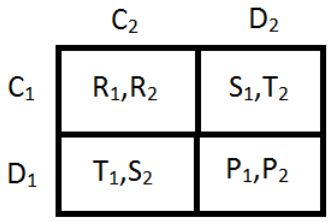

Figure 1.2: *Prisoner's Dilemma Payoffs*: In each pair, the row player will receive the payoff on the left while the column player receives the payoff on the right.

1.2.3 The Prisoner's Dilemma

The Prisoner's Dilemma game is one of the most famous games in the field of game theory. It has been analyzed extensively in both its strategic-form, where the two players make one decision each and an outcome is reached, and in its iterated form, where the two players are partnered for an extended series of strategic-form games and the outcomes are assessed based on the cumulative payoffs of each of the players. Both forms of the game will be briefly discussed here to provide a basis for future analysis.

Figure 1.2 shows the general form of the Prisoner's Dilemma game, where each player has the choice to either cooperate, C, or defect, D. The payoffs are described as: a Reward for mutual cooperation, R, the Temptation, T, and the Sucker's payoff, S, if one player defects while the other cooperates, or Punishment, P, should both players defect. The payoffs must satisfy

$$\mathrm{S}_1 < P_1 < R_1 < T_1$$
$$\mathrm{S}_2 < P_2 < R_2 < T_2$$

The choice to defect strictly dominates the choice to cooperate, as no matter how the opponent plays, the outcome from defecting will always be greater than the outcome of cooperation. This leaves both players with the *Punishment* outcome at the only Nash Equilibrium of this game. Clearly, both players would prefer

the *Reward* payoff, yet rational play will not allow it.

The one-off Prisoner's Dilemma game can be used as a useful tool for modelling simple interactions between two separate parties faced with a choice of whether or not to cooperate. Examples of such interactions include countries seeking to establish policy to reduce emissions, or deciding whether or not to own a firearm. The analysis of this game has helped shed light on the seemingly paradoxical behaviour of parties who would stand to benefit from cooperation, but fail to do so.

Although the simple nature of this game allows for its insights to be readily applied to a wide range of scenarios, it does neglect possible key elements in real-world situations. For example, this one-off style of game does not entertain the possibility of future interactions between the players, nor does it account for the players' personal interpretation of the situation. Fortunately, the analysis of the Iterated Prisoner's Dilemma helps to shed some light on these areas of uncertainty.

The Iterated Prisoner's Dilemma has been particularly useful in the understanding of the development of behaviour patterns within groups on individuals who may interact on a regular basis. This analysis has helped us to understand seemingly irrational behaviour patterns observed in the natural world by highlighting the benefit of cooperation in long-term partnerships. These ideas will be explored in greater depth in Chapter 2.

1.2.4 The Hawk-Dove Game

The Hawk-Dove game, although similar in nature to the Prisoner's Dilemma game, offers insight into rational decision-making when two-players enter into direct conflict over a resource of value, V. Analysis of this game was at the forefront of the development of the field of Evolutionary Game Theory, and its

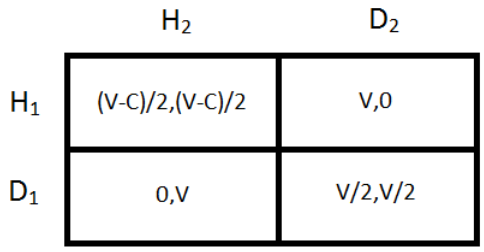

Figure 1.3: *Hawk-Dove Game Payoffs*: This is the simplest form of the Hawk-Dove Game.

many forms have been widely used to interpret animal behaviour [26].

Figure 1.3 shows the breakdown of the game, where each player chooses to play either the Hawk Strategy, and make an aggressive play on the resource, or the Dove strategy, and make a passive play on the resource. If both players play Hawk, the cost of injuries from the ensuing conflict, C, will be deducted from their final payoff. It is assumed that the value of the resource is greater than the cost of injury. If one player plays Hawk while the other plays Dove, the aggressive Hawk player will obtain the full value of the resource, V, while the Dove player is left empty-handed. Should both players play Dove, they will be able to share the resource equally. If the payoffs satisfy

$$0 < C < V$$

then there is no pure Nash Equilibrium in the Hawk-Dove game, but there will be a mixed-strategy Nash Equilibrium that is dependent on the Values of V and C. The mixed strategy Nash Equilibrium sees both players employing both strategies at a fixed rate to optimize their payoff. If the payoffs satisfy

$$0 < V < C$$

then a pure Nash Equilibrium exists where both players play Hawk and thus receive a negative payoff.

Although it is represented as a strategic-form game, where both players act simultaneously, the Hawk-Dove game often models situations where each player has the opportunity to infer their opponent's strategy. For example, a bird puffing his chest when confronted by another bird could be a signal that he intends to be aggressive, while a bird who hunches in a similar situation may be signalling that she intends to be passive. In this light, the crucial takeaway from the analysis of the Hawk-Dove game would be that it is best to be passive against aggressive opponents to avoid major loss, and best to be aggressive against passive opponents to maximize gain.

1.2.5 Extensive-Form Games

An extensive-form game is a game in which a player will make decisions based on the occurrence of a number of events in a sequence. Although it shares a similar interaction in decisions between the players as seen in such strategic-form games as the Prisoner's Dilemma or the Hawk-Dove game, extensive-form games allow for the possibility of players to gain information, for example about an opponent through the decisions that the opponent makes through the course of the game. This information-gathering process can be used to inform future decisions, thus becoming a critical component of the analysis of this type of game.

A distinguishing feature of extensive-form games is the *game tree*, a sequence of decision nodes, chance nodes, and terminal nodes that all stem from a single root, or origin. Decision nodes are the opportunities at which a specific player must make a decision which will impact the future of the game. Stemming from each Decision Node are the possible choices by that player at that particular time. Stemming from the Chance Nodes are the probabilities associated with landing at the next node in the branch. Terminal Nodes are found at the

end of each unique path in the game tree, and are associated with outcomes corresponding to the play through that path.

Extensive-form games of perfect information are typically solved using *backward induction*, where analysis begins at a terminal node and the game is traced backwards through the game tree to the origin by determining what decision each player would make at each node, given the terminal outcome that would result from that path. The underlying assumption in this method is that the players will behave rationally and make decisions corresponding to their optimal outcome. A *sub-game perfect equilibrium*, or *SPE*, is a set of decisions through the entire game tree corresponding to rational decisions made by each of the players. This method of analysis may provide multiple SPEs, corresponding to multiple ways that the game would logically play out, though the outcomes of the SPEs may differ, with players having different preferences between them.

Players are said to have *perfect information* if each player is fully aware of their opponent's decisions and other game events prior to each decision node of the player. The assumption of perfect information within a game significantly restricts a rational player's options, and therefore the possible outcomes of the game. Those seeking to model situations as extensive-form games must be careful to understand the implications of the underlying assumptions and how they may affect the results from the model.

1.3 Normative vs Descriptive Modelling

Decision problems can be modelled in a variety of different ways, usually depending on the specific goal of the modelling endeavour. Two of the most common types of models are normative models and descriptive models. Normative models typically deal with considering what the optimal decision would be in a given scenario, while descriptive models seek to describe which decisions are actually

made by people facing a certain type of decision problem. Both methods yield insight into the decision-making process and can be used to help guide future decisions.

Normative modelling requires the usage of a perfectly rational agent, one who has a complete and continuous set of preferences and is capable of computing an optimal path of action given the available information. In some literature, this rational agent is referred to as *homo economicus*, the economic man, and a cousin of *Homo sapien* who is solely concerned with maximizing their payoff. This representation of human decision-making agents has been criticized for its ignorance of aspects of human nature and outside social influences, but can nonetheless prove useful when analyzing decision problems.

Descriptive modelling revolves around the observation of what decisions are made in the real world and seeks to build a model to encapsulate the understanding of human behaviour. The benefit of this style of modelling is that it can point to underlying truths about certain factors which affect decision-making, thereby creating a foundation for more accurate models in the future.

Prescriptive modelling lies in between normative and descriptive modelling, prescribing which decisions *should* be made given that decision-makers may not be completely rational agents and can be vulnerable to unstructured outside influences. This project will take on a more prescriptive style, with the end goal being to develop a stronger intuition around real-world bargaining decisions.

1.4 Evolutionary Stability

The concept of evolutionary stability relates to the study of populations and their dynamics. More specifically, it refers to the *fitness* of the constituents of the population and the potential fitness of would-be invaders to the system. In this context, fitness refers to an individual's ability to reproduce. A high

level of fitness signifies that an individual is more likely to pass on behaviour patterns and biological traits to future generations of the population than an individual with a lower fitness level. Within the context of game theory, fitness can be simply understood as the amount of utility acquired through the play of a game by a particular behaviour pattern or species. Given a specific playing environment, each strategy will be associated with a particular expected utility, and therefore expected fitness within this game.

For a population to remain at a certain equilibrium [1], its constituents must employ a strategy that is superior to the other strategies that may be employed in that environment. If there is a group of individuals employing a strategy that is superior to those currently in use in an environment, this group would destabilize the native population from its current equilibrium. The following theorem explains the conditions necessary for invasion by a new strategy.

Theorem 1. *Suppose that a population consists of players playing either strategy A or strategy B, with a fraction α playing A and the fraction $1-\alpha$ playing B. After a series of interactions between members of the two populations, the resulting fitness of the two populations can be represented as follows:*

$$D_A = \alpha E(A,A) + (1-\alpha)E(A,B)$$
$$D_B = \alpha E(B,A) + (1-\alpha)E(B,B)$$

where E(i,j) is the expected payoff to a player using strategy i in an encounter with a player employing strategy j, and D_i represents the fitness level of population i.

If $D_B > D_A$, then a sufficiently large bridgehead of players using B will displace an established population of those using A. For this to be true, either of the following conditions must be satisfied:

[1] If a population is at an equilibrium, there is no change in the relative proportions of the sub-populations (different species) within the population.

$$E(B, A) > E(A, A) \text{ or}$$
$$E(B, A) = E(A, A) \text{ and } E(B, B) > E(A, B)$$

If a population is able to withstand any invasion of mutants with different strategies, it is said to be evolutionarily stable. Although the definition of evolutionary stability does not invoke the notion of rational decision, it does imply a certain level of local optimization of behaviour pattern. At the very least, evolutionary stability provides an intuitive sense of how populations will select strategy over time. The idea of an evolutionarily stable strategy can be thought of as a parallel to the idea of a Nash equilibrium when the dimensions of time and adaptation are considered in a game. Evolutionary stability arises when an equilibrium is reached after every player in a population continues to play their best possible strategy. This concept will be further explored in Chapter 4.

1.5 Rationality

The traditional understanding of rationality involves a decision-maker to whom is assigned stable, complete, and consistent preferences over a range of alternatives of which they have complete knowledge. Within this framework, a decision-maker's actions can be accurately predicted and understood in virtually any setting. This definition of rationality enables a rigorous axiomatic approach to solving games, but allows no room for an unstable environment, changing perspectives or needs, or confrontation with irrational agents.

The concept of *posterior rationality* explains intentions via backward induction of an individual's observed actions. In this sense, a decision-maker's actions precede the formal definition of their goals [17]. This notion complies with Richard Dawkins' [11] observation of the dominance of memes in both biological and social contexts. According to Dawkins, decisions and actions are

formulated through the applications of heuristics, or rules-of-thumb, as opposed to the traditional conceptualization of purely centralized command.

Although these two concepts do not share a common approach, they can both be equally useful in certain contexts. For example, traditional rationality is quite effective at analyzing simple strategic-form games where both players have complete knowledge of the possible outcomes and of each others preferences. Conversely, in more complex games where the outcomes and preferences are unknown to the players, modes of analysis such as evolutionary stability, predicated on the use of posterior perspective on rationality, can assist in making useful predictions about how the game will play out.

Throughout this document, games will be analyzed without the assumption of strict rationality of the players, either due to the lack of information or due to the overwhelming computational cost of strategic analysis in complex situations. The impact of this assumption will be discussed along the way, with the intent of correctly assessing real-world behaviour and accurately prescribing decision making strategy.

Chapter 2

Literature Review

This section will provide a foundation of previously published literature from which this project will be built. The main concepts to be discussed herein include the fundamental analysis of bargaining in Game Theory, observed bargaining behaviour patterns within the domain of behavioural psychology, and the value and applications of agent-based modelling approaches.

2.1 The Rubinstein Model

One of the most influential contributions to bargaining theory was made by Ariel Rubinstein [23] in his paper "Perfect Equilibrium in a Bargaining Model". In this paper, Rubinstein set out to solve the "bargaining problem", that he defined as:

> Two individuals have before them several possible contractual agreements. Both have interests in reaching agreement but their interests are not entirely identical. What "will be" the agreed contract, assuming both parties behave rationally?

To answer this question, Rubinstein proposed an alternating-offers model where two players would make offers to each other with regard to the division of a pie of size 1. Upon receiving an offer, a player would face the choice of either accepting their opponent's proposed division of the pie, or refusing their offer and returning a counter-offer. The key parameter to this model was the fact that both players would incur a cost for any delay in reaching agreement. The cost of delay is referred to as δ, a discounting factor applied to each player's outcomes after each round and thus creating the effect of a shrinking pie. Although the structure of the model allows for infinitely many offers from either player, the effect of time will force rational players to reach agreement immediately.

The Rubinstein model for bargaining offers a few attractive features for analysis. First of all, the simple structure which allows players to make alternating offers maps relatively well to real-world negotiation scenarios, allowing for the development of a clear and concise intuition around these problems. Another feature of note is that this model allows for analysis from a strategic approach, rather than from an axiomatic approach which may produce some seemingly "stylized and artificial results" [23]. By modelling the bargaining process as a strategic-form game, a deeper analysis can be performed by analyzing the indi-

vidual decisions of the players. Finally, the model offers as a key insight that an individual's bargaining power rests in their patience. If a player faces strict penalties for delaying an agreement, they will be forced to accept a smaller portion of the pie than their more patient counterpart. In this case, a more patient bargainer is a player who has a higher δ value than their opponent. Altogether, these features of the Rubinstein model have made it an excellent platform for bargaining analysis.

Now that the structure of the game has been defined, it is possible to explore how the game will be played. In this model, one player (Player 1) is chosen to make the first offer while the other player (player 2) gets to decide whether to accept Player 1's division of the pie. Players can make offers at times in the set $T = 0, 1, 2, ...$ Any offer is of the form (x_1, x_2) where x_i signifies the portion of the pie to be received by Player i. These offers must satisfy

$$X=\{(x_1, x_2) \in \mathbb{R} : x_1 + x_2 = 1 \text{ and } x_i \geqslant 0 \text{ for } i=1,2\}$$

The basic framework of the game is predicated on the following assumptions, where $x, y \in X$ are allocations of the pie to be received by the player and $t, s \in T$ are discrete points of time when a bargaining agreement can be made :

"(A-1) Pie is desirable (for any $t \in T$ and $x, y \in X$, $(x, t) \succ (y, t)$ if and only if $x > y$)
(A-2) Time is valuable (for any $t, s \in T$ and $x \in X$, $(x, t) \succeq (x, s)$ if $t < s$, and with strict preference if $x > 0$)
(A-3) Continuity (Let $\{(x_n, t_n)\}_{n=1}^{\infty}$ and $\{(y_n, s_n)\}_{n=1}^{\infty}$ be convergent sequences of members of $X \times T_{\infty}$ with limits (x, t) and (y, s), respectively. Then $(x, t) \succeq_i (y, s)$ whenever $(x_n, t_n) \succeq_i (y_n, s_n)$ for all n)
(A-4) Stationarity (the preference of (x, t) over $(y, t + 1)$ is independent of t)
(A-5) The larger the portion, the more compensation a player needs for a delay of one period to be immaterial to him" [20]

A point of note here is that the preferences of the players over the partitioning of the pie are diametrically opposed as any increase in pie for one player will result in a proportional decrease for the other player. Although the players are selfishly motivated to maximize the amount of pie they receive, they are indirectly motivated to cooperate to ensure that the time-discounting effect does not significantly erode their share of the pie .

As such, the strategies employed in this game involve actions to be made at every possible node in the game. That is, a player will know what their next offer would be should their opponent reject their current offer. Using this knowledge, a player will always accept an offer that is greater to or equal than their next offer discounted by one time period, which is equivalent to the following, where x_i^t is the received offer of the pie at time period t, x_i^{t+1} is the offer that the player would make in the next time period, and δ is the discount factor.

$$x_i^t \succeq x_i^{t+1} \times \delta$$

If the current offer does not satisfy this inequality, the offer will be rejected and the player will counter-offer with x_i^{t+1}.

2.2 Empirical Results in Bargaining

To optimize the procedural rationality involved in bargaining, a basis must first be established which explains empirically observed behaviour patterns in this domain. This section will discuss some of the documented components of human decision-making involved in bargaining games, how these components interact, and the consequent experimental outcomes when the components are analyzed. Although the procedures of decision-making here are not likely to be optimal, it is important to first establish a descriptive model of human bargaining behaviour before proposing a normative model.

In common bargaining games in the real world, players have limited information regarding their opponent's preferences and preferred strategies. In fact, it is often advantageous for players to minimize the information available to their opponent in an effort to maximize their bargaining power. Consequently, players must possess mechanisms through which to make quick and effective decisions given imperfect information. These mechanisms, or heuristics, are shared by many players and thus lead to similar but not parallel behaviour patterns. Heuristics can be used to solve such problems as strategy inference, best response approximation, and preference ordering. The following sections explore some of these common heuristics and discuss their value in bargaining games.

2.2.1 Perception of Firmness and Toughness

Of principal importance in bargaining is the ability to discern an opponent's so-called bargaining-power. In the Rubinstein model, a player's bargaining power is completely described by their discount for time, where the player who can afford to be more patient has the upper hand in the game. In real-world bargaining scenarios, additional confounding factors influence a player's bargaining power, some of which may be unknown or unknowable. Therefore, players must have some form of procedure for inferring their opponent's bargaining power so as to correctly formulate their strategy. To do so, players may rely on a perception of opponent "toughness" as a proxy to their bargaining power.

Perceptions of an opponent's bargaining position can be generated through a number of different means, including reputation, initial stance, and concession behaviour [5, 27, 28]. It has been shown that bargainers with reputations for being tough can find themselves in situations with a self-fulfilling prophecy, where negotiations may break down due to perceived unfairness, or result in a

poor outcome due to excessive bargaining friction [28]. Conversely, a pattern of "tough" tactics may incite an opponent to concede rapidly, resulting in a better bargaining outcome [5, 27]. The resulting intuition is that a bargainer's strategy is dictated by their confidence, which is in turn a product of how they mentally frame the situation [19]. Therefore, to better predict bargaining outcomes, an effort must be made to understand a bargainer's perception of their situation.

Clearly, the perception of an opponent's bargaining power will have an effect on strategy selection and the eventual outcome of the bargaining game. Any attempt to model bargaining where there is incomplete information about the players' preferences and strategies must therefore incorporate some form of inference mechanism through which a player can gather information about their opponent and adjust their behaviour accordingly.

