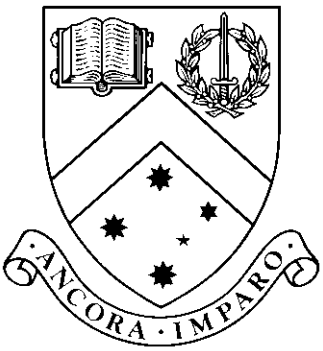
Simulated Evolution of Non-Regular Strategies for Repeated Games

by

Bradon Thomas Hall, BSc, BCompSc



Thesis

Submitted by Bradon Thomas Hall

in partial fulfillment of the Requirements for the Degree of

Bachelor of Computer Science with Honours (1608)

Supervisor: Dr. Julian Garcia

Clayton School of Information Technology
Monash University

November, 2014

Contents

List of Tables

List of Figures

Simulated Evolution of Non-Regular Strategies for Repeated Games

Bradon Thomas Hall, BSc, BCompSc
bthal2@student.monash.edu.au
Monash University, 2014

Supervisor: Dr. Julian Garcia
julian.garcia@monash.edu.au

Abstract

Thesis abstract page

Simulated Evolution of Non-Regular Strategies for Repeated Games

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Bradon Thomas Hall
November 7, 2014

Acknowledgments

I would like to thank my supervisor Julian Garcia, as well as my friends and colleagues for their support over this past year.

Bradon Thomas Hall

Monash University
November 2014

Chapter 1

Introduction

Here I introduce the research area and my contributions to it.

In competitive environments, cooperation often emerges. Most notably in biology, with members of the population (not necessarily the same species) often working together for survival. We are considering cases where there is some benefit to being cooperated with, and some cost to cooperating. Take two predators sharing the food they capture in their separate hunts equally as an example. The predator can choose to share half the spoils of their hunt with their partner (cooperate), or to deceive their partner and keep an uneven share. For simplicity, the deception is either full or not at all. There is a benefit to be had from mutual cooperation; assuming two equally adept hunters, a bad night hunting had by one will often be partially offset by the partners success. That is, risk is reduced, so there is a benefit being cooperated with. On the other hand, there is a cost to cooperating with the partner- part of the food is shared. A risk exists of deception by the partner, depriving the hunter of the benefit.

We have some understanding of cooperation from experience and observation. If an encounter between two individuals is likely to re-occur, establishing cooperative behaviour is likely to pay off in the future. It might be worth taking the risk of being exploited now, to secure cooperation in the future. If reputation exists, so other members of a population can observe how an individual behaves with other members, cooperating could be a good idea so as to develop a reputation that you can be trusted. If success is measured by your genes passing on to a new generation, then it can be beneficial to cooperate with kin so they pass on the genes you share with them.

It is possible to investigate competition and cooperation mathematically- Game Theory could be described quite adequately as the study of competition and cooperation. Consider the case of repeated interactions; for a given scenario, how high does the likelihood of repeating an interaction have to be for cooperating to be the best strategy? In the case of cooperating with kin; how close should the relatives be for individuals to cooperate- should we cooperate fifth cousins, who share only a small number of genes?

The literature into the study of cooperation is broad. Direct reciprocity, where cooperation appears because the chance of encountering the same individual is high enough has been well studied. Assortment, or population structure (such as a likelihood to interact with kin) has also been well studied. More recently, the interplay between reciprocity, structure, and resulting behaviour has been investigated. To study cooperation, a simple representation is used for the individuals that interact. One popular method is to use Finite State Automata. In short, these machines have a collection of states and transitions between states. Based on history of actions, the transitions are followed. How the individual behaves depends on what state it ends up on after following this history.

This representation cannot represent all possible ways of choosing how to interact- all strategies. In this thesis, I will examine the interplay between reciprocity and population

structure, using a representation that examines a wider range of strategies than Finite State Automata. Finite State Automata define a class of strategies called Regular strategies. I will examine some that Fininite State Automata cannot represent: Non-Regular Strategies (specifically, Deterministic Context-Free Strategies).