2.2.2 Strategy Adjustment

Since the information available to a player may change throughout a bargaining game, it is fair to assume that their strategy will adapt accordingly. These adaptations seem to be in part driven by a social norm for reciprocity, where a player will be more willing to make larger concessions if they get the sense that their opponent is playing "nice" with them [5]. Should both players adopt this behaviour pattern, a positive-feedback cycle will ensue with both players making large concessions until agreement is reached. However, "rewarding" an opponent for concessionary behaviour can have unintended effects which lead to worse outcomes. Research has shown that bargainers on the receiving end of concessions may receive a confidence boost and therefore adopt less friendly tactics to take advantage of their opponent's accommodating behaviour [16, 27]. This represents a dichotomy similar to that seen in the Prisoner's Dilemma game, where the incentive for cooperation is counter-balanced by an aversion to

being taken advantage of.

To contend with this behaviour, bargainers may adapt different concession strategies in an attempt to elicit greater concessions from their opponent. In addition, pattern recognition would become an important aspect of their behaviour to ensure a best-response to the opponent's own concession patterns. In this case, concession patterns could be seen as an attempt to communicate information between bargainers, but the credibility of this information would remain uncertain. Strategy selection should therefore entail some form of decision mechanism which interprets an opponent's intentions through their actions.

2.2.3 A Changing Utility Function

A typical assumption in the usual normative analysis of bargaining is that each player is equipped with a defined and unchanging utility function through which to evaluate the potential outcomes in the game. Although this assumption allows for a more rigorous and complete analysis, it does not carry over to the real world, where players are subject to alter their aims through the course of a game. To more accurately model a player's decision-making, a closer look must be made at the driving force of their outcome-evaluation process.

The common framework in the field of psychology which most closely resembles the utility function optimization framework is the *level of aspiration*. A player's level of aspiration can be understood as the measure of their perceived entitlement to the outcomes in the game. From this perspective, the player's aspiration level will therefore guide their decisions throughout a game, similar to how a game theorist would view a player to reference their utility function to guide their play [24].

The key difference between level of aspiration and utility theory is that level of aspiration can often be a fluid concept, subject to change with new infor-

mation or an altered perspective. Level of aspiration will dictate initial offers in a bargaining context, but is subject to change throughout the game in response to an opponent's behaviour [16]. Factors such as bargainer confidence and opponent concession patterning can have an effect on a player's level of aspiration, with decreased self-confidence in bargaining and perceivably tough concessions from the opponent leading to a decrease in entitled or desired outcome. Seemingly, as more information becomes available to the players, they update their beliefs about their value of the outcomes to accommodate their new-found perspective.

This updating process has a distinct effect on decision-making, as a player is essentially approaching each decision within a game with a potentially different utility function. This method of decision-making can produce results quite different than those obtained through an analysis involving utility theory analysis and could be responsible for seemingly irrational behaviour in real-world bargaining.

2.3 Agent-Based Modelling

Agent-based modelling is a method of system analysis based around the interactions of autonomous agents at the most basic level of the system. In essence, it is a bottom-up way of looking at a system, enabling observers to understand how seemingly minute dynamics at its lowest level will affect the entire system. Often, simple mechanics at the base level of a system will lead to complex dynamics at its highest level, making this bottom-up approach a necessary consideration in many modelling undertakings. Agent-based modelling is often found in the study of ecology and is becoming popular in the fields of economics and political science.

An agent-based model begins with the elaboration of a set of rules and

assumptions that will govern the limits of the behaviour of the agents within the model. These rules are typically concerned with the mechanics of how agents interact, and basic structures of the model, such as how to dictate the passage of time or the accumulation of resources. Subsequently, agents will be programmed with a fixed set of instructions of how they must interact within the model. These instructions can be different for any of the agents within the model and they can range in complexity depending on the goals of the model. Once the structure and behaviours of the agents have been established, their interactions can be simulated, thus inducing a range of dynamics at all levels of the system.

By endowing independent agents with a set of instructions which completely govern their interactions, it is possible to correlate the macro-dynamics of a system directly to the mechanics of the individual agents. This approach has been used in a wide range of fields, from infectious disease and genetic modelling, to macroeconomic and political action modelling [2]. In addition, the inherent structure of agent-based models is compatible with machine learning techniques that can deepen the level of analysis and expand the domain of study, as in the work of Herbert Simon [25]. By focusing on how certain dynamics arise in a system rather than what those dynamics are, a more complete perspective can be created, allowing for a more intuitive understanding of the system and its future behaviour. Furthermore, this modelling approach allows analysts to substitute the assumption of optimal equilibrium behaviour with the bounded rationality[1] of the agents. As a consequence, the departure of an agent's behaviour from normative models can be explained by an emergent procedural rationality from players in a game.

This style of modelling is particularly useful for analyzing high-order patterns

[1]Bounded rationality is the idea that an agent is limited in its decision-making with respect to cognitive processing abilities, available information, and/or time with which to make the decision

in complex systems. By simulating the interactions of individually motivated agents, it is possible to observe the emergence of more complex dynamics and better understand their cause-and-effect relationship. Furthermore, agent-based modelling allows for space to test the sensitivity of model parameters and the robustness of observed patterns or equilibria. By building models from the ground up, researchers can gain deeper insight into complex phenomena by better understanding the underlying driving factors.

2.4 The Axelrod Tournament

In 1980, Robert Axelrod [3] ran a tournament for the iterated Prisoner's Dilemma game in which the entrants were computer programs submitted by some of the top minds in the world in the fields of mathematics, psychology, and political science. These computer programs were pre-set decision rules for the playing of an iterated Prisoner's Dilemma game. In his tournament, Axelrod had every one of the programs play each other in a round-robin style tournament [2] to determine which decision rule represented the best strategy for playing the iterated game. The results of each match-up were added up to determine a strategy's total score. The tournament broke new ground in not just its approach, but also in its implications about the natural trend towards cooperation in natural systems, quickly becoming a cornerstone work in the field of game theory and social sciences.

2.4.1 The Tournament Approach

In an attempt to determine the best strategy for the iterated Prisoner's Dilemma game, Axelrod solicited strategies from a range of experts in different disciplines

[2] A round-robin tournament is a contest in which every player plays each of the other players in a game.

for his tournament. Due to the complex nature of the iterated game, a clean analytic solution was not viable, making the crowd-sourcing effort for effective strategies a worthwhile endeavour. Axelrod hoped that by casting a wide net that would capture a diverse group of strategies, selective pressures would produce a clear winner and therefore dominant strategy in the iterated Prisoner's Dilemma game.

This tournament approach for solving the iterated Prisoner's Dilemma game can be understood as an agent-based model, where agents of bounded rationality and behaviour interact in a pre-determined fashion. The structure of the tournament had each decision rule play the iterated Prisoner's Dilemma game against each other decision rule, including itself, one time. After all of the games had been played, each of the entrant's total scores were averaged over their number of games played and they were ranked according to their performance. This procedure benefited the analysis of the robustness of the performance of each of the decision rules as their performance was weighted equally against each of its opponents. Through analysis of the average scores and of the outcomes of individual matchups, conclusions were drawn about the validity of the strategies themselves and the dynamics of their interactions.

To expand upon the results of the round-robin tournament, the same decision rules were then entered into an ecological model [3] to further evaluate their performance. Although an ecological model would typically be used to model the spread of genetic patterns through a population, this approach can be extended to include strategy selection as well, since it is reasonable to assume that poor-performing strategies would be rejected in favour of better performing strategies

[3]The ecological style model is an agent-based model which allows agents to interact through a game. As the games are played, the model tracks the success of the agents through their relative population changes in the model. A simple example of this kind of model is a scenario where there are chipmunks and squirrels competing for nuts in a forest. If the chipmunks are better at finding nuts, the chipmunk population will increase, while the squirrel population will decrease.

over the course of time. Consequently, this approach would quickly highlight which strategies were truly the best, as they would have to out-perform similarly competent strategies, rather than taking advantage of weaker ones.

The ecological model took an equal number of agents representing each of the strategies from the round-robin tournament and placed them in a simulated ecosystem where they would be matched up to play the iterated Prisoner's Dilemma game. The probability of being matched up with an agent of a particular strategy would be proportional to the population size of the agents representing that strategy. In each generation of this model, the expected score of each decision rule, or its "fitness", would be calculated and the population sizes of each of the decision rules would be proportional to its fitness from the previous generation. Consequently, if a decision rule is performing poorly, it will be less represented in future generations, while better performing decision rules will see greater relative representation in the population. Over time, the environment would be expected to settle towards a final equilibrium, where the best strategies are the most represented within the population.

2.4.2 Tournament Results and Implications

Multiple versions of the computer tournament were run by Axelrod, beginning with his initial round-robin tournament, and followed by expanded round-robin tournaments and ecological tournaments. Each of these tournaments consisted of the same procedure of simulating extensive iterated Prisoner's Dilemma games between a diverse group of computer programs representing strategies with which to play the game.

The most noteworthy outcome from the tournaments that Axelrod ran using the iterated Prisoner's Dilemma was the outstanding performance of the Tit-For-Tat decision rule, submitted by Anatol Rapoport[3]. Tit-For-Tat was one of

the simpler programs submitted to the computer tournament, a rule that would Cooperate in the first game, then mirror its opponent's previous move in every subsequent game. The Tit-For-Tat strategy was described as nice, provocable, fair, and clear, meaning that it would initially cooperate, it would retaliate if mistreated, it would not seek a payoff greater than its reasonable share, and it would be obvious in its intentions. These characteristics were also observed in other decision rules that performed well, but Tit-For-Tat best encapsulated and employed all of these ideas.

Overall, these tournaments were able to justify how cooperation would arise organically and rationally within natural environments, even when there could exist a temptation for selfish behaviour. Although the iterated Prisoner's Dilemma game can not capture many of the complexities of the real world due to its simple structure, the key intuitions that come from it can help guide real-world understanding and policy.

2.5 An Heuristic Approach to Rational Choice

Although much of the literature on decision theory is predicated on the assumption that people will make rational choices unless compromised by emotional or environmental factors, it has been shown that this assumption can fail in practice, leading to seemingly unexplained and irrational choices. Over the course of a few decades, Amos Tversky and Daniel Kahneman caused a massive shift in the field of decision theory by showing through experiments how the default state of human decision making fails to match with the once common notion of the rational decision maker, or *homo economicus* [15]. Rather, people typically rely on a complex and integrated system of rules-of-thumb, or heuristics, to formulate decisions in situations where rational choice would be computationally costly. This system of heuristics is ear-marked by a few key features that create

recognizable departures from the common rational decision theory, leading to decidedly divergent decision outcomes.

2.5.1 Heuristics

The process through which humans process information and reach decisions remains far from a settled science, but experimentation has led to the discovery of a number of patterns in processing and thought. When compared to other processors of information such as computers, humans possess very weak short-term memory skills, but extensive and somewhat mysterious abilities to access information stored in long-term memory. This observation is compatible with empirical evidence in psychology which suggests that often-times an individual will unconsciously avoid making difficult mental computations by substituting in a more simple and comparable problem to solve, as shown on numerous occasions by Daniel Kahneman and Amos Tversky [15, 29]. Although this substitution clearly diverges from the practices of *homo economicus*, it must be contended with to establish a firm foundation of the decision-making process.

Employing heuristics in decision-making involves breaking down a complex problem into a series of much simpler problems that can be easily solved with readily available information. Although this method decreases computation time, it may disregard critical aspects of the original problem, leading to potential gross oversights or missteps in decision-making. Fortunately, through selective pressures over the course of time, these heuristics have been refined so as to produce favourable outcomes in most settings. Therefore, by acknowledging that a substitution has been made from the original problem, decision problems can be better understood by analyzing the heuristics that make up the driving force of the real computation being performed.

2.5.2 Biases

It has been shown that an heuristical approach to problem solving can produce favourable results, but it is of great importance to understand its possible shortcomings. When a complex problem is substituted for a simpler one, the decrease in resolution can create blind-spots in the decision-making process. These blind-spots are inherent in the structure of the heuristic framework and, over time, will create chronic biases in decision-making. When viewed through the lens of rational choice, these blind-spots may appear to be inexplicable and irrational behaviour, while in actual fact they are simply categorizable side-effects of the working system of the human mind. Proper understanding and awareness of biases is critical for the employment of this decision-making framework.

Much literature has been directed towards recognizing and understanding the effects of a growing list of cognitive biases, and most of those will not be discussed here. Although many of these biases have profound effects in the real world, a few are particularly noteworthy in the field of decision-making, and in particular, bargaining.

Of particular interest in the area of evaluating preferences and utility are the biases of the *framing effect* and the *anchoring effect*. Framing relates to how the same set of information may be interpreted differently given different accompanying contextual information [15, 19]. For example, the derivation of a utility function may be affected by a perceived adversarial relationship with a bargaining opponent, thus leading to decisions which diverge from the normally prescribed rational choice theory. Similarly, the anchoring effect describes how an individual's preferences may be involuntarily altered by contextual information that may not be directly relevant to the decision problem which they face. An obvious example of this can often be seen in marketplace haggling, where the seller establishes an outrageously high initial price for the good that they are

selling. Although the buyer may not be fully aware of it, this outrageous price demand can impact their evaluation of the value of the seller's good, leading to different haggling scenarios than would have performed had they not been exposed to the seller's initial offer. This initial offer can act as an anchor for future offers, even though it was clearly never a viable outcome in the bargaining effort. The anchoring effect plays well into the critique of Nash's [18] fourth axiom in his bargaining solution, the independence of irrelevant alternatives. Although in his theory a non-viable offer should have no impact on the final outcome, many have criticized Nash's approach for being overly artificial and failing to translate to real-world scenarios. For a theory of bargaining to accurately map to real-world behaviour, it must account for the potential biases involved in the formulation of preferences and value functions.

When analyzing the cooperative aspect of bargaining, it is important to consider the possibility of noise created by the *endowment effect* or the *zero-sum effect*. Respectively, they are the tendency of individuals to over-value property that they feel entitled to, and the tendency to frame interactions as zero-sum endeavours. Bargaining is inherently a positive-sum enterprise where neither party can claim to "own" any of the positive benefits, yet it has been observed that decision-making practices can suffer from the skewed framing caused by inherent biases, as shown in work by Fiegenbaum, Hart, and Schendel [12].

Chapter 3

An Adaptation of the Rubinstein Model

This section will include the proposal of a new bargaining game and an approach through which to analyze strategy selection for the game. Here, the game will be defined, a set of possible strategies introduced and tested, and the results discussed. For a complete set of results and explanation of the naming system for the strategies, see Appendices A.1, A.2, and A.3.

In this project, the crucial departure from the classical analysis of Rubinstein's bargaining model will be the usage of automata as the primary players in an agent-based model. The purpose of employing an agent-based model is not just to make more accurate predictions about future bargaining interactions, but more so to gain a deeper insight into the underlying principles involved in this style of game. While the assumption of purely rational choice can be useful for normative modelling, a model driven by independent agents with bounded rationality[1] can offer deeper insights into how the world actually is, as opposed to what it ought to be.

This new model will not incorporate the traditional assumptions of rational choice and perfect information that are typically found in game theoretic analysis. Rather, the agents in the model will be provided with a bounded rationality and access to limited information surrounding their decisions. This approach has been used to analyze the iterated Prisoner's Dilemma and similar games of social interaction to great effect [3, 2]. The goal is that this style of modelling will provide deeper insight into how and why decisions might be made in a bargaining game and provide a better intuition with which to analyze bargaining-type scenarios in the future.

3.1 Automata as Bargainers

This project will employ agents of bounded rationality to represent players in the Rubinstein bargaining game. These agents can be represented as automata, simple machines that are governed by distinct rule sets or instructions that can interact with other such machines. Whereas much of the previous research in the field of bargaining has focused its analyses on how preference of outcomes affect

[1] Bounded rationality is the idea that an agent will make decisions with limited cognitive processing abilities and/or access to information. Their decisions are therefore rational, but only within the bounds of the computing power and information that they have access to.

the final results, this approach will allow for analysis of the *procedural rationality* of the players, where the primary concern is the rationality of the procedure used to reach a decision, as opposed to the rationality of the decision itself [30]. By understanding what may *cause* the actions in a bargaining game, it may be possible to more accurately predict the outcomes of future games [25, 17]. Therefore, in this project, structures of decision-making will be programmed into finite automata so analysis can be made of their playing of the Rubinstein bargaining game.

By using automata to represent bargaining strategies, certain assumptions may be eliminated that would typically be used to analyze the outcomes of a bargaining game. Since the automata will only make decisions based on the input that they receive and their decision-making structure, it is not necessary to assume that they are rationally optimizing their utility function through their play. In fact, it is not necessary to assume that they have a distinct utility function as their decision-making process makes no reference to such a structure. Upon viewing an automaton's actions, it may be possible to infer an utility function via *posterior rationality*, but this will not be critical in the current project. In addition, it will not be necessary to assume that the automata have perfect information about the game. In the real world, it is more common than not that players *do not* have perfect information upon which to base their decisions [30]. Therefore, if an automaton is to represent a "real" strategy set, it would be useful to equip them with a decision-making structure which allows them to function in the more likely environment.

Although this project will not consider these typical assumptions, it will employ the assumption of a certain "sensibility", that is, the automata will prefer to receive more of a payoff than less. In addition, although they do not have access to complete information, they will make use of whatever information

they have to guide their decisions towards a higher payoff. The intent of this framework is to consider a wide range of strategies that are "sensible", though not necessarily rational, and to find from among them some aspects of strategy which would lead to "sensible" decision-making in a world of similarly "sensible" agents.

3.2 A New Bargaining Game

The following section will describe the process of creating a new bargaining model built on the structure of Rubinstein's bargaining game. The intent of this modelling approach is to capture a close approximation to real-world bargaining behaviour and to analyze it within a similar framework of classical game theory. Analysis of this model will follow the precedents of similar such modelling approaches, like Axelrod's analysis of the iterated Prisoner's Dilemma game.

The general outline of this model will be as follows. Two players will be paired to bargain over a fixed amount of surplus utility. The players will then be able to make offers over the allocation of this utility that can be accepted by the other player. One player will be chosen at random to make the first offer, after which the players will make alternating offers until an offer is accepted at which point each player is allocated their agreed upon portion of the utility. Both players will be impatient, meaning that the value of the surplus will diminish over time.

Effectively, this will be a positive-sum strategic form game with imperfect information. At each decision node, the players will be uncertain about how their opponent would respond further down any given branch of the game tree. Consequently, players will need to infer behaviour patterns from their opponent to deduce what their future reactions could be before making a decision. As

such, this model will allow different forms of strategic inference to be assessed when players have limited access to computational power.

3.2.1 Defining the Game

The following is a formal definition of the bargaining game used in this model. Hereafter, this will be referred to as "the bargaining game".

Two players are paired to bargain over a pie of surplus and transferable utility of size 1. Once players have agreed upon an allocation of the pie, they will receive their agreed upon portion. All agreements will be in the form (x_1, x_2) where x_i is Player i's share of the pie. These agreements belong to the set of all possible agreements $(x_1, x_x) \in X$ and must satisfy

$$\begin{gathered} (x_1, x_2) \in \mathbb{R}^2 \\ x_1 + x_2 = 1 \\ x_i \geq 0 \text{ for } i\text{=1,2} \end{gathered} \tag{3.1}$$

Note that x_i only refers to the size of Player i's allotted pie and not his utility for that portion of pie. Players are concerned with maximizing their share of the pie, and therefore their utility, at the expense of their opponent receiving a smaller share. For any $x, y \in X$ where X is the set of all possible portions of a pie of size 1, a player will prefer x to y if and only if $x > y$, meaning that a player will always prefer to receive a larger share of pie.

Each offer in the game will take place at a particular point $t \in T$ in the infinite set of time $T = \{0, 1, 2, ...\}$. At time $t = 0$, one player will be selected as Player 1. At this time, Player 1 will propose an offer to Player 2. Player 2 then either chooses to accept Player 1's offer and receive the agreed upon portion of pie, or else reject the offer. Should Player 2 reject the offer, time will advance to $t = 1$ at which point Player 2 will make an offer to Player 1. This process

will continue until an agreement is reached. There is no limit to the number of offers that can be made, nor is their a limit to the content of the offers other than the constraints listed in (3.1).

Each player will have a preference for time, denoted by a discount factor δ. This discount factor will create the perception of a shrinking pie, where each player has less utility for a particular portion of pie the longer it takes to acquire it. If x and y are both portions of pie that a player can offer or be offered, for any $x, y \in X$ and $t \in T$, a player prefers $(x,t) \succ (x,t+1)$, as well as $(x,t) \succ (y,t)$ when $x > y$. At any time $t \in T$ in the game, Player 1 may make an offer $x \in [0,1]$ to Player 2. Player 2 must decide whether to accept the offer, or return a counteroffer $y \in [0,1]$ to player 1, effectively comparing $(1-x)$ to $y \times \delta$. The sensible player accepts the offer when $(1-x) > y \times \delta$ and rejects it when $y \times \delta > (1-x)$.

At any given decision node, a player only knows their own discount factor, and all of the previous offers made in the game. Since the players do not have information regarding their opponent's preferences, their decisions will not necessarily mimic equilibrium behaviour as they will not always be able to correctly infer their opponent's future moves. In addition, the bargainers will not be assumed to be perfectly rational agents, as the decision-making processes will be entirely governed by the structure of the automata, and thus subject to a bounded rationality, meaning that they possess limited cognitive processing capabilities. Also note that the game tree contains unbounded branches that are an infinite sequence of rejected offers. These branches can be understood as a disagreement, where both players do not receive any of the pie.

3.2.2 Assumptions

This model will be built on the presupposition that neither player will know how their opponent values time, therebye forcing them to rely upon inferences of their opponent's position based on the offers that they receive. These inferences will be made in different fashions within each class of automata. A similar model was reviewed by Cramton [10], but his analysis focused on the passage of time between offers as the primary means of communicating information about the players' position, rather than a sequence of revealing offers. This model will instead focus on a sequence of alternating offers at consecutive discrete instances in time.

In addition, the following formal assumptions will be imposed in this model. Recall from earlier [22] that these assumptions are used in the classical Rubinstein alternating-offers bargaining model.

(A-1) *Pie is desirable* For any $t \in T$ and $x, y \in X$, we have $(x,t) \succ (y,t)$ if and only if $x > y$

(A-2) *Time is valuable* For any $t, s \in T$ and $x \in X$ with $s > t$, we have $(x,t) \succ (x,s)$ if $x > 0$

(A-3) *Continuity* Player preferences are continuous

(A-4) *Stationarity* If $(x,t_1) \succeq (y,t_1+1)$, then $(x,t_2) \succeq (y,t_2+1)$, meaning that the preference of (x,t) over $(y,t+1)$ is independent of t.

For the sake of the analysis in this project we will invoke another assumption to confine the actions of the players. Although a principal finding from the classical analysis of the Rubinstein model was that the bargaining outcomes are governed by the relative impatience between the players, this model can not provide the same result as the players are not aware of their opponent's level of impatience. Since the conventional analysis of the sub-game perfect equilibrium can not be performed in this case of incomplete knowledge, the ensuing outcomes

can not be calculated in the same manner. To constrain the possible outcomes for analysis, it will be assumed that both players are equally impatient, but are not aware of this fact.

(A-5) *Unknown similarity of time preference* Both players have equal urgency to reach agreement, but they are unaware of their opponent's preferences.

3.3 Round-Robin Bargaining Tournament

Now that the underlying game has been defined, we must create a framework through which to evaluate the playing of the game. Since we have rejected the assumption of complete information, this model allows for a wide range of possible outcomes that can not be analyzed with the traditional method. Therefore, a new method of analysis must be adopted that can shed some light onto the new-found complexity of the bargaining game.

Since the players in this bargaining game do not have access to complete information, they can not make completely rational decisions and therefore, there is no clear cut solution to the game. Fortunately, there are ways to approximate rational behaviour in complex interactions, even when a perfect solution does not exist.

Enter the agent-based modelling approach. As previously discussed, Robert Axelrod [3] ran his famous Iterated Prisoner's Dilemma tournaments to sift through possible strategies to the iterated game in pursuit of the best possible strategy. Clearly in the case of the iterated Prisoner's Dilemma game, there does not exist a perfect strategy that will dominate every other possible strategy, but that does not mean that it is impossible to discover very good strategies and their underpinning principles. Tit-For-Tat proved to be a very robust strategy that could succeed against a variety of opponents, but more importantly, it helped highlight a number of key characteristics that would make a strategy successful.

By simulating an agent-based model with a complex game, important intuitions can be discerned from the results.

The present study will adopt a similar methodology to Axelrod, but in the context of the bargaining game previously outlined. The first step in this process is to create a tournament format within which a number of independent and uniquely programmed agents can play each other in the bargaining game. The agents will be represented by finite automata, embodying a range of strategies. The tournament will be a round-robin format, where each automaton plays the bargaining game against every other automaton, including a copy of itself. In each such match-up, an automaton's outcome will be calculated as the average amount of discounted pie it receives while playing as Player 1 and as Player 2. The score for each agent will constitute its average outcome across all of the games it played in the tournament. Agents will then be ranked according to their score, with the best performers ranked highest and the worst performers ranked lowest. This ranking will allow for a preliminary analysis of the robustness and performance of different strategies.

3.3.1 The Automata

The purpose of using automata in this analysis is that they afford to us a distinct structure with an understandable and bounded range of possible actions. The set of automata to be used in the tournament will represent a range of bargaining strategies, ranging both in complexity and in intent. These strategies will take root from a number of different fields. Some of these strategies will be representations of simple rules-of-thumb which are commonly espoused in real-world bargaining situations with incomplete information [11], while some others may have integrated a number of such simple approaches into a unique strategy. Finally, a set of strategies will be implemented that seek to model empirically

observed human bargaining behaviour. This set of strategies will be inherently more complex than their counterparts, but will still rely on a few fundamental bargaining heuristics. As previously discussed, these heuristics are effectively just simple decision-making structures which can be employed in an attempt to solve a complex problem that would otherwise be computationally expensive or even impossible. Relying on experimental results in a number of studies on bargaining, these heuristics will be integrated so as to attempt to form a crude model of the human "bargaining mind". Although any finite automaton will almost surely represent an oversimplification of such a complex structure, the intent of this modelling approach is to create a framework to understand in a general sense how one might better approach a bargaining situation in the real world.

For the sake of analysis, the automata can be seen as "black boxes", with bounded rationality and an unalterable structure. Therefore, we must consider a meta-game based on the selection of automata to play the bargaining game on one's behalf. Automaton selection games have been extensively analyzed in the context of the iterated Prisoner's Dilemma game [1, 7], but also within the framework of Rubinstein's bargaining game by Binmore et al. [6]. Typically, automaton selection games are primarily concerned with maximizing expected payoff, with secondary goals including minimizing complexity, or achieving evolutionary stability. In the present study, meta-players will be chiefly concerned with selecting an automaton that maximizes their expected payoff in the bargaining game, but considerations will be briefly discussed for complexity and evolutionary stability. The purpose of the meta-game is to provide a framework through which to optimize strategy selection within the bargaining game, and thereby understand strategy optimization within a bargaining context.

The following sections will describe the structures of the automata to be

considered in this report. See Appendix B.1 for the Matlab code governing the automata and their decision-making process.

Structure

The automata to be considered in this report will vary in complexity, but consist of an identical underlying structure. Each automaton will begin the bargaining game in an initial state. Once the game starts, it will receive inputs (offers from the opponent) which will cause the automaton to change to a different state. Each state will correspond to a particular method for processing an output. Figure 3.1 shows a visual representation of this operation.

The precise definition of these automaton is as follows. The automaton will receive inputs and produce outputs representing a demand for a share of the pie $x \in [0, 1]$. An automaton
$A =< S, s_Y, \zeta, f >$ consists of four components [6].

- A set of states $S = \{s_0, s_1, ...\}$, including the initial state, s_0, which can be understood as an *initial offer*, an entitlement, or *level of aspiration.*
- An acceptance state s_Y wherein the automaton accepts his opponent's offer and signals the end of the bargaining game. $s_Y \rightarrow Y$
- An output function ζ corresponding to each of the states. The output of an automaton $x \in [0, 1]$ refers to the outgoing offer, the automaton's demand for his share of the pie. The output function may refer to the received input while producing its output. $\{S \times [0, 1]\} \rightarrow [0, 1]$
- A transition function f which changes the state of the automaton according to the received input. $\{S \times [0, 1]\} \rightarrow S$

Essentially, each state has a fixed process for translating an input into an output, while a transition function has fixed conditions which govern when an

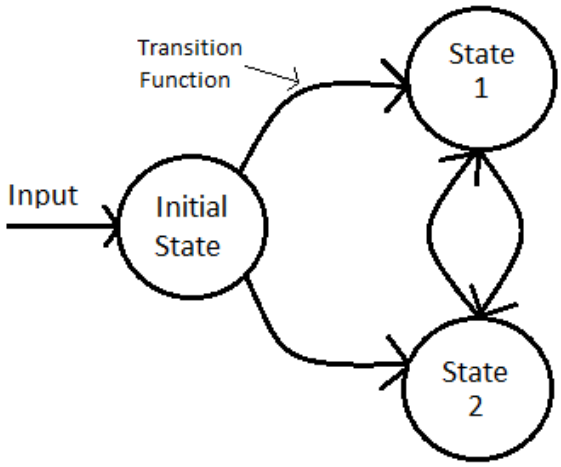

Figure 3.1: *Simple Automaton Diagram*: The automaton will receive an input then produce an output in accordance with its set of states and transition functions. In this context, the input will consist of: an offer of pie, the current time period, and the history of past offer. The output will either be a signal of acceptance or a counter-offer.

automaton will change states. If an input is received that fulfills the transition conditions, the automaton will shift to the corresponding new state, and that state will process the input and produce an output. In the context of the bargaining game, an input would be an offer received from an opponent, while an output would either be a counter-offer or a signal of acceptance. The states, transition conditions, and methods of processing inputs will be unique for each of the automata explored here.

Features

A set of 47 unique automata were designed for this project, with each representing a different strategic approach to the bargaining game. These strategies were designed to either mimic empirically observed bargaining behaviour in experiments with similar situations to the bargaining game, or to represent common rules-of-thumb for how to approach bargaining. The structure of their decision making process can be found in Appendix B.1 and the naming system can be found in Appendix A.1.

A key feature in each of the automata is their level of initial offer. Variance in initial offer size has been shown to stem from misperceptions of available utility, individual level of aspiration, or tactical chest-puffing to signal a strong position to the opponent [24, 19, 5, 16]. As such, automata from each major class were programmed to begin the bargaining game with an initial offer at either a High (60% of the pie), Medium (55% of the pie), or Low (50% of the pie) level[2]. This allowed the questions of how initial offer sizes would affect particular strategies and how they would affect strategy in general to be explored.

Seven distinct major classes of automata were programmed to represent particular strategic approaches. Some classes were more complex and allowed their

[2]These initial offer sizes were selected as they produced significant variability within the results, while maintaining a limited scope of the project. In the future, a wider range of offers could be tested using a genetic algorithm (see Chapter 5).

corresponding bargaining agents to recognize bargaining patterns and respond to them in a specific manner. Other classes were more simple and used less information to guide their play in the bargaining game. The seven classes of automaton are the Hawk/Dove class, the Reasonableness class, the Tit-For-Tat class, the Prospect Theory class, the Mean class, the Yield class, and the Split-the-Difference class.

The Hawk/Dove class of automata were designed to interpret bargaining behaviour as if it were playing the Hawk/Dove game. Therefore, an automaton in this class would consider their opponent's offer size and concession patterns and decide whether the opponent was playing as a Hawk or a Dove, and then make a rational response in the bargaining game. For example, if an opponent were to make minimal concessions and thus show little regard for the time-discounting of the surplus utility, an automaton of this class would respond with larger concessions to reach an agreement faster. The third and fourth component of an automaton's name[3] from this class designates its default concession pattern, while the Hawk/Dove aspect will scale their concession size accordingly. This default concession pattern can be understood as what the automaton of this class deems to be fair bargaining behaviour. Therefore, if an opponent makes concessions larger than what a Hawk/Dove automaton would deem to be fair, the Hawk/Dove automaton determines that the opponent is a Dove, and will consequently make a smaller concession. Similarly, should the opponent make a concession smaller than what the Hawk/Dove automaton would make should the roles be reversed, the Hawk/Dove automaton would respond with a larger concession.

The Reasonableness class of automata was designed to represent empirically observed pattern recognition and response behaviour in bargaining. An automaton of this class would consider factors such as the opponent's initial

[3]See Appendix A.1.

offer size and concession size to infer their opponent's "reasonableness". If an opponent were deemed to be reasonable, an automaton from this class would make initially large concessions, but decrease the size of its concessions in an attempt to secure a greater allotment of the surplus if the opponent continued to make concessions in turn. If the opponent were deemed unreasonable, an automaton from this class would decrease its concession size until the opponent would change its behaviour. Similar to the Hawk/Dove class, the third and fourth components of the names of automata from this class dictate a base-rate concession pattern which is then scaled by interpretation of the opponent's reasonableness. Again, this base-rate for concession making can be understood as the automaton's perception of what fair bargaining might look like, and their impression of their opponent is made relative to this sense of fairness.

The underlying concession patterns used by the Hawk/Dove and Reasonableness automata will call for either Increasing concession size, or Decreasing concession size. In addition, the change in concession size will either be Linear or Exponential, in which case the change of concession size between each offer is initially large, then slows down, similar to an exponential decay process. The justification for the pattern-recognition behaviour of this class can be found in Section 2.2.

The Tit-For-Tat class of automata was designed to translate the famous strategy from the iterated Prisoner's Dilemma game to the bargaining game. In this case, this class does not attempt to recognize patterns, but merely mirrors its opponent's concessions until an agreement is reached.

The Prospect Theory class of automata was designed to represent the risk behaviour of agents facing uncertainty proposed by Daniel Kahneman and Amos Tversky[15, 29]. The central idea to this design is that the agent has a preconceived notion, or frame, of what their fair share of the surplus should be.

Automata from this class will each have a distinct "frame" value, either High, Medium, or Low. Following the behaviour predicted by the Prospect Theory, the agent would respond to unfavourable offers with more risk-seeking behaviour, while responding to favourable offers with risk-averse behaviour. In the context of the bargaining game, risk-seeking behaviour would be represented by minimal concession making in an effort to force the hand of the opponent to accept a smaller allocation of the surplus, while risk-averse behaviour would be seen as making more lenient concessions in an effort to reach agreement faster. Further discussion on this type of reasoning can be found in Section 2.5. This strategy involves considering the amount of discounted pie that it could receive in each round, and comparing this value to its "frame". Given the relationship between this value of discounted pie and the frame, it will decide upon the appropriate concession size.

The Mean class of automata is analogous to the "Always Defect" strategy in Axelrod's [3] Prisoner's Dilemma tournaments. In this case, the Mean class will make an initial offer and refuse to make any concession, only repeating its same initial offer.

The Yield class of automata is effectively antithetic to the Mean class. The Yield class will accept any offer immediately, preventing any loss from the time-discounting of the surplus, while the Mean class will refuse to make any concession from their initial offer. These two tactics work as benchmarks and ensure the robustness of the other strategies.

The Split-the-Difference class of automata was designed to represent the commonly espoused approach to meet-in-the-middle in a negotiation setting. An automaton from this class will concede half of the discrepancy between his and his opponent's previous offers.

The single automaton in the SPE class represents the sub-game perfect equi-

librium strategy, where the automaton will figure out whether it is Player 1 or Player 2, and make a demand in accordance with the classic sub-game perfect equilibrium method of solving the Rubinstein bargaining game. If an agreement is not immediately reached, the automaton will insist on the same division of the pie until the opponent accepts the proposal. In addition, this strategy will only accept offers where they will receive at least the proportion of utility dictated by the sub-game perfect equilibrium solution. This strategy represents the ideal play of the bargaining game and can thus be used as a barometer with which to measure the performance of other strategies. If a bargainer were to employ the SPE strategy against an opponent using the same strategy, they would attain the optimal result.

The following is an example of a bargaining interaction between two of the automata to further demonstrate their structure and operation. The first table shows the bargaining game being played by H-Rea-D-E[4] as Player 1 and M-50 as Player 2. The second table shows M-50 playing as Player 1 and H-Rea-D-E playing as Player 2. In this instance, the discount factor will be $\delta = 0.95$.

In both of the following tables, the numbers in each box represent an offer made by the corresponding automaton. The offers are the non-discounted portion of the pie that they are demanding from their opponent. When an automaton decides to accept its opponent's previous offer, it will respond with a Y acceptance of the offer.

H-Rea-D-E	0.60		Y
M-50		0.55	

M-50	0.55		0.475	
H-Rea-D-E		0.60		Y

In this match-up, when H-Rea-D-E is Player 1, he will finish with a dis-

[4]The naming system for the automata can be found in Appendix A.1.

counted payoff of 0.4275[5], while M-50 receives 0.5225. When M-50 is Player 1, she will receive 0.4287, while H-Rea-D-E receives 0.4738. Therefore, the expected payoff for H-Rea-D-E in this match-up is 0.4507, while the expected payoff for M-50 is 0.4756, as each player has an equal chance of making the first offer.

The pseudo-code for these automata is as follows:

H-Rea-D-E

- If first offer, make offer of size 0.60. Else decide concession size.
  - Concede a portion of the difference between my last offer and the current offer (This portion decreases exponentially as time passes).
  - If opponent made a larger initial offer than me, decrease my concession size.
  - If the relative size of the opponent's concessions is decreasing, increase my concession size.
- If opponent's offer is larger than my previous offer less my current concession, discounted by one time period, accept the offer (Y). Else send offer of previous offer less my current concession.

M-50

- If first offer, make offer of size 0.55. Else decide concession size.
  - Concede one half of the difference between my last offer and the current offer.
- If opponent's offer is larger than my previous offer less my current concession, discounted by one time period, accept the offer (Y). Else send offer of previous offer less my current concession.

[5]The discounted payoffs are the portion [0,1] of the original pie that a player receives.