1.1 Objectives and Research Questions

1.2 Thesis Structure

Chapter 2

Research Context

2.1 Game Theory

A game is a competition between two or more individuals who make choices as to how to interact. Depending on the choices made, each individual gains a payoff. I consider games where two players compete at once in particular, with any number of possible players in the entire population. Each player plays using a strategy, which we might represent explicitly, or we might simply state the payoff that the strategies gain against each other. Game Theory is the study of what strategies will be successful in a given scenario.

The general approach in game theory is to consider a scenario, and simplify it. By doing this we examine the general dynamics of a situation. Each individual playing a game is assumed to try to maximize their personal payoff. For example; let us look circumstances where aggression or non-aggression are possible strategies. Two birds compete for a benefit; they fight for some food. A Hawk strategy escalates the fight. A Dove strategy fights with normal intensity against other Doves, avoiding injury, or retreats in response to escalation. A cost is assigned to escalating a fight (due to injury); call it c. A benefit is assigned to winning a fight, call it b. So, if an individual plays a Hawk strategy against a Hawk strategy, they have a 50/50 chance of winning since they are equivalent. Hawk-Hawk results in an expected payoff of $\frac{b-c}{2}$. When two Doves encounter each other, they don't get injured. They have a 50/50 chance of winning the non-escalated fight, so the expected payoff is $\frac{b}{2}$. A Dove retreats from a Hawk, avoiding injury but also losing any chance of getting the benefit. So, they get a payoff of 0. A Hawk wins by default since the Dove retreats, avoids injury, and gets all the benefit; they recieve a payoff b.

The payoff matrix for this scenario is given in Table 2.1.

	Hawk	**Dove**
Hawk	$\frac{b-c}{2}$, $\boldsymbol{\frac{b-c}{2}}$	b,$\mathbf{0}$
Dove	0,$\boldsymbol{b}$	$\frac{b}{2}$,$\boldsymbol{\frac{b}{2}}$

Table 2.1: Payoff Matrix for the Hawk-Dove game

What is the best strategy to play? Assume that each individual has a strategy they are going to use. In a population of Doves, a lone Hawk will do well in comparison to the rest of the population, gaining the benefit (the food) without paying the cost (of injury). The rest of the population (Doves) gains either $\frac{b}{2}$ or 0.

In a population of Hawks, how well a Dove does depends on the parameters. If the cost of injury is higher than benefit of winning the fight and getting the food, Hawks can be expected to get a negative payoff most of the time– and so the lone Dove, with 0 expected payoff may have the best expected payoff. In a population of 100 Hawks and

1 Dove, where each individual plays another individual randomly, a Hawk is expected to make $\frac{99}{100} \cdot \frac{b-c}{2} + \frac{1}{100} \cdot b$ and a Dove is expected to make 0.

Existing literature in Game Theory allows us to analyse scenarios like these, and answer what is the best strategy to play.

2.1.1 Nash Equilibria

The Nash Equilibria for a game are the strategies from which individuals have no incentive to switch. That is, it cannot get a higher payoff by choosing, on its own, to switch to a new strategy.

A strategy x is a Nash Equilibrium if it cannot increase its payoff playing any other strategy (T) in the set of possible strategies (S) by changing to another strategy (x'):

$$\forall T \neq x, \{T, x\} \in S : \pi(x, T) \geq \pi(x', T) \tag{2.1}$$

Where $\pi(A, B)$ is the payoff for A when it plays against B. If the relation is greater than, it is a Strict Nash equilibrium. If it is greater than or equal to, it is a Weak Nash equilibrium.

2.1.2 Evolutionary Stable Strategies

In an environment where strategies evolve, an extra condition must be added to see what is the best performing strategy.