H-Y	60		Y (47.5)
Lo-Mean		50	(47.5)
	0	1	

Lo-Mean	50	(50)
H-Y		Y (50)
	0	

Figure 3.2: H-Y vs Lo-Mean

The following are representative bargaining games played by a number of different automata to demonstrate how they would play a bargaining game and to highlight some of the different behaviours between the different classes of automata. These games are all played with a discount factor of $\delta = 0.95$. The numbers in each box represent that player's demanded portion of the pie in that round, and the numbers at the bottom of the tables track how much time has passed in the game, and therefore the level of discounting that will be applied to the final agreement. The discounted amount of pie that each player will receive is noted in parentheses in the last column of each table. Note that in the case of SPE vs SPE, the outcome will be the same, regardless of which copy of this automaton is Player 1. The SPE strategy recognizes whether it is Player 1 or Player 2, and makes the corresponding sub-game-perfect equilibrium offer.

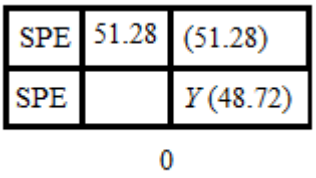

Figure 3.3: SPE vs SPE

Figure 3.4: SPE vs H-Mean

Lo-T4T	50		45		44.33	(36.1)
H-Pro-H		60		59.33		*Y* (45.35)

0 1 2 3 4

H-Pro-H	60		59.5		58.98		*Y* (42.88)
Lo-T4T		50		49.5		48.78	(34.5)

0 1 2 3 4 5

Figure 3.5: Lo-T4T vs H-Pro-H

3.3.2 Preliminary Tournament Results

The initial round-robin tournament was run at three different levels of the discount factor, namely 0.93, 0.95, and 0.97 [6]. This was done so as to explore the robustness of the strategies and to observe whether there would be significant changes in their performance given different conditions. A table showing the results of the different runs of this tournament can be found in Appendix A.2.

Before discussing the results, it is first important to note that only limited conclusions should be drawn from this tournament. First and foremost, it must be noted that the outcomes generated by the automata are directly related to the mechanics of their opponents' strategies, which are quite limited. Therefore, caution must be employed before making any inference about the absolute usefulness of a strategy, as this approach only demonstrates their relative value within this artificial model framework. Second, although the round-robin ap-

[6] At discount factors below 0.93, there is no relative change from the results of the simulations using a discount factor of 0.93, and therefore these cases have been omitted.

proach is useful for broadly stratifying the results of a diverse group of strategies, it is important to consider that each match-up in the tournament is given equal weight in the final ranking of performances. Therefore, an agent who can successfully take advantage of a particular large class may benefit from inflated scores against that class while performing poorly against the other classes, but still scoring well on average.

The results of the tournament provide a few points of interest concerning strategy selection in the bargaining game. First, it is interesting to note that the worst performers were generally the most successful at securing a larger proportion of the pie, but at a very high cost of time, and thus leading to a very poor final score[7]. On the other hand, the very best performers typically secured close to or less than half of the pie on average, what would be considered close to the fair division[8], and with minimal delay[9]. Second, although the very top performers typically began the game with a large initial demand of the pie, on average, those with Low initial offers out-performed their counterparts[10]. Automata making High initial demands showed the greatest variability in performance, representing the very top performers and the very worst. Finally, there was a distinct correlation between successful automata and their ability to reach agreement quickly[11]. This may seem obvious given the time-discounting factor, but there does not appear to be a direct trade-off between achieving a fair allocation of the pie and reaching agreement quickly as a few of the automata were able to manage both of these successfully.

[7] Strategies obtaining on average greater than 50% of the pie scored less than average with significance $p < 0.001$

[8] Strategies ranking in the top third obtained less than 50.5% of the pie with significance $p < 0.001$

[9] Strategies ranking in the top third reached agreement faster than average with significance $p < 0.001$

[10] Low initial offer strategies placed better than their counterparts using a Medium or High initial offer size with significance $p < 0.001$

[11] Correlation between average number of turns and final ranking was 0.891, using the Initial Round-Robin Rankings in Appendix A.2 and the corresponding Average Turns for each strategy.

3.3.3 Thinning the Herd

To narrow the focus onto effective strategies, strategies which proved to be ineffectual, or otherwise replaceable were removed from consideration. The first method employed in the effort to eliminate strategies and reduce the scope of analysis was to look for strict dominance among the set of all automata. Due to the complex nature of the interactions in the bargaining game, strict domination was very rare, with only three strategies eliminated via this method. Since the common technique of solving a game via the elimination of dominated strategies did not prove fruitful, the subsequent approach involved searching for significant dominance in the average outcomes, rather than the set of individual match-up outcomes. For example, in the Prospect class, the automata with a High initial offer performed significantly worse on average than their counterparts with smaller initial offer sizes[12] and were thus eliminated from further study in the round-robin tournament.

These strategies were eliminated with the intent of running the round-robin tournament again, but with a smaller group of automata. This elimination process was useful to shift the focus onto the performance of strategies against other strategies that may actually be employed by an opponent, rather than considering how they would fare against strategies that were almost certainly deficient in some manner.

3.3.4 Tournament Re-Run

The round-robin tournament was run again using the smaller set of automata which had survived the elimination process. The outcomes of this tournament can found in Appendix A.3, again at the discount factors of 0.93, 0.95, and 0.97. Twenty-six Automata were involved in this tournament, representing the same

[12] As can be seen in Appendix A.2, these automata finish at the bottom of the rankings in the Initial Round-Robin Tournaments.

set of strategic classes that were present in the first tournament.

The most noticeable aspect of the results of this second tournament is the decrease in the difference between the scores of the top finishers and the bottom finishers. Although the top performers from the first round did not fare significantly better in this tournament, the mean outcome increased[13] as there were fewer extremely poor performers. This observation follows the conclusions in the paper by Binmore et al. [6] on evolutionary stability in the Rubinstein bargaining game which suggested that when exposed to selective pressure, strategies would converge towards the sub-game perfect equilibrium in the Rubinstein model. Although the process of strategy elimination used here does not necessarily mirror the selective pressures implied by this work, it does follow the principle of eliminating agents based on some measure of fitness.

Another point of note in the results of the second run of the tournament is the correlation between a high average allocation of the pie (greater than 50% of the pie) and poor overall performance[14], similar to what was seen in the first run, further suggesting the critical importance of reaching agreement quickly. The second run of the tournament consisted of automata who had previously been able to, on average, reach agreement in the bargaining game faster than their counterparts, as seen in the results in Appendix A.3, where there are fewer strategies averaging greater than 6 turns per game. Even in this more "cooperative" setting, the strategies who were more demanding still were not able to overcome the costliness of extended bargaining in an attempt to secure a greater proportion of the pie. At the other end of the spectrum, there was more strategic diversity among the top performers. Some strategies were successful by reaching agreement very quickly (average game lasted less than 2 turns) at the cost of a smaller proportion of pie, for example, the Yield class,

[13] Thinned Round-Robin Tournaments had a higher average score than Initial Round-Robin Tournaments with significance $p < 0.001$

[14] Correlation between a strategy's Average Allocation of pie and final ranking was 0.696.

while others were able to secure a more even division after only a small delay, such as the Reasonableness class.

3.4 Tournament Discussion

The round-robin tournament provided a broad look at the interaction of a broad range of bargaining strategies and began to shed light on some of the relatively better strategies. Although this style of analysis proved fruitful in the Axelrod [3] iterated Prisoner's Dilemma tournament, the additional degrees of freedom present in the bargaining game increase the noisiness of the analysis and can thus only induce general conclusions about the results. Fortunately, a number of noticeable trends became apparent in the running of the round-robin tournaments for the bargaining game.

Obviously, success in the bargaining game relies upon one's ability to secure a large portion of the pie in a short amount of time. The automata used to represent strategies in this tournament were diverse in approach, necessitating proficient and robust bargaining tactics to generate a best-response to each individual strategy. Interestingly, one automaton in particular was able to achieve the best performance in tournaments using each of the discounting factors explored. This strategy, H-Rea-D-E, was able to secure a large portion of the pie against more forgiving opponents, while being able to quickly identify and reach an agreement with delaying opponents before the payoff had decayed substantially[15]. The pattern recognition heuristics employed by this automaton were based on empirically observed bargaining behaviours in humans, and their success suggests that more effort to understand them may prove useful.

A brief glance at the table of results of the round-robin tournaments in

[15]As can be seen in the table of results in Appendices A.2 and A.3, this strategy averages very close to half of the allocation of the pie while requiring a relatively low average number of turns to do so.

Appendices A.2 and A.3 will show that the strategies who were excessively demanding in their bargaining performed quite poorly overall. Nearly all of the automata who on average obtained greater than one half of the pie ended up in the bottom half of the standings due to the excessive time delays required to obtain their larger portion. Ironically, by trying to obtain more of the pie, these automata ended up with a worse outcome. Interestingly, this result runs somewhat parallel to the principle espoused by Axelrod [3] at the conclusion of the iterated Prisoner's Dilemma tournament, "Don't be envious". In the Prisoner's Dilemma game, being envious meant defecting in an attempt to achieve a better outcome, which is similar to creating delays in the bargaining game to achieve a greater relative portion of the pie. This idea is further corroborated by the results of the bargaining tournaments, as the Yield class of automata performed quite well on average, simply by accepting any offer to avoid the erosive forces of time. Clearly this strategy is exploitable as it is prone to accepting unfair offers, but it helps illustrate the general principle of the need for fairness.

An intentional design feature of the round-robin tournament was the usage of the SPE automaton as a barometer to evaluate the value of the other strategies. Since the SPE strategy represents the optimal solution to the bargaining game, its relative success in a round-robin tournament will show how close the other strategies are to the optimal solution. The better this automaton were to perform, the better the other strategies must be. The lack of success for this strategy in these round-robin tournaments could suggest the more demanding strategies are ineffective as their narrow focus on proportion of the pie spoils the final outcome against reasonable bargainers. This idea about the SPE automaton as a barometer will be explored in further detail later on.

It is important to note the the effect of the strategies who take many rounds to reach agreement. As mentioned earlier, the Thinned Round-Robin tourna-

ment saw a better average score for its participants than the Initial Round-Robin tournament, as a number of the poorer performers were removed from contention. Typically, these poor performers performed poorly due to the length of time that it took them to reach agreement. In their attempts to secure a higher proportion of the pie, they actually decreased the final discounted amount of pie they would receive, leading to worse outcomes. Since this behaviour leads to a bad performance for both the individual bargainer using this strategy and for his opponent, it would be best to avoid selecting such strategies in practice. The tendency to unnecessarily prolong the negotiations is therefore irrational, it would be practical to consider the potential downside of selecting a bargaining strategy that may lead to delays.

A final point of note about the results of the round-robin tournament is that most of the successful automata were very accurate in their estimations of what a "fair share" of the pie would look like, as they were allocated close to half of the available pie on average. A critical challenge in real-world bargaining is the ability to identify whether or not a surplus exists in a bargaining scenario, how much exists and if a particular party has a specific entitlement to the surplus. Although not accounted for in this model, the ability to accurately assess the size of any surplus is critical to bargaining success. Therefore, any conclusions or intuitions developed from this model must account for this blind-spot.

Chapter 4

An Evolutionary Look at Bargaining Strategy

This section will discuss using an alternative approach to optimizing strategy selection in the bargaining game. A general model for this approach will be proposed and simulations will be run to verify its usefulness. Finally, results of the simulations will be discussed along with any possible conclusions that can be extracted from them.

To begin to understand the dynamics of strategy selection in the real world, we must lean on the tools of evolutionary biology to adopt a proper context with which to analyze this choice. Binmore et al. [6] showed how this approach could be used specifically in the Rubinstein bargaining game, while many others have used similar methods in the analysis of the iterated Prisoner's dilemma game [9, 21]. While these approaches relied upon some form of bounded rationality, the measures used thus far have further bounded the rationality of the agents employed in the model, as these agents are confined to the use of heuristics or decision frameworks which have been observed in human nature, as discussed in Chapter 2. The intent now is to discover how these heuristic-based strategies will adapt to selective pressure.

According to James March [17], given a stable world and stable preferences, an adaptive-agent model will, over time, promote the emergence of approximately rational behaviour from its constituents. Therefore, any change in behaviour in an agent-based model will likely be an adaptive response given the current environment. However, caution must be used in analysis of this model, as adaptations may be drawn towards local optima which may be dissimilar from the absolute optimum. For example, the automobile is a superior form of human transportation than the use of horse-drawn carriages, but selective pressures would likely never be able to create adaptations in horses which make them resemble an automobile. This is just to say that the nature of selective pressures and the available modes of adaptation can significantly restrict the set of possible results, and therefore conclusions from such approaches should not he treated as absolute truths.

The following section will explore an ecological approach to determining dominance across a broad range of strategies. The ecological approach is similar to an evolutionary model, but does not allow agents to undergo adaptations,

but merely evaluates them on a measure of fitness and maps those evaluations to a global population structure. This modelling approach will provide a more realistic context for the bargaining game and the corresponding bargaining strategy selection meta-game.

4.1 An Ecological Modelling Approach

The ecological modelling approach used in this project is an agent-based modelling approach where the agents to be studied are represented by the previously discussed automata which play the bargaining game. In this model, each of the strategies represented by the automata discussed in the second round of the round-robin tournament will be represented by an equal initial "population size" in an ecosystem. This ecosystem will be simulated for a number of generations, wherein each generation, the agents will play the bargaining game against each of the other agents in the ecosystem. After each generation, the total discounted amount of pie obtained by each of the agents will be referred to as their individual *fitness* in that generation. Then, according to a population change algorithm, each agent will reproduce itself into the next generation of the simulation in proportion to its fitness level in that generation. In general, if a particular type of agent has a high level of fitness in a particular generation, it will be represented in greater numbers in the subsequent generation, while a type of agent with a low fitness level will see its numbers decrease in the next generation. This simulation process will continue until a population equilibrium is reached. Along the way, it is expected that certain populations of agents will die out altogether, while others expand into the ecosystem. Once an equilibrium is reached, the overall dynamics of the simulation can be analyzed.

This approach follows that used by Axelrod [3] in his analysis of the iterated Prisoner's Dilemma game tournament. This method proved to be an

effective way of finding the best strategies in the iterated Prisoner's Dilemma game, since as the simulation wore on, a strategy would have to continue to perform well against similarly successful strategies to continue to appear in future generations. In addition, this style of model can help represent the strategic environment of a society, where individual agents are able to choose a strategy which best suits them, and then these agents must employ successful strategies to continue playing the game with other members of society. In this scenario, the ecological model style is well-suited to represent societal dynamics in real-life, as real world agents would likely stop using a strategy that is unsuccessful and adopt a strategy that had been used successfully by others in the past. This style of societal agent-based modelling has been used again by Axelrod [2] in modelling societal norms and political preferences.

4.1.1 Population Change Algorithms

The driving force of population change in an ecological model is the relative fitness of a "species" compared to the rest of the population. In the context of this project, fitness is defined as the expected score in the bargaining game against the rest of the current populations in the ecosystem at the current time. Each "species" will be a strategy represented by one of the automata discussed in Chapter 3. Two methods of driving population dynamics will be discussed here.

The following equation represents the proportional change in population size for each sub-population using a particular strategy. The sole driver of the population dynamics in this ecological simulation is the comparison of a strategy's fitness in a particular time period, D_j, compared to the population's average fitness in the same time period, $\bar{D}$. If the overall population size is assumed to be constant, the size of the next generation of a sub-population

using a particular strategy, j, is represented by

$$P_j^{i+1} = P_j^i + \Delta \left(\frac{D_j}{\bar{D}} - 1 \right) P_j^i \tag{4.1}$$

where Δ is a fraction that scales the rate of change of the population structure, and P_j^i represents the current size of the sub-population using strategy j. This approach is referred to as *fitness proportionate selection* and is used in genetic algorithms concerned with selecting useful solutions [3, 8].

The second approach to population change dynamics used in this project is as follows. At any given generation, the fitness of each individual sub-population will be compared to the mean fitness of the whole population at that time. If a species' fitness is greater than one standard deviation above the mean, that species will increase in size by one member in the subsequent generation, whereas if a species has a fitness level less than one standard deviation *below* the mean, it will lose one member in the next generation. If a species' fitness level is within one standard deviation of the mean, its size will remain the same in the next generation. Going forward, this dynamic will be referred to as the *standard deviation algorithm.* This approach provides a lower resolution look at the population dynamics of an ecosystem, but is useful for quickly stratifying the performance of the different strategies in consideration.

4.1.2 Narrowing the Field

Building from the results of the round-robin tournament approach discussed in Chapter 3, the automata who were present in the second round of the tournament will now be considered in an ecological tournament. These surviving strategies will now be represented by agents in an ecosystem and subjected to selective pressures to further analyze their performance in a more detailed context.

Before beginning the simulation of this new ecological tournament, it is important to consider the fact that ecological models can be prone to positive feedback loops. The purpose of this modelling approach is to select a dominant bargaining strategy that out-performs similarly competent strategies. To effectively do so, the strategies to be selected from must be compared on somewhat equal footing to ensure that the best option is chosen. Therefore, a method of control must be instilled in the simulation process to ensure a semblance of fair competition.

To do so, the first step of the ecological tournament will be to drop all surviving automata into an ecosystem simulation using the standard deviation algorithm. The simulation will be run until an equilibrium is reached and the surviving strategies will be noted. This can be seen in Appendix A.4 in Figures A.1, A.2, and A.3 where the labeled are the survivors. Subsequently, these surviving "species" will be placed into an ecological simulation using the delta factor algorithm, each beginning with an equal initial population size. This simulation will then be run until the system reaches an equilibrium and the best strategies will be ranked according to their population size or by the amount of time it took for their kind to die out. This process ensures not only that the eventual winners were successful against other competent strategies, but also that their success did not come from taking advantage of weaker strategies. Essentially, by undergoing a more demanding selection process, the robustness of the top performing strategies is ensured.

The following figures demonstrate this filtration process and show the dynamics of the ecological simulation. Figure 4.1 shows the surviving strategies from Chapter 3 placed in a simulation using the standard deviation algorithm. Figure 4.2 then shows the surviving strategies from the standard deviation simulation being re-established on equal footing in a simulation using the delta

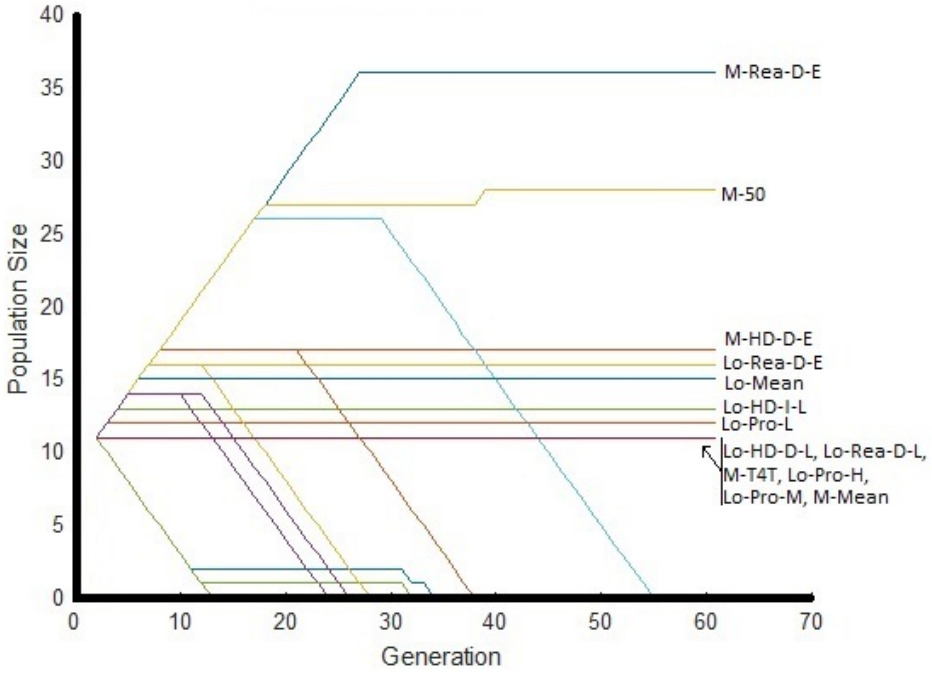

Figure 4.1: *Standard Deviation Algorithm Simulation* $\delta = 0.95$: In this simulation, each strategy noted in the Thinned Round-Robin Tournament begins with the same initial population size. Population change occurs in each generation according to the standard deviation algorithm until a final equilibrium is reached. The surviving strategies at this equilibrium are those noted.

factor algorithm.

4.2 Ecological Tournament Results

Similar to the Round-Robin Tournament discussed in Chapter 3, the Ecological Tournament was run using the methods described in Figure 4.1 and 4.2 three separate times and using three different discount factors. The discount factors used in this tournament were the same as those used in the Round-Robin Tournament, namely 0.93, 0.95, and 0.97. Each tournament began with the same

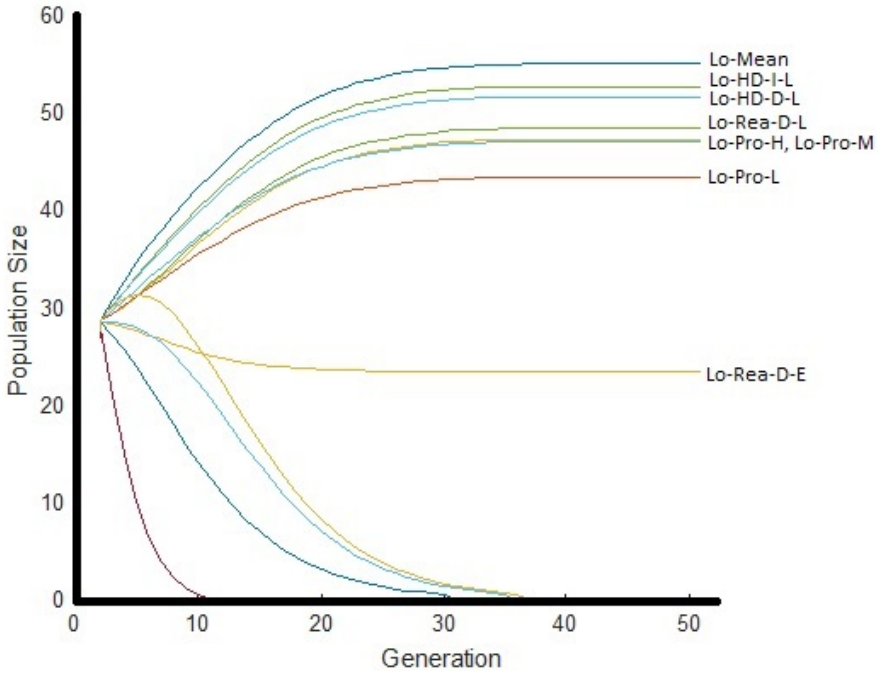

Figure 4.2: *Filtered Ecological Simulation* $\delta = 0.95$: The surviving strategies noted in Figure 4.1 begin this simulation with the same initial population size. The simulation is run according to the proportional population change algorithm until a final equilibrium is reached. The surviving strategies are ranked according to their final population size.

filtering process in which the Standard Deviation Algorithm was used to eliminate weaker strategies, before beginning a new simulation with each "species", or strategy, beginning with the same initial population size. The simulations were then run until a final equilibrium was reached and the performances of each strategy were assessed from there. Complete results from these simulations can be found in Appendix A.4, plus a figure showing the final rankings at the end of this section. At any point in these simulations, strategies with larger population sizes can be said to be performing better than those with smaller population sizes. If multiple strategies are present at the final equilibrium in the simulation, these strategies are ranked in superiority according to their population size. For the strategies that die out through the course of a simulation, the strategies that die out later in the simulation are ranked higher than those who died out earlier.

In the simulation using a discount factor of 0.93, the top three finishers were the SPE, H-Y, and M-Mean strategies. It is interesting to note that in this simulation where time was the most valuable, the tournament was dominated by strategies which were designed to reach agreement as fast as possible with minimal regard for pattern recognition, putting on a facade of toughness, or other tactics. The fact that the SPE strategy performed better in the ecological simulation (placing first) than in the round-robin style tournament (placing 20th) could suggest that there is convergent pressure towards the optimal bargaining solution given certain adaptive mechanisms. In addition, it is interesting to note that SPE dominated both the initial filtering simulation and the final ecological simulation after re-starting on equal footing in the latter. This is likely due to the fact that SPE performs "perfectly" against itself and well enough against other strategies, that in a time-demanding scenario, it will dominate a population given a small foothold. Figures A.4, A.5, and A.6 in Appendix A.4 contain

the final results of the ecological simulations.

In the simulation using a discount factor of 0.95, the top three finishers were Lo-Mean, Lo-HD-I-L, and Lo-HD-D-L. This simulation saw a final equilibrium with a large number of strategies still "alive" in the simulation, meaning that there was not enough variance in their fitness to cause more to die out. The key similarity between these surviving strategies was their Low initial offer size which in general leads to an agreement being reached more quickly compared to strategies demanding a larger initial offer[1]. It is interesting to note in this simulation that the strategies who performed well in the initial filtering simulation did not survive the final ecological simulation[2]. These strategies that died out had higher initial demands, which would lead to longer bargaining games and thus depreciate the value of the pie. It therefore seems that a group of somewhat cooperative strategies may be able to fend off "tougher" strategies through their own cooperation in a sense.

In the simulation using a discount factor of 0.97, the top three finishers were Lo-Mean, Lo-HD-I-L, and Lo-Pro-M. The general dynamics of this simulation were quite similar to those from the simulation using a discount factor of 0.95 and the final equilibrium involved a number of strategies using a small initial offer, while the filtering simulation had its top performers using a medium initial offer size. That being said, the strategies which performed the best in this simulation were slightly different than the top performers form the previous. In this simulation, the Prospect class of strategies performed relatively well in comparison to the other strategies and also the other simulations using different discount factors. This class of strategy was designed to imitate empirically

[1]Low initial offer strategies reached agreement on average faster (lower number of average turns to reach agreement) than Medium or High initial offer strategies with significance $p < 0.001$

[2]M-Rea-D-E, M-50, and M-HD-D-E were the top three most represented strategies in the filtering simulation, but did not survive through the final ecological simulation. This can be seen in Appendix A.4, comparing Figure A.3 to Figure A.4.

observed human behaviour in decisions involving risk, and its relative success here may suggest that the heuristics that humans have adapted to simplify risky decision-making may have some merit. In this simulation in general, strategies involving more pattern recognition and other such tactics performed better than the simpler strategies.

The following tables provide the final rankings from the ecological simulations. Again, if the simulation reaches a final equilibrium, strategies are ranked according to their population size at this equilibrium. If they do not reach an equilibrium, they are ranked according to how long it took for their population to die out.

Ecological Simulation Final Results $\delta = 0.93$			
Rank	Name	Final Population Size	Generation Died Out
1	SPE	487	∞
2	H-Y	0	45
3	M-Mean	0	33
4	M-50	0	32
4	M-T4T	0	32
6	M-Rea-D-E	0	31
7	Lo-Mean	0	25
7	Lo-Pro-H	0	25
7	Lo-Rea-D-L	0	25
7	Lo-HD-D-L	0	25
7	Lo-HD-I-L	0	25
12	Lo-Pro-M	0	24
13	Lo-Rea-D-E	0	23
13	Lo-T4T	0	23
13	Lo-Pro-L	0	23
13	Lo-50	0	23
17	M-HD-D-E	0	58 (Filtering)
18	H-Rea-D-E	0	51 (Filtering)
19	Lo-HD-D-E	0	26 (Filtering)
20	H-50	0	24 (Filtering)
20	M-Pro-L	0	24 (Filtering)
22	M-HD-I-L	0	20 (Filtering)
23	H-Mean	0	13 (Filtering)
23	M-Pro-M	0	13 (Filtering)
23	M-HD-D-L	0	13 (Filtering)
23	H-HD-I-L	0	13 (Filtering)

Figure 4.3: *Ecological Simulation Results* $\delta = 0.93$

Ecological Simulation Final Results $\delta = 0.95$			
Rank	Name	Final Population Size	Generation Died Out
1	Lo-Mean	55	∞
2	Lo-HD-I-L	53	∞
3	Lo-HD-D-L	52	∞
4	Lo-Rea-D-L	48	∞
5	Lo-Pro-H	47	∞
5	Lo-Pro-M	47	∞
7	Lo-Pro-L	43	∞
8	Lo-Rea-D-E	23	∞
9	M-50	0	37
10	M-T4T	0	36
11	M-Rea-D-E	0	31
12	M-HD-D-E	0	31
13	M-Mean	0	11
14	H-Rea-D-E	0	55 (Filtering)
15	H-50	0	38 (Filtering)
16	M-Pro-L	0	34 (Filtering)
17	M-Pro-M	0	31 (Filtering)
18	H-Y	0	28 (Filtering)
19	Lo-50	0	26 (Filtering)
19	Lo-T4T	0	26 (Filtering)
21	Lo-HD-D-E	0	23 (Filtering)
22	H-HD-I-L	0	13 (Filtering)
22	M-HD-I-L	0	13 (Filtering)
22	M-HD-D-L	0	13 (Filtering)
22	H-Mean	0	13 (Filtering)
22	SPE	0	13 (Filtering)

Figure 4.4: *Ecological Simulation Results* $\delta = 0.95$

Ecological Simulation Final Results $\delta = 0.97$			
Rank	Name	Final Population Size	Generation Died Out
1	Lo-Mean	64	∞
2	Lo-HD-I-L	61	∞
3	Lo-Pro-M	56	∞
4	Lo-Pro-H	50	∞
5	Lo-HD-D-L	47	∞
6	Lo-Pro-L	45	∞
7	Lo-Rea-D-E	11	∞
8	M-50	0	66
9	M-Rea-D-E	0	53
10	H-Rea-D-E	0	32
11	H-50	0	24
12	M-Mean	0	18
13	Lo-Rea-D-L	0	50 (Filtering)
14	SPE	0	47 (Filtering)
15	M-HD-D-E	0	33 (Filtering)
16	M-Pro-L	0	28 (Filtering)
17	Lo-50	0	26 (Filtering)
18	H-Y	0	23(Filtering)
18	Lo-T4T	0	23(Filtering)
18	Lo-HD-D-E	0	23(Filtering)
21	M-Pro-M	0	22 (Filtering)
22	H-Mean	0	13 (Filtering)
22	M-T4T	0	13 (Filtering)
22	M-HD-D-L	0	13 (Filtering)
22	M-HD-I-L	0	13 (Filtering)
22	H-HD-I-L	0	13 (Filtering)

Figure 4.5: *Ecological Simulation Results* $\delta = 0.97$

4.3 Ecological Tournament Discussion

The process of using selective pressure to determine dominance within a set of strategies leans on principles similar to that of the Evolutionarily Stable Strategy, or ESS. Although there is no risk of a population being invaded by a group of rival strategies not previously present in the ecosystem, the proportions of the overall population will adapt according to the definition of fitness and its dynamics defined in Chapter 1 on page 13. The ecological study does not allow the strategies themselves to adapt to change, but rather has the entire population change to maximize the overall fitness level. Although Boyd and Lorberbaum [9] showed that no pure ESS exists in the simpler iterated Prisoner's Dilemma game, Binmore et al. [6] in 1998 showed a modified ESS was attainable in an alternating-offers bargaining game. Binmore's definition of a modified ESS predicts a population of adaptive automaton bargainers to converge towards a fifty-fifty split of the surplus in the Rubinstein bargaining game as the players become more patient. As the players become more patient in the simulations with discount factors of 0.95 and 0.97, strategies using small initial demands but assertive concession patterns were able to survive until the final equilibrium while the other strategies died out[3]. This approach appeared to be collectively stable in the simulations involving lighter time-penalties, as a number of different strategies sharing these characteristics were able to sustain themselves in these simulations, while not in others. This success shows a resemblance to Axelrod's principles from the iterated Prisoner's Dilemma tournaments, "Be nice" and "Be provocable" [3]. In this case, "Be nice" correlates to making a smaller initial demand so that both players in the bargaining game begin closer to their eventual agreement, thus requiring less time to get there. Here, "Be provocable"

[3]As seen in Figures 4.4 and 4.5, strategies making Low initial were the top finishers. In addition, strategy classes such as Yield, Tit-for-Tat, and Fifty-Fifty, who do not recognize patterns in the play of their opponent, did not place amongst the top performers.

relates to standing tough against bargainers making excessive demands.

As the players became less patient, as in the simulations using a discount factor of 0.93, the SPE strategy dominated the population, as seen in Figure A.2 in Appendix A.4. As the time discounting became more severe, the cost of delaying a round to secure a higher proportion of the pie became too much to justify the delay. Therefore, strategies who were able to reach agreements immediately performed well in this scenario, such as SPE and H-Y, who placed first and second in this ecological simulation, respectively. In this case where time is very valuable in the bargaining, the advantage of making the first offer becomes quite apparent since those strategies who could not reach immediate agreement quickly died out, as discussed by Rubinstein in his sub-game perfect equilibrium solution to the bargaining game [22].

Although there was a limited number of strategies explored within the world of this model, a few key observations can be made about how certain strategies performed. First of all, it pays to "Be Nice" and make an initial offer that is close to the fair division of the pie, as seen by the success of the strategies employing a Low initial offer size in these ecological simulations. Second, it does not pay to be resolute in your stance, as strategies who did not make concessions in their demands performed very poorly, especially when their were selective pressures involved in the ecological simulations. Finally, when selecting a bargaining strategy, it is important to recognize your own level of patience, as different strategies will perform better in certain contexts. For example, if a bargainer were to find himself in a situation where any delay would be quite costly, it would be imperative to select a strategy designed to reach agreement immediately, or as soon as possible. Although these observations were made in an artificial and bounded world, intuition says that these principles could translate to real-world bargaining scenarios.

The ecological simulations used here were necessarily limited in the scope of strategies explored, but nonetheless provided some insight and detail to previous analysis on similar games. It was interesting to note that even without any adaptive mechanisms in the model, there did appear to be a convergence towards a fair splitting of the pie in the ecological simulations with small discount factors, as predicted by Binmore et al. Future work is needed to solidify these observations. Additionally, the principles espoused by Axelrod for the playing of the iterated Prisoner's Dilemma game seem relatively applicable to the bargaining game explored through this project. Again, more extensive study would be required to determine whether this is just a product of the artificial models used in this project, or whether these ideas point toward a general truth in non-cooperative games which incentivize cooperation.

Chapter 5

Conclusions

This section will revisit the initial motivations of the project, summarize the approach used, and reiterate the key findings. In addition, possible avenues of future investigation will be presented and considered.

5.1 Summary of Work

The purpose of this project was to explore the dynamics of strategy and strategy selection in bargaining and how these phenomena map to the real world. A game was proposed to represent real-world bargaining scenarios and which could be used to analyze potential strategies. A set of strategies was designed to represent observed human behaviour in bargaining and simple generally accepted bargaining tactics, creating a diverse range of strategies to explore and analyze.

The round-robin tournaments and ecological simulations were used as means to evaluate the set of strategies where more traditional analytical methods could not ascertain dominance. The strategies considered were designed to represent different heuristics for forming decisions when complete information was not available. It was found that strategies involving some form of pattern recognition were relatively useful when the cost of delay was small, but were dominated by the strategy capable of computing the presumed sub-game perfect equilibrium as time delays became more costly. Further analysis will be required to understand the extendability of these results, as success in the bargaining game is largely dependent upon the play of possible opponents.

It is important to note that the strategies explored in this project were completely deterministic, and not necessarily representative of the set of all strategies that could be used in the real world. As such, rigorous statistical tools could not be applied to analyze the results of the simulations in depth. However, the work presented here provides a foundation for the approach and methods of future research in this area.

Although the specific results of this project do not carry much weight as they are the product of a very artificial model, a few general comments can be made about bargaining strategy. First of all, as is trivially obvious, bargainers should

select strategies which will lead to agreement quickly to avoid unnecessary losses to time-delay. In addition, strategies which seek a fair agreement will be more robust to a wide range of opposing strategies, and greediness will usually lead to time delays. Consequently, it would be beneficial to recognize quickly whether or not an opponent is seeking more than their fair-share or is quite patient, so as to respond accordingly before time-delays erode the gains to be had from the bargaining. Finally, the strategies explored in this project relied upon the assumption that players would be able to accurately assess the amount of surplus utility to be had from bargaining. For these principles to hold true, real-world bargainers must have a certain aptitude for correctly determining the size of the gains to be had from bargaining.

5.2 Future Work

This project created a new foundation from which to explore bargaining strategy, but left a number of avenues of approach untouched or unexplored in depth. Here will be discussed a number of possible ways to expand upon the model and otherwise apply and validate it.

5.2.1 Extension of the Model

A noticeable shortcoming of this work is that the strategic scope of the agents has been limited by the bounded imagination of the author. Luckily, Nature is not similarly limited, as random mutation allows for the continued exploration of all possible strategic alternatives. To piggy-back off of Nature's ability to propose diverse solutions to ever-changing and complex problems, a genetic algorithm could be implemented into the strategy formulation process. Such an approach was used by Riolo [21] to attempt to optimize strategy in an iterated Prisoner's Dilemma game, building off of previous work by Holland [14, 13] in

the area of adaptive agents in economic theory.

Given the increased complexity in a bargaining game over that of the Prisoner's Dilemma, it is reasonable to expect that such an approach would yield more intelligent results than have been established in this project. By instituting selective pressures and random mutation as the primary forces of strategy optimization, a much wider net could be cast to capture any possible initial strategy, then allow fitness to dictate the ensuing adaptations. This process would involve successful strategies to reproduce with random mutation, while unsuccessful strategies would attempt to adapt towards the more successful strategies. This mechanism of change would likely map well to the real world, where individuals tend to adapt the behaviour patterns of those more successful than them and reject the patterns of those less so.

Another way to extend the results of this model would be to incorporate a level of random error within the strategies. This error would represent the possibility of a miscommunication, or other such occurrence that could affect a strategy's implementation in real life. By incorporating this error in a simulation involving selective pressures, strategies would be forced to reduce unnecessary complexity or redundancies. Abreu and Rubinstein [1] explored an analytical method for quantifying and optimizing the complexity of automata in decision-making games, but the random error approach may provide a more versatile quantification and assessment of the necessary complexity.

Finally, it is important to note that the analysis in this project covered only constrained and symmetric bargaining scenarios, where the players had equal bargaining power (time-discount) and equal knowledge of their opponent. In the real world, bargainers may possess a range of different preferences over risk and time, while only certain time preferences and only neutral risk preferences were discussed here. The results of an automaton selection meta-game may look

quite different if the players had unique preferences, and this avenue must be explored to further consolidate the conclusions discussed here.

5.2.2 Applications of the Model

The intent of this project was to develop a clear intuition about the nature of bargaining strategies in the real world. To validate the analysis performed here, the results of the project must therefore be confirmed in real-world bargaining scenarios. To do so, it would be feasible to organize experiments involving real humans placed in a bargaining situation against an entity employing the strategies discussed in this project. Furthermore, these experiments could also simply monitor humans playing against each other in a bargaining game to verify and improve upon the model for the human bargaining mind.

Should this approach be undertaken, the data gathered from these experiments could be used to validate and enrich the conclusions drawn here about the nature of real-world bargaining. Using the methods employed in this project, suitable recommendations on how strategies should be adapted to converge towards the optimal bargaining outcomes. Consequently, the dissemination of these conclusions could begin to affect how people bargain and improve the outcomes of bargaining encounters in the real world.

Long-term, should future work in this area provide fruitful, novel bargaining games could be developed to be analyzed in the classical sense. For example, should there be a more succinct classification of "types" of bargainers and their frequency in the real world, the Rubinstein Bargaining model could be analyzed through Perfect Bayesian Equilibrium, tying this whole pursuit back into the realm of pure Game Theory.

Bibliography

[1] Abreu, D., Rubinstein, A., 1988. "The Structure of Nash Equilibrium in Repeated Games with Finite Automata", *Econometrica*, 56, pp. 1259-1281

[2] Axelrod, R., 1997. *The Complexity of Cooperation*. Princeton University Press

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

[4] Banks, J.S., Sundaram, R.K., 1990. "Repeated Games, Finite Automata, and Complexity", *Games and Economic Behavior*, 2, pp. 97-117.

[5] Bateman, T.S., 1980. "Contingent Concession Strategies in Dyadic Bargaining," *Organizational Behavior and Human Performance,*26, pp. 212-221.

[6] Binmore, K., Piccione, M., Samuelson, L., 1998. "Evolutionary Stability in Alternating Offers Bargaining Games", *Journal of Economic Theory*, 80, pp. 257-291.

[7] Binmore, K., Samuelson, L., 1992. "Evolutionary Stability in Repeated Games Played by Finite Automata," *Journal of Economic Theory,*57, pp. 278-305.

[8] Blickle, T., Thiele, L., 1996. "A Comparison of Selection Schemes Used in Evolutionary Algorithms," *Evolutionary Computation*, 4, pp. 361-394

[9] Boyd, R., Lorberbaum, J.P., 1987. "No pure strategy is evolutionarily stable in the repeated Prisoner's Dilemma game," *Nature*, 327, pp. 58-59.

[10] Cramton, P., 1992. "Strategic Delay in Bargaining with Two-Sided Uncertainty," *Review of Economic Studies*, 59, pp. 205-225.

[11] Dawkins, R., 1976. "The Selfish Gene," Oxford Univ. Press, Oxford

[12] Fiegenbaum, A., Hart, S., Schendel, D., 1996. "Strategic Reference Point Theory," *Strategic Management Journal,* 17, pp.219-235.

[13] Holland, J. H., 1973. "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing,* 2(2), pp. 88-18.

[14] Holland, J.H., Miller, J.H., 1991. "Artificial Adaptive Agents in Economic Theory," *AEA Papers and Proceedings,* 81, pp. 365-370.

[15] Kahneman, D., 2011. "Thinking, Fast and Slow," Farra, Straus, and Girrou, New York

[16] MacMurray, B.K., Lawler, E.J., 1986. "Level-of-aspiration theory and initial stance in bargaining," *Representative Research in Social Psychology,* 16, pp. 35-44.

[17] March, J.G., 1978. "Bounded Rationality, Ambiguity, and the Engineering of Choice," *The Bell Journal of Economics,*9, pp. 587-608.

[18] Nash, J.F., 1950. "The Bargaining Solution," *Econometrica,*18, pp. 155-167.

[19] Neale, M.A., Bazerman, M.H., 1985. "The Effects of Framing and Negotiator Overconfidence on Bargaining Outcomes," *Academy of Management Journal,*28, pp. 34-49.

[20] Osborne, M.J., Rubinstein, A., 1990. "Bargaining and Markets," Academic Press, Inc. San Diego.

[21] Riolo, R.L., 1992. "Survival of the Fittest Bits", *Scientific American,* 267, pp. 114-117.

[22] Rubinstein, A. On the Interpretation of Two Theoretical Models, 1995. In K.J. Arrow(Ed.), R.H. Mnookin(Ed.), L. Ross(Ed.), A. Tversky(Ed.), R.B. Wilson(Ed.), *Barriers to Conflict Resolution* 120-130. New York: W.W. Norton and Company.

[23] Rubinstein, A., 1982. "Perfect Equilibrium in a Bargaining Model", *Econometrica*, 50, pp. 97-112.

[24] Siegel, S., 1953. "Level of Aspiration in Decision Making," *Psychological Review,*64, pp. 253-262.

[25] Simon, H.A., 1978. "On How to Decide What to Do", *The Bell Journal of Economics,* 9, pp. 494-507

[26] Smith, J.M., Price, G.R., 1973. "The Logic of Animal Conflict", *Nature*, 246, pp. 15-18.

[27] Sullivan, B.A., O'Connor, K.M., Burris, E.R., 2006. "Negotiator Confidence: The impact of Self-Efficacy on Tactics and Outcomes," *Journal of Experimental Social Psychology*, 42, pp. 567-581.

[28] Tinsley, C.H., O'Connor, K.M., Sullivan, B.A., 2002. "Tough Guys Finish Last: The Perils of a Distributive Reputation," *Organizational Behavior and Human Decision Processes,*88, pp. 621-642.

[29] Tversky, A., Kahneman, D., 1970. "Propsect Theory: An Analysis of Decision Under Risk," *Econometrica,*48, pp. 263-291.

[30] Young, H.P., 1993. "An Evolutionary Model of Bargaining," *Journal of Economic Theory,*59, pp. 145-168.

Appendices

Appendix A

A.1 Automaton Naming System

Automaton Naming System		
First Component	Initial Offer Size	H→ High M→ Medium Lo→ Low
Second Component	Tactic	HD→ Hawk/Dove Rea→ Reasonableness Con→ Constant Concession T4T→ Tit-For-Tat Pro→ Prospect Theory Y→ Yield Mean→ No Concession 50→ Split-the-Difference
Third Component	If #-Pro-# H→ High Frame M→ Medium Frame L→ Low Frame	Otherwise I→ Increasing Concession Size D→ Decreasing Concession Size
Fourth Component	Concession Rate	L→ Linear E→ Exponential

A.2 Round-Robin Results

Initial Round-Robin Results $\delta = 0.93$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	45.65	1.4	50.31
2	H-Y	44.11	0.5	45.59
3	Lo-Rea-D-E	44.02	1.2	47.78
4	M-Y	43.95	0.5	45.23
5	Lo-Y	43.81	0.5	45.25
6	H-HD-D-E	42.88	1.4	47.33
7	M-Rea-D-E	42.80	1.8	45.58
8	M-HD-D-E	42.74	1.3	47.02
9	Lo-HD-D-E	42.63	1.2	46.29
10	Lo-T4T	42.57	1.6	47.51
11	Lo-50	42.48	1.4	46.74
12	M-50	42.29	1.8	47.97
13	H-50	41.83	2.3	48.99
14	Lo-Mean	41.03	2.5	47.83
15	Lo-HD-I-L	40.85	2.7	48.75
16	Lo-HD-I-E	40.83	2.7	48.88
17	M-T4T	40.76	3.1	49.80
18	Lo-Rea-I-L	40.75	2.9	49.11
19	Lo-Rea-I-E	40.66	2.9	49.13
20	Lo-HD-D-L	39.83	3.4	49.32
21	M-Con	39.72	3.5	50.08
22	Lo-Pro-L	39.54	3.5	48.12
23	Lo-Rea-D-L	39.32	3.8	49.42
24	M-Mean	39.05	4.4	49.28
25	Lo-Pro-H	38.99	4.1	48.74
26	Lo-Pro-M	38.96	4.1	48.62
27	M-HD-I-L	38.05	4.5	51.00
28	M-HD-I-E	37.85	4.6	51.10
29	M-Rea-I-L	37.53	4.9	51.47
30	M-Rea-I-E	37.06	5.2	51.62
31	SPE	36.77	8.8	46.47
32	H-Mean	36.12	7.1	51.10
33	H-HD-I-L	35.05	6.1	52.28
34	H-HD-I-E	34.92	6.3	52.77
35	M-Pro-L	34.89	7.7	50.88
36	H-Rea-I-L	34.72	6.5	52.81
37	H-T4T	34.49	9.5	47.67
38	M-HD-D-L	34.34	7.2	52.23
39	H-Rea-I-E	34.03	7.0	53.11
40	M-Rea-D-L	33.61	8.1	52.48
41	M-Pro-M	33.52	9.6	50.09
42	M-Pro-H	33.15	9.9	49.94
43	H-HD-D-L	28.85	11.5	54.42
44	H-Pro-L	28.56	14.1	52.86
45	H-Rea-D-L	27.91	12.9	54.58
46	H-Pro-M	27.65	15.1	53.18
47	H-Pro-H	26.98	15.5	53.32

Initial Round-Robin Results $\delta = 0.95$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	46.88	1.4	50.25
2	H-Y	44.53	0.5	45.59
3	M-Y	44.37	0.5	45.43
4	Lo-Y	44.22	0.5	45.25
5	H-HD-D-E	44.05	1.4	47.27
6	M-Rea-D-E	43.87	1.9	48.25
7	M-HD-D-E	43.8	1.4	46.92
8	Lo-Rea-D-E	43.40	1.8	47.44
9	Lo-HD-D-E	43.33	1.2	46.05
10	M-50	43.25	2.0	47.77
11	Lo-50	43.14	1.5	46.35
12	H-50	42.99	2.5	48.63
13	Lo-T4T	42.34	2.5	47.60
14	Lo-Mean	41.41	3.1	47.48
15	Lo-HD-I-L	40.75	3.8	48.62
16	Lo-HD-I-E	40.64	3.9	48.60
17	Lo-Rea-I-L	40.43	4.1	48.80
18	Lo-Rea-I-E	40.21	4.4	48.91
19	M-T4T	40.20	4.9	48.86
20	M-Con	40.18	4.8	50.45
21	Lo-Pro-L	39.77	4.7	48.13
22	M-Mean	39.56	5.7	49.57
23	Lo-HD-D-L	39.47	5.0	49.10
24	Lo-Pro-M	39.16	5.4	48.47
25	Lo-Rea-D-L	38.89	5.5	49.24
26	Lo-Pro-H	38.30	6.5	48.16
27	M-HD-I-L	38.02	6.4	51.18
28	M-HD-I-E	37.92	6.6	51.42
29	H-Mean	37.62	8.7	51.07
30	M-Re-I-L	37.28	7.0	51.41
31	M-Rea-I-E	37.12	7.3	51.51
32	H-HD-I-L	36.40	8.1	52.86
33	H-HD-I-E	35.96	8.5	53.13
34	SPE	35.91	11.4	45.87
35	H-Rea-I-L	35.58	8.9	53.35
36	M-Pro-L	35.57	9.8	51.28
37	H-Rea-I-E	35.24	9.4	53.53
38	H-T4T	34.78	11.3	47.69
39	M-HD-D-L	34.28	9.9	52.07
40	M-Pro-M	34.04	12.1	50.34
41	M-Rea-D-L	33.41	11.2	51.85
42	M-Pro-H	33.4	12.7	50.52
43	H-Pro-L	30.91	16.1	53.51
44	H-HD-DL	30.26	14.5	54.35
45	H-Pro-M	29.13	18.4	52.75
46	H-Rea-D-L	29.07	16.6	53.66
47	H-Pro-H	27.8	20.2	52.48

Initial Round-Robin Results $\delta = 0.97$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	45.38	2.8	49.09
2	H-HD-D-E	45.15	2.0	47.91
3	H-Y	44.95	0.5	45.59
4	M-HD-D-E	44.93	1.7	47.20
5	H-50	44.82	2.7	48.46
6	M-Rea-D-E	44.80	2.6	48.09
7	M-Y	44.79	0.5	45.43
8	M-50	44.71	2.1	47.56
9	Lo-Y	44.64	0.5	45.26
10	Lo-HD-D-E	44.39	1.2	46.03
11	Lo-50	44.14	1.9	46.60
12	Lo-Rea-D-E	43.47	3.4	46.87
13	Lo-Mean	42.73	4.0	47.45
14	Lo-T4T	42.40	4.3	47.61
15	Lo-Pro-L	41.68	5.4	47.98
16	Lo-HD-I-L	41.42	5.8	48.51
17	M-Con	41.36	6.7	50.00
18	Lo-HD-I-E	41.33	5.9	48.55
19	M-Mean	41.24	7.3	49.56
20	Lo-Rea-I-L	41.09	6.3	48.69
21	Lo-Pro-M	40.85	6.9	48.45
22	Lo-Rea-I-E	40.78	6.9	48.81
23	Lo-HD-D-L	40.48	7.1	48.88
24	H-Mean	39.99	10.8	51.04
25	Lo-Rea-D-L	39.92	7.9	49.05
26	M-T4T	39.78	8.5	48.53
27	Lo-Pro-H	39.74	8.4	48.15
28	M-HD-I-L	39.08	9.7	51.25
29	M-HD-I-E	38.89	9.9	51.32
30	M-Pro-L	38.64	11.1	50.90
31	H-HD-I-L	38.51	11.9	53.54
32	M-Rea-I-L	38.43	10.6	51.44
33	H-HD-I-E	38.35	12.2	53.68
34	M-Rea-I-E	38.18	11.0	51.05
35	H-Rea-I-L	37.83	12.9	53.77
36	M-Pro-M	37.44	13.4	51.43
37	H-Rea-I-E	37.42	13.5	53.40
38	H-T4T	37.10	12.9	48.57
39	M-HD-D-L	36.30	13.5	51.95
40	SPE	36.23	14.7	45.72
41	M-Pro-H	35.82	15.8	50.70
42	M-Rea-D-L	35.25	15.3	51.69
43	H-Pro-L	35.04	18.2	52.52
44	H-Pro-M	33.86	20.5	51.41
45	H-HD-D-L	33.32	19.3	54.47
46	H-Rea-D-L	31.97	21.7	53.75
47	H-Pro-H	30.34	25.7	48.76

A.3 Tournament Re-Run

Thinned Round-Robin Results $\delta = 0.93$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	45.30	1.4	50.00
2	H-Y	45.16	0.5	46.73
3	Lo-Rea-D-E	44.88	1.1	48.46
4	M-Rea-D-E	44.79	1.5	49.69
5	M-50	44.33	1.6	49.68
6	Lo-T4T	44.24	1.2	48.25
7	Lo-Mean	44.22	1.5	49.04
8	M-HD-D-E	44.22	1.3	48.55
9	Lo-50	44.17	1.2	47.99
10	Lo-HD-D-E	44.16	1.1	47.70
11	Lo-HD-I-L	43.94	1.7	49.47
12	M-T4T	43.89	2.1	50.61
13	Lo-HD-D-L	43.80	1.9	49.72
14	Lo-Pro-L	43.74	1.7	48.72
15	Lo-Rea-D-L	43.68	2.0	49.76
16	Lo-Pro-M	43.67	2.0	49.47
17	Lo-Pro-H	43.65	2.0	49.48
18	H-50	43.53	2.1	50.45
19	M-Mean	43.19	2.6	50.60
20	SPE	43.09	3.4	49.39
21	M-HD-I-L	41.99	3.4	52.56
22	M-Pro-L	40.78	4.2	51.59
23	M-HD-D-L	39.71	4.8	53.23
24	M-Pro-M	39.55	6.0	50.84
25	H-Mean	39.07	5.7	52.47
26	H-HD-I-L	37.89	5.3	53.61

Thinned Round-Robin Results $\delta = 0.95$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	46.56	1.4	49.98
2	M-Rea-D-E	45.82	1.5	49.46
3	H-Y	45.61	0.5	46.73
4	M-50	45.30	1.7	49.37
5	M-HD-D-E	45.21	1.3	48.41
6	Lo-Rea-D-E	45.15	1.3	48.21
7	Lo-50	44.88	1.2	47.73
8	H-50	44.83	2.3	50.22
9	Lo-HD-D-E	44.79	1.1	47.44
10	Lo-T4T	44.79	1.6	48.45
11	Lo-Mean	44.75	1.8	48.78
12	Lo-HD-I-L	44.28	2.3	49.48
13	Lo-Pro-L	44.13	2.3	49.07
14	Lo-Pro-M	44.02	2.6	49.31
15	Lo-HD-D-L	43.86	2.7	49.61
16	Lo-Pro-H	43.75	2.8	49.48
17	Lo-Rea-D-L	43.68	2.8	49.68
18	M-Mean	43.56	3.6	51.18
19	M-T4T	43.35	3.9	49.38
20	SPE	42.47	5.4	48.30
21	M-HD-I-L	41.73	5.0	52.71
22	M-Pro-L	41.38	5.6	52.29
23	H-Mean	40.49	7.0	52.43
24	M-Pro-M	39.83	7.6	51.35
25	H-HD-I-L	39.24	6.9	54.12
26	M-HD-D-L	39.10	6.9	52.99

Thinned Round-Robin Results $\delta = 0.97$				
Rank	Name	Avg. Score	Avg. Time	Avg. Allo.
1	H-Rea-D-E	47.20	2.0	50.04
2	M-Rea-D-E	46.90	1.7	49.27
3	M-50	46.62	1.8	49.20
4	H-50	46.62	2.4	50.10
5	M-HD-D-E	46.30	1.5	48.46
6	Lo-Rea-D-E	46.14	1.6	48.40
7	H-Y	46.06	0.5	46.73
8	Lo-50	45.99	1.5	48.04
9	Lo-HD-D-E	45.82	1.1	47.41
10	Lo-Mean	45.78	2.2	48.66
11	Lo-T4T	45.37	2.4	48.56
12	Lo-Pro-L	45.19	2.9	48.90
13	Lo-Pro-M	45.03	3.4	49.23
14	Lo-HD-I-L	44.86	3.5	49.37
15	M-Mean	44.83	4.8	51.08
16	Lo-Pro-H	44.71	3.8	49.36
17	Lo-HD-D-L	44.52	3.9	49.50
18	Lo-Rea-D-L	44.28	4.2	49.59
19	M-Pro-L	43.45	6.5	51.88
20	H-Mean	42.81	8.8	52.45
21	M-Pro-M	42.72	7.9	52.33
22	M-T4T	42.63	7.2	48.88
23	SPE	42.61	7.2	48.08
24	M-HD-I-L	42.43	7.7	52.62
25	H-HD-I-L	41.44	10.3	55.10
26	M-HD-D-L	40.45	10.1	52.91

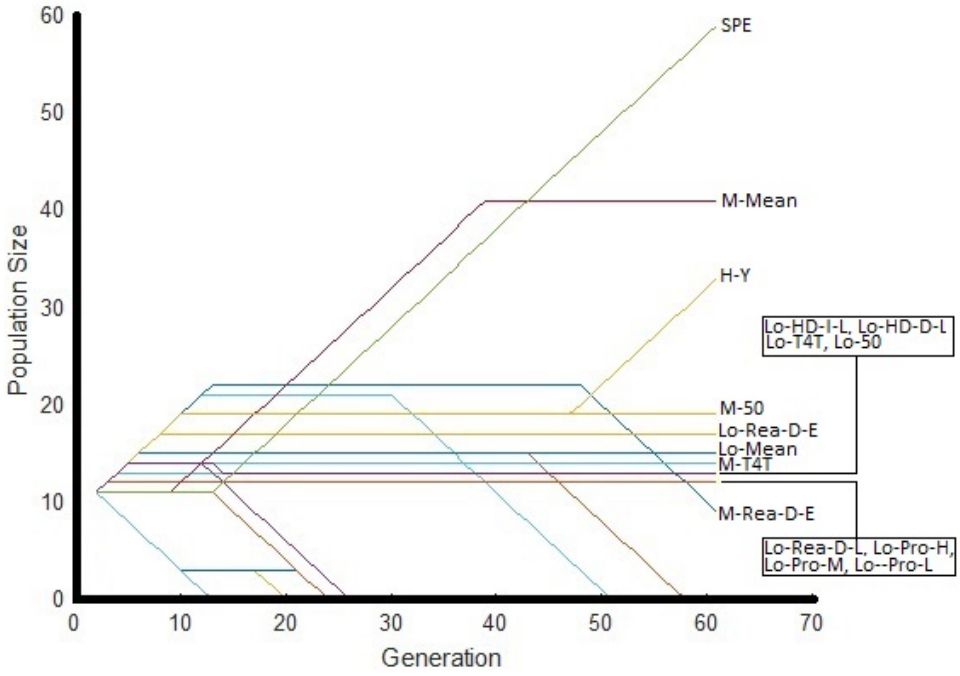

Figure A.1: Standard Deviation Algorithm Simulation $\delta = 0.93$

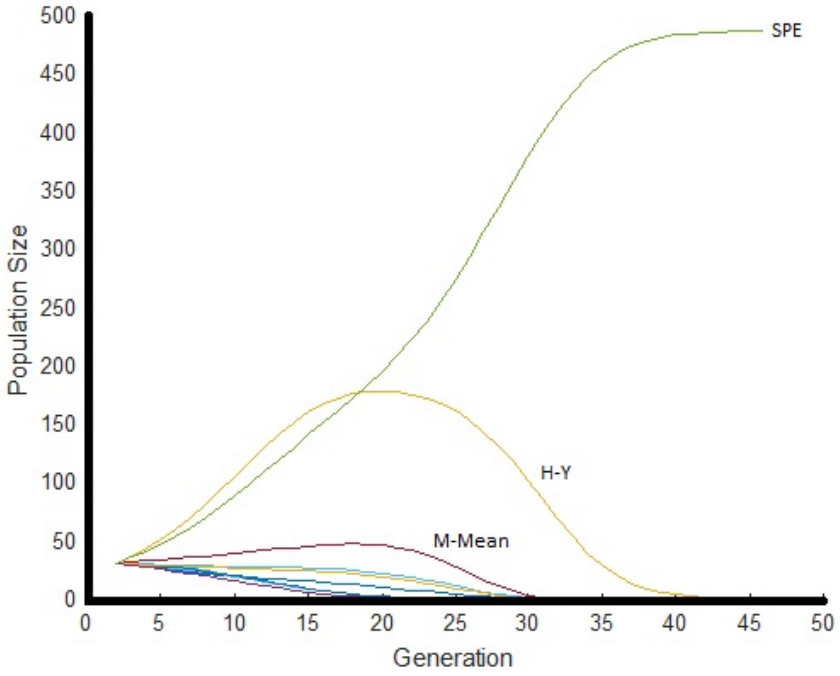

Figure A.2: Filtered Ecological Simulation $\delta = 0.93$

A.4 Ecological Tournament Results

Ecological Simulation Final Results $\delta = 0.93$			
Rank	Name	Final Population Size	Generation Died Out
1	SPE	487	∞
2	H-Y	0	45
3	M-Mean	0	33
4	M-50	0	32
4	M-T4T	0	32
6	M-Rea-D-E	0	31
7	Lo-Mean	0	25
7	Lo-Pro-H	0	25
7	Lo-Rea-D-L	0	25
7	Lo-HD-D-L	0	25
7	Lo-HD-I-L	0	25
12	Lo-Pro-M	0	24
13	Lo-Rea-D-E	0	23
13	Lo-T4T	0	23
13	Lo-Pro-L	0	23
13	Lo-50	0	23
17	M-HD-D-E	0	58 (Filtering)
18	H-Rea-D-E	0	51 (Filtering)
19	Lo-HD-D-E	0	26 (Filtering)
20	H-50	0	24 (Filtering)
20	M-Pro-L	0	24 (Filtering)
22	M-HD-I-L	0	20 (Filtering)
23	H-Mean	0	13 (Filtering)
23	M-Pro-M	0	13 (Filtering)
23	M-HD-D-L	0	13 (Filtering)
23	H-HD-I-L	0	13 (Filtering)

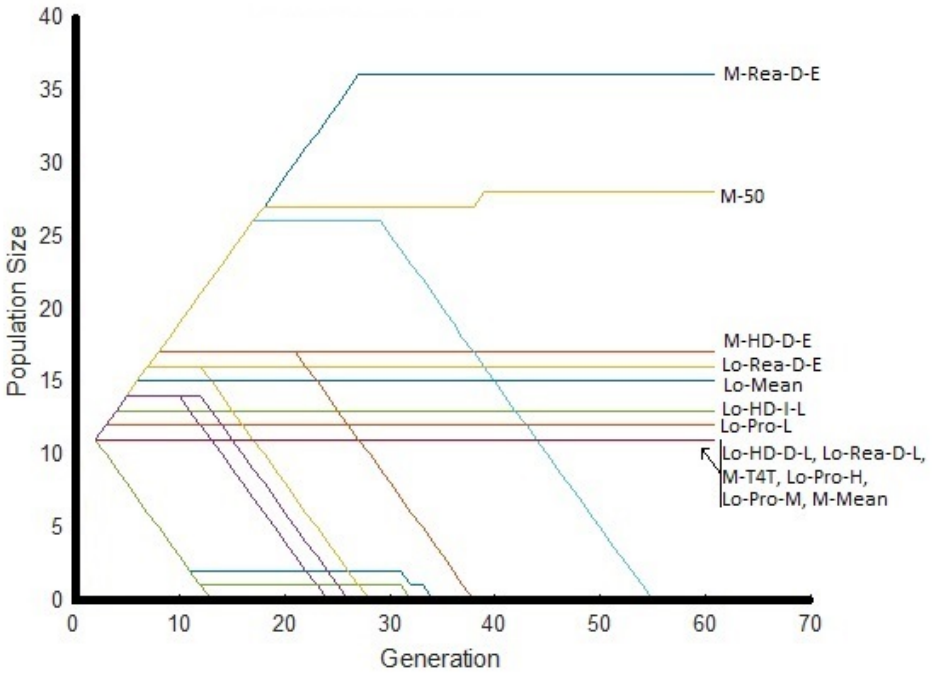

Figure A.3: Standard Deviation Algorithm Simulation $\delta = 0.95$

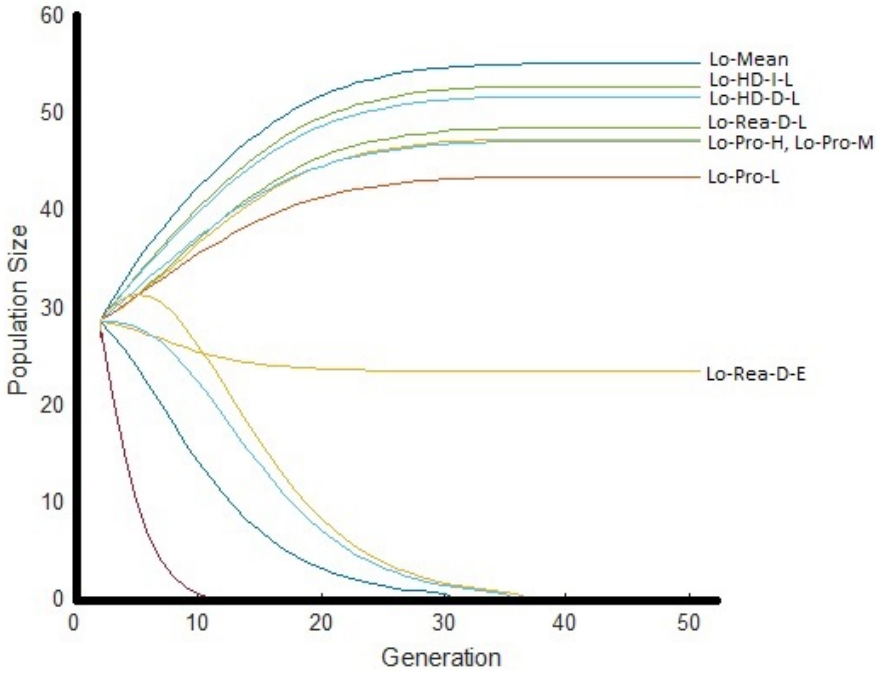

Figure A.4: Filtered Ecological Simulation $\delta = 0.95$

Ecological Simulation Final Results $\delta = 0.95$			
Rank	Name	Final Population Size	Generation Died Out
1	Lo-Mean	55	∞
2	Lo-HD-I-L	53	∞
3	Lo-HD-D-L	52	∞
4	Lo-Rea-D-L	48	∞
5	Lo-Pro-H	47	∞
5	Lo-Pro-M	47	∞
7	Lo-Pro-L	43	∞
8	Lo-Rea-D-E	23	∞
9	M-50	0	37
10	M-T4T	0	36
11	M-Rea-D-E	0	31
12	M-HD-D-E	0	31
13	M-Mean	0	11
14	H-Rea-D-E	0	55 (Filtering)
15	H-50	0	38 (Filtering)
16	M-Pro-L	0	34 (Filtering)
17	M-Pro-M	0	31 (Filtering)
18	H-Y	0	28 (Filtering)
19	Lo-50	0	26 (Filtering)
19	Lo-T4T	0	26 (Filtering)
21	Lo-HD-D-E	0	23 (Filtering)
22	H-HD-I-L	0	13 (Filtering)
22	M-HD-I-L	0	13 (Filtering)
22	M-HD-D-L	0	13 (Filtering)
22	H-Mean	0	13 (Filtering)
22	SPE	0	13 (Filtering)

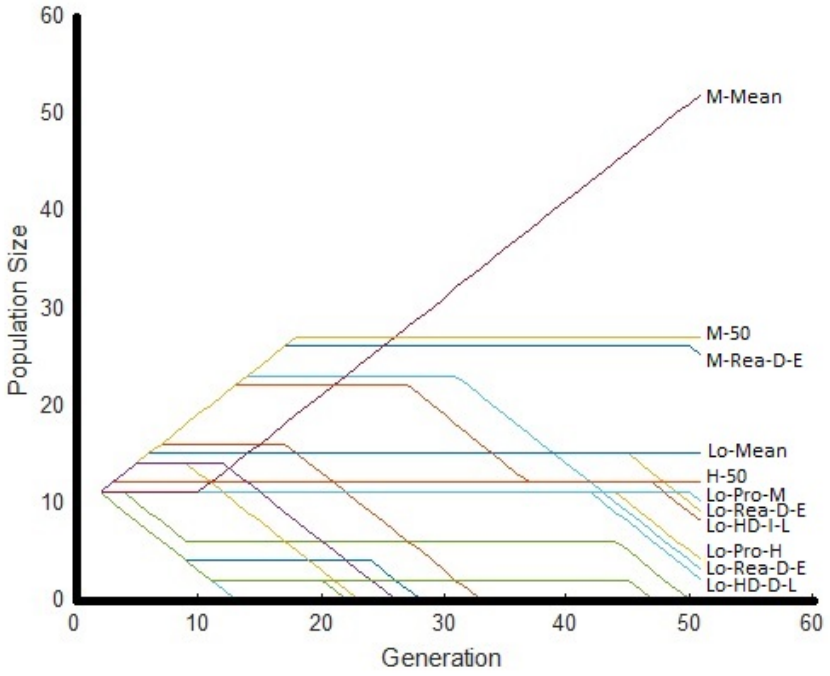

Figure A.5: Standard Deviation Algorithm Simulation $\delta = 0.97$

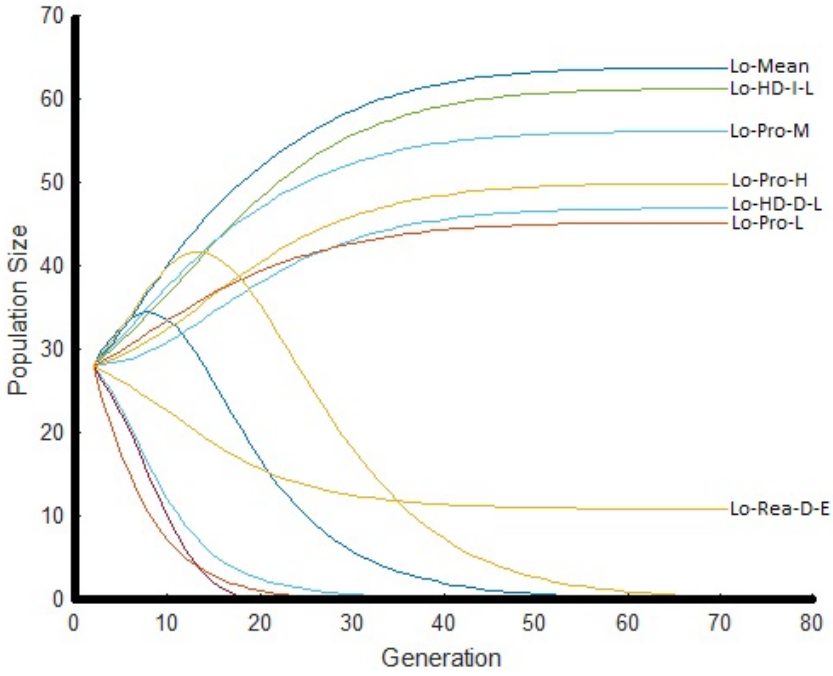

Figure A.6: Filtered Ecological Simulation $\delta = 0.97$

Ecological Simulation Final Results $\delta = 0.97$			
Rank	Name	Final Population Size	Generation Died Out
1	Lo-Mean	64	∞
2	Lo-HD-I-L	61	∞
3	Lo-Pro-M	56	∞
4	Lo-Pro-H	50	∞
5	Lo-HD-D-L	47	∞
6	Lo-Pro-L	45	∞
7	Lo-Rea-D-E	11	∞
8	M-50	0	66
9	M-Rea-D-E	0	53
10	H-Rea-D-E	0	32
11	H-50	0	24
12	M-Mean	0	18
13	Lo-Rea-D-L	0	50 (Filtering)
14	SPE	0	47 (Filtering)
15	M-HD-D-E	0	33 (Filtering)
16	M-Pro-L	0	28 (Filtering)
17	Lo-50	0	26 (Filtering)
18	H-Y	0	23(Filtering)
18	Lo-T4T	0	23(Filtering)
18	Lo-HD-D-E	0	23(Filtering)
21	M-Pro-M	0	22 (Filtering)
22	H-Mean	0	13 (Filtering)
22	M-T4T	0	13 (Filtering)
22	M-HD-D-L	0	13 (Filtering)
22	M-HD-I-L	0	13 (Filtering)
22	H-HD-I-L	0	13 (Filtering)

Appendix B

B.1 Automaton Code

```
function[offer,acceptance]=Master_Automaton(automaton,ledger,k,d)
%automaton represents the I.D. number of the automaton receiving an offer
%ledger provides the history of offers in this game
%k represents the total number of offers made so far in the game
%d is the discount factor
acceptance=0; %default acceptance condition is off
s=0.25; %concession scaler
% i_c_rate=1/5; %old
% d_c_rate=1/8;
% d_adjust=5/8;
i_c_rate=1-d; %concession functions incorporate discount factor
d_c_rate=0.05/d;
d_adjust=min(0.5/d,15/16);
min_con=1/16;
med_con=0.5;
ref_high=50;
ref_med=45;
ref_low=40;
pt_slope_high=0.035;
pt_slope_med=0.0389;
pt_slope_low=0.0437;
pt_trans_high=-1.25;
pt_trans_med=-1.2504;
pt_trans_low=-1.2485;
max_risk=0.75;
log_scale=2;
io_decay=6;
high=60;
medium=55;
low=50;
frame=45;

if automaton==1     %H-HD-I-L
IO=high;
```

```
    if k==0                          %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1           %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession beha
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                         %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end

elseif automaton==2      %H-HD-D-L
IO=high;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
```

```
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end

elseif automaton==3      %M-HD-I-L
IO=medium;

    if k==0                          %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1           %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
```

```
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                        %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==4     %M-HD-D-L
IO=medium;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
```

```
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==5     %Lo-HD-I-L
IO=low;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                        %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
```

```
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==6     %Lo-HD-D-L
IO=low;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
```

```
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==7     %H-Rea-I-L
IO=high;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                        %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==8     %H-Rea-D-L
IO=high;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);                %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==9     %M-Rea-I-L
IO=medium;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                        %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==10        %M-Rea-D-L
IO=medium;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);            %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==11        %Lo-Rea-I-L
IO=low;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=ceil(k/2)*i_c_rate*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                        %if player 2
            concession=ceil(k/2)*i_c_rate*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==12        %Lo-Rea-D-L
IO=low;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                    %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);              %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=max(min_con,-(ceil(k/2)*d_c_rate+d_adjust))*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
```

```
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==13        %H-HD-I-E
IO=high;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behavi
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                        %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end

elseif automaton==14        %H-HD-D-E
```

```
IO=high;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end

elseif automaton==15          %M-HD-I-E
IO=medium;

    if k==0                          %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
```

```
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                    %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==16        %M-HD-D-E
IO=medium;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
```

```
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==17        %Lo-HD-I-E
IO=low;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
```

```
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                    %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==18        %Lo-HD-D-E
IO=low;

    if k==0
        if ledger(1,1)==0           %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ceil(k/2)>1
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<concession        %hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ceil(k/2)>1
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<concession
```

```
%hawk-dove
                    concession=concession*(1+s);
                else
                    concession=concession*(1-s);
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==19        %H-Rea-I-E
IO=high;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2))); %define basic conc
            if ledger(1,2)>IO                               %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);            %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                    %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
```

```
            end
        end
        if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
        if offer<frame
            offer=ledger(end,2);
        end
    end
end

elseif automaton==20        %H-Rea-D-E
IO=high;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
```

```
            end
        end
        if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
        if offer<frame
            offer=ledger(end,2);
        end
    end
end

elseif automaton==21        %M-Rea-I-E
IO=medium;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2))); %define basic conc
            if ledger(1,2)>IO                               %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);            %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                        %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
```

```
            end
        end
        if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
        if offer<frame
            offer=ledger(end,2);
        end
    end
end

elseif automaton==22        %M-Rea-D-E
IO=medium;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
```

```
            end
        end
        if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
        if offer<frame
            offer=ledger(end,2);
        end
    end
end

elseif automaton==23        %Lo-Rea-I-E
IO=low;

    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end,1)-(100-ledger(end,2))); %define basic conc
            if ledger(1,2)>IO                               %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end
                    concession=concession*(1+s);            %if they are making smaller concessions, make large
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else                        %if player 2
            concession=(-exp(-ceil(k/2)*i_c_rate)+1)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end
                    concession=concession*(1+s);
```

```
                end
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
            if offer<frame
                offer=ledger(end,2);
            end
        end
    end

elseif automaton==24        %Lo-Rea-D-E
IO=low;

    if k==0
        if ledger(1,1)==0          %mechanics of first offer, whether player 1 or 2
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end,1)-(100-ledger(end,2)));
            if ledger(1,2)>IO                                   %firmness, react to their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,2)-ledger(end-1,2))/ledger(end-1,2)<(ledger(end-1,2)-ledger(end-2,2))/ledger(end-2,2)
                    concession=concession*(1+s);            %if they are making smaller concessions, make larger concessions
                end
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
            if offer<frame
                offer=ledger(end,1);
            end
        else
            concession=exp(-ceil(k/2)*d_c_rate*log_scale)*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if ledger(1,1)>IO                           %firmness, check their IO
                concession=concession*(1-exp(-ceil(k/2)/io_decay)*s);
            end
            if k>4
                if (ledger(end,1)-ledger(end-1,1))/ledger(end-1,1)<(ledger(end-1,1)-ledger(end-2,1))/ledger(end-2,1)
                    concession=concession*(1+s);
```

```
            end
        end
        if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
        if offer<frame
            offer=ledger(end,2);
        end
    end
end
elseif automaton==25        %M-Con
IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=s*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                        %if player 2
            concession=s*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end

elseif automaton==26        %H-T4T
IO=high;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
```

```
        end
    else
        if mod(k,2)==1          %if player 1
            concession=med_con*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if k>2
                concession=ledger(end-1,2)-ledger(end,2);
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                    %if player 2
            concession=med_con*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if k>2
                concession=ledger(end-1,1)-ledger(end,1);
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==27        %M-T4T
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=med_con*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if k>2
                concession=ledger(end-1,2)-ledger(end,2);
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off to
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                    %if player 2
            concession=med_con*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if k>2
                concession=ledger(end-1,1)-ledger(end,1);
            end
```

```
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==28        %Lo-T4T
    IO=low;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1          %if player 1
            concession=med_con*abs(ledger(end,1)-(100-ledger(end,2))); %define basic concession behaviour
            if k>2
                concession=ledger(end-1,2)-ledger(end,2);
            end
            if 100-ledger(end,2)>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else                        %if player 2
            concession=med_con*abs(ledger(end-1,2)-(100-ledger(end,1)));
            if k>2
                concession=ledger(end-1,1)-ledger(end,1);
            end
            if 100-ledger(end,1)>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==29        %H-Pro-H
    IO=high;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
```

```
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_high
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_high
                concession=pt_slope_high*(100-ledger(end,2))+pt_trans_high;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_high
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_high
                concession=min_con*abs(100-ledger(end,1)-ledger(end,2));
            elseif (100-ledger(end,1))*d^k<=ref_high
                concession=pt_slope_high*(100-ledger(end,1))+pt_trans_high;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_high
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,1)-ledger(end,2));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better of
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==30        %M-Pro-H
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_high
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_high
                concession=pt_slope_high*(100-ledger(end,2))+pt_trans_high;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_high
```

```
            concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
        else
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
        end
        if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it i
            offer=100-ledger(end,2);
            acceptance=1;
        else
            offer=ledger(end,1)-concession;
        end
    else %if player 2
        if (100-ledger(end,1))*d^k<max_risk*ref_high
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
        elseif (100-ledger(end,1))*d^k<=ref_high
            concession=pt_slope_high*(100-ledger(end,1))+pt_trans_high;
        elseif(100-ledger(end,1))*d^k<1/max_risk*ref_high
            concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
        else
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
        end
        if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
    end
end
elseif automaton==31        %Lo-Pro-H
    IO=low;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_high
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_high
                concession=pt_slope_high*(100-ledger(end,2))+pt_trans_high;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_high
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,2);
                acceptance=1;
            else
```

```
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_high
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_high
                concession=pt_slope_high*(100-ledger(end,1))+pt_trans_high;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_high
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better of
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==32          %H-Pro-M
    IO=high;
    if k==0                          %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,2))+pt_trans_med;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_med
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,1))+pt_trans_med;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_med
```

```
            concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
        else
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
        end
        if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it
            offer=100-ledger(end,1);
            acceptance=1;
        else
            offer=ledger(end-1,2)-concession;
        end
    end
end
elseif automaton==33        %M-Pro-M
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,2))+pt_trans_med;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_med
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,1))+pt_trans_med;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_med
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it
                offer=100-ledger(end,1);
                acceptance=1;
            else
```

```
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==34        %Lo-Pro-M
    IO=low;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,2))+pt_trans_med;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_med
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_med
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_med
                concession=pt_slope_med*(100-ledger(end,1))+pt_trans_med;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_med
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better of
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==35 %H-Pro-L
    IO=high;
    if k==0                         %initial offer check
        if ledger(1,1)==0
```

```
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,2))+pt_trans_low;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,1))+pt_trans_low;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==36        %M-Pro-L
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
```

```
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,2))+pt_trans_low;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it is better off
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,1))+pt_trans_low;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it is better of
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==37        %Lo-ProL
    IO=low;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            if (100-ledger(end,2))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,2))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,2))+pt_trans_low;
            elseif(100-ledger(end,2))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,2))/20-2.5)+1.5;
            else
```

```
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d %if not first offer, player checks whther it i
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            if (100-ledger(end,1))*d^k<max_risk*ref_low
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            elseif (100-ledger(end,1))*d^k<=ref_low
                concession=pt_slope_low*(100-ledger(end,1))+pt_trans_low;
            elseif(100-ledger(end,1))*d^k<1/max_risk*ref_low
                concession=-exp((100-ledger(end,1))/20-2.5)+1.5;
            else
                concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            end
            if 100-ledger(end,1)+1>=(ledger(end-1,2)-concession)*d %if not first offer, player checks whther it
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end-1,2)-concession;
            end
        end
    end
elseif automaton==38        %H-Y
    IO=high;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        else
            offer=100-ledger(1,1);
            acceptance=1;
        end
    else
        if mod(k,2)==1
            offer=100-ledger(end,2);
            acceptance=1;
        else
            offer=100-ledger(end,1);
            acceptance=1;
        end
    end
elseif automaton==39        %M-Y
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        else
            offer=100-ledger(1,1);
            acceptance=1;
        end
    else
```

```
        if mod(k,2)==1
            offer=100-ledger(end,2);
            acceptance=1;
        else
            offer=100-ledger(end,1);
            acceptance=1;
        end
    end
elseif automaton==40          %Lo-Y
    IO=low;
    if k==0                           %initial offer check
        if ledger(1,1)==0
            offer=IO;
        else
            offer=100-ledger(1,1);
            acceptance=1;
        end
    else
        if mod(k,2)==1
            offer=100-ledger(end,2);
            acceptance=1;
        else
            offer=100-ledger(end,1);
            acceptance=1;
        end
    end
elseif automaton==41          %H-Mean
    IO=high;
    if k==0                           %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=min_con*abs(100-ledger(end,1)-ledger(end,2));
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
```

```
        end
    end
elseif automaton==42          %M-Mean
    IO=medium;
    if k==0                           %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=min_con*abs(100-ledger(end,1)-ledger(end,2));
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
        end
    end
elseif automaton==43          %Lo-Mean
    IO=low;
    if k==0                           %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=min_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=min_con*abs(100-ledger(end,1)-ledger(end,2));
```

```
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
        end
    end
elseif automaton==44        %H-50
    IO=high;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=med_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=med_con*abs(100-ledger(end,1)-ledger(end,2));
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
        end
    end
elseif automaton==45        %M-50
    IO=medium;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=med_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
```

```
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=med_con*abs(100-ledger(end,1)-ledger(end,2));
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
        end
    end
elseif automaton==46        %Lo-50
    IO=low;
    if k==0                         %initial offer check
        if ledger(1,1)==0
            offer=IO;
        elseif 100-ledger(1,1)>IO
            offer=100-ledger(1,1);
            acceptance=1;
        else
            offer=IO;
        end
    else
        if mod(k,2)==1 %if player 1
            concession=med_con*abs(100-ledger(end,2)-ledger(end,1));
            if 100-ledger(end,2)+1>=(ledger(end,1)-concession)*d
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=ledger(end,1)-concession;
            end
        else %if player 2
            concession=med_con*abs(100-ledger(end,1)-ledger(end,2));
            if 100-ledger(end,1)+1>=(ledger(end,2)-concession)*d
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=ledger(end,2)-concession;
            end
        end
    end
elseif automaton==47        %SPE
    if k==0
        if ledger(1,1)==0
            only_offer=1/(1+d)*100;
            offer=only_offer;
        else
            only_offer=d/(1+d)*100;
            if 100-ledger(1,1)+1>=only_offer
                offer=100-ledger(1,1);
                acceptance=1;
```

```
            else
                offer=only_offer;
            end
        end
    else
        if mod(k,2)==1
            only_offer=1/(1+d)*100;
            if 100-ledger(end,2)+1>=only_offer
                offer=100-ledger(end,2);
                acceptance=1;
            else
                offer=only_offer;
            end
        else
            only_offer=d/(1+d)*100;
            if 100-ledger(end,1)+1>=only_offer
                offer=100-ledger(end,1);
                acceptance=1;
            else
                offer=only_offer;
            end
        end
    end


end

end
```

B.2 Round Robin Tournament Code

```
n=47;
%n represents the number of automata to be involved
% p=1/10;
%p represents the probability of breakdown after each rejected offer.
% Risk of breakdown is currently commented out for round robin
k=0;
%initialize count variable
d=0.95;
%discount factor, arbitrary right now
cutoff=64;
%limits the maximum number of rounds to prevent infinite loops with
%insignificant outcomes
results=zeros(n,6,n);
%results contains the total utility accumulated by each automaton in each
%matchup, and how many turns it lasted
totals=zeros(n,3);
%initialize an array that tracks the total utility gained by each automaton
%,number of turns taken to get there, discounted total
ledger=zeros(1,3);
%initialize ledger

for i=1:n
```

```
    for j=1:n
     [initiali,y]=Master_Automaton(i,ledger,k,d);
     first_offer=[initiali,y];

     ledger=[first_offer(1),0,0];

     [initialj,y]=Master_Automaton(j,ledger,k,d);
     second_offer=[initialj,y];

      ledger=[first_offer(1), second_offer(1), second_offer(2)];

      while ledger(end,3)==0
%           if u>p
              k=k+1;
              if mod(k,2)==1
                   [offer,y]=Master_Automaton(i,ledger,k,d);
                   newoffer=[offer,y];
                   ledger=[ledger;newoffer(1),100-newoffer(1),newoffer(2)];

              else
                   [offer,y]=Master_Automaton(j,ledger,k,d);
                   newoffer=[offer,y];
                   ledger(end,2)=newoffer(1);
                   ledger(end,3)=newoffer(2);

              end
%               u=rand;
%           else
%               ledger=[ledger;0,0,1];
%           end
                if k>=cutoff
                     ledger=[0,0,1];
                end
      end
      results(j,1,i)=ledger(end,1);
      results(j,2,i)=k;
      results(j,3,i)=results(j,1,i)*d^results(j,2,i);
      %(end,1) because this for loop creates each matchup where the
      %principal automaton is player 1. Will need to run again with him as
      %player 2
      ledger=zeros(1,3);
      %wipes ledger after each game
      k=0;
      %reset count
    end
%     totals(i)=[sum(results(:,1,i)),sum(results(:,2,i))];
    totals(i,1)=sum(results(:,1,i));
    totals(i,2)=sum(results(:,2,i));
    totals(i,3)=sum(results(:,3,i));
    %sums results for each automaton from each matchup it had
end

for i=1:n
```

```
    for j=1:n
     [initialj,y]=Master_Automaton(j,ledger,k,d);
     first_offer=[initialj,y];

     ledger=[first_offer(1),0,0];

     [initiali,y]=Master_Automaton(i,ledger,k,d);
     second_offer=[initiali,y];

      ledger=[first_offer(1), second_offer(1), second_offer(2)];
      %initialize ledger for this matchup
      %column three represents the acceptance condition, 0 is not accpeted,
      %1 is accepted offer

      while ledger(end,3)==0
%           if u>p
              k=k+1;
              if mod(k,2)==1
                  [offer,y]=Master_Automaton(j,ledger,k,d);
                  newoffer=[offer,y];
                  ledger=[ledger;newoffer(1),100-newoffer(1),newoffer(2)];

              else
                  [offer,y]=Master_Automaton(i,ledger,k,d);
                  newoffer=[offer,y];
                  ledger(end,2)=newoffer(1);
                  ledger(end,3)=newoffer(2);

              end
%               u=rand;
%           else
%               ledger=[ledger;0,0,1];
%           end
                  if k>=cutoff
                      ledger=[0,0,1];
                  end
      end
      results(j,4,i)=ledger(end,2);
      results(j,5,i)=k;
      results(j,6,i)=results(j,4,i)*d^results(j,5,i);

      ledger=zeros(1,3);
      k=0;
    end
    totals(i,1)=totals(i,1)+sum(results(:,4,i));
    totals(i,2)=totals(i,2)+sum(results(:,5,i));
    totals(i,3)=totals(i,3)+sum(results(:,6,i));

end

average=totals/(2*n)
```

B.3 Ecological Tournament Algorithm

```
function[n,results]=Arena(run)
n=47;
k=0;
d=0.95;
cutoff=64;
results=zeros(n,6,n);
totals=zeros(n,3);
ledger=zeros(1,3);
if run==1
for i=1:n

    for j=1:n
     [initiali,y]=Master_Automaton(i,ledger,k,d);
     first_offer=[initiali,y];

     ledger=[first_offer(1),0,0];

     [initialj,y]=Master_Automaton(j,ledger,k,d);
     second_offer=[initialj,y];

      ledger=[first_offer(1), second_offer(1), second_offer(2)];

      while ledger(end,3)==0
               k=k+1;
               if mod(k,2)==1
                    [offer,y]=Master_Automaton(i,ledger,k,d);
                    newoffer=[offer,y];
                    ledger=[ledger;newoffer(1),100-newoffer(1),newoffer(2)];

               else
                    [offer,y]=Master_Automaton(j,ledger,k,d);
                    newoffer=[offer,y];
                    ledger(end,2)=newoffer(1);
                    ledger(end,3)=newoffer(2);

               end
                 if k>=cutoff
                      ledger=[0,0,1];
                 end
      end
      results(j,1,i)=ledger(end,1);
      results(j,2,i)=k;
      results(j,3,i)=results(j,1,i)*d^results(j,2,i);
      ledger=zeros(1,3);
      k=0;
    end
    totals(i,1)=sum(results(:,1,i));
    totals(i,2)=sum(results(:,2,i));
    totals(i,3)=sum(results(:,3,i));
end

for i=1:n
    for j=1:n
```

```
        [initialj ,y]=Master_Automaton(j,ledger,k,d);
        first_offer=[initialj ,y];

        ledger=[first_offer(1),0,0];

        [initiali ,y]=Master_Automaton(i,ledger,k,d);
        second_offer=[initiali ,y];

         ledger=[first_offer(1), second_offer(1), second_offer(2)];

         while ledger(end,3)==0
                  k=k+1;
                  if mod(k,2)==1
                       [offer ,y]=Master_Automaton(j,ledger,k,d);
                       newoffer=[offer ,y];
                       ledger=[ledger;newoffer(1),100-newoffer(1),newoffer(2)];

                  else
                       [offer ,y]=Master_Automaton(i,ledger,k,d);
                       newoffer=[offer ,y];
                       ledger(end,2)=newoffer(1);
                       ledger(end,3)=newoffer(2);

                  end
                    if k>=cutoff
                         ledger=[0,0,1];
                    end
         end
         results(j,4,i)=ledger(end,2);
         results(j,5,i)=k;
         results(j,6,i)=results(j,4,i)*d^results(j,5,i);

         ledger=zeros(1,3);
         k=0;
      end
      totals(i,1)=totals(i,1)+sum(results(:,4,i));
      totals(i,2)=totals(i,2)+sum(results(:,5,i));
      totals(i,3)=totals(i,3)+sum(results(:,6,i));

      end
end
end

[n, results]=Arena(1);
g=300; %number of generations
i_p=10; %initial population size for each automaton
delta=0.2; %fraction of population change
population=zeros(n,g); %record of populationm numbers through entire history
population(:,1)=i_p; %initial system population
weight=zeros(n,g); %weight of results from each matchup
f=zeros(n,g); %total fitness of each automaton
w_f=zeros(n,g); %weighted average fitness
f_bar=zeros(1,g); %average fitness across every generation
min_fit=zeros(1,g); %calculate miminum fitness at each generation
```

```
for i=1:g
    for j=1:n
        weight(j,i)=population(j,i)/sum(population(:,i));
    end
    for j=1:n
        for k=1:n
            if population(j,i)>0
                f(j,i)=f(j,i)+weight(k,i)*(results(k,3,j)+results(k,6,j))/2;
            end
        end
        w_f(j,i)=weight(j,i)*f(j,i);
    end
    f_bar(i)=sum(w_f(:,i));
    for j=1:n
        % proportional population algorithm
        population(j,i+1)=max(0,n*i_p*w_f(j,i)/sum(w_f(:,i)));
        if population(j,i+1)<0.5
           population(j,i+1)=0;
        end
        % std algorithm
%         if f(j,i)>= f_bar(i)+std(f(:,i)~=0)
%             population(j,i+1)=max(0,population(j,i)+1);
%         elseif f(j,i)<=f_bar(i)-std(f(:,i)~=0)
%             population(j,i+1)=max(0,population(j,i)-1);
%         else
%             population(j,i+1)=max(0,population(j,i));
%         end
        % delta fraction algorithm
%         population(j,i+1)=max(0,population(j,i)+delta*(f(j,i)-f_bar(i))*population(j,i));
%         if population(j,i+1)<0.5
%             population(j,i+1)=0;
%         end
    end
    min_fit(i)=60;
    for j=1:n
        if population(j,i)>0
            if f(j)<min_fit(i)
                min_fit(i)=f(j);
            end
        end
    end
end

hold on
figure(1)
plot(population')
plot(f_bar,'--')
plot(min_fit,':')
title('Ecological Study')
xlabel('Generation')
ylabel('Population Size')
% legend('Population Size','Average Fitness','Minimum Fitness')
hold off
```