2.1.3 Replicator Dynamics

2.2 The Prisoner's Dilemma

2.3 Simulating Strategies

2.3.1 Representations

Lookup Tables

One approach to represent strategies playing the prisoner's dilemma is to have a list of possible histories, and how to respond to that history. This is called a Lookup Table. Research proposal Strategy Representations

Finite State Automata

Strategies can also be represented by Finite State Automata. A Finite State Automaton consists of a set of states, a set of transitions from state to state, and each state can be marked as an accepting state. Based on the history between two players in the current interaction, the transitions are followed until the history has been read. If the state the automaton ends up on is an accepting state, the individual represented by that automaton Cooperates, if not, they Defect.

More formally, an FSA consists of:
1. A finite, set of states $S \neq \emptyset$
2. An input alphabet, $\Sigma \neq \emptyset$
3. A transition function $\delta : S \times \Sigma \rightarrow S$
4. A set of final/accepting states $F \subseteq S$
5. The initial state $q_0 \in S$

It is common that the transition function δ is a total function in Prisoner's Dilemma's- for every possible input (of a symbol from the input alphabet Σ) at any state a transition

exists. This is not a strict requirement; if no valid transition exists for an input the input is not in the language, but the FSA is still valid.

FSA with multiple valid transitions for some input are Non-Deterministic FSA. FSA with only one valid transition for any input are Deterministic FSA. For every Non-Deterministic FSA, there is a Deterministic FSA that produces the same result for any valid input, equivalently; the two types of FSA can represent the same classes of languages/strategies (**?**).

Research Proposal Strategy Representations

2.4 Related Work

Literature Review Direct Reciprocity Section **REWORD ALL OF ME**

2.4.1 Axelrod's Tournaments

? investigated the behaviour of strategies using computer tournaments, with repeated Prisoner's Dilemma games played between strategies submitted by people to the tournaments. This study compared fitness (payoff of games) of submitted strategies playing each other. Strategies play each other probabilistically; the game is not a fixed size, instead a continuation probability determines if they continue to play. Out of the 14 submitted strategies, Tit-For-Tat (TFT) performed best. This is a simple strategy that reciprocates the other player's behaviour, after initially cooperating. Initially it cooperates, then it repeats the other player's last move. When playing a strategy that always cooperates (ALLC), it performs identical to ALLC. When playing a strategy that always defects (ALLD), it loses on the first turn, then performs identical to ALLD- given a long enough number of games played, the strategies have similar payoffs.

Axelrod and Hamilton identified three criteria to determine if a strategy can be successful. The strategy must be robust- it must perform fairly well against a wide variety of strategies. TFT does so, including against the a population composed of the simple ALL* examples, but only if there is a sufficient chance of the game repeating. Secondly, the strategy must be stable- a mutant strategy must not be able to invade in the event the strategy becomes established. For example, ALLC is not stable against ALLD. A single ALLD in a population of ALLC will gain better payoffs (fitness) and invade. Lastly, the strategy must be initially viable- it must be able to gain a foothold.

The Tournaments provide insight into the role Direct Reciprocity can play as a mechanism for the success of cooperation. A desirable extension is to allow evolution of the strategies, rather than just allowing the example strategies to compete. From the outcome of the tournaments one might initially suspect Tit-For-Tat might emerge and always dominate. However, **?** showed this might not be the case; Tit-For-Tat is not (and indeed no strategy is) evolutionary stable in the repeated dilemma game (**??**).

2.4.2 Evolving Strategies in Simulations

? used a simple Genetic Algorithm approach to study the evolution of strategies for the Prisoner's Dilemma (**?**). A half-memory approach is one in which the opponent's sequence of Cooperate (C) or Defect (D) moves are remembered. A full memory approach remembers its own moves, or equivalently the results (both C is the same as Reward, R, both D is the same as Punish, P, etc). A string of {C,D} represents each strategy. A Full-Memory approach was used in **?**.

Based on previous moves, a location in that string is looked up, and states the next move to perform. The first 6 bits determines the initial strategy, then there is not a history of 3 moves. It gives a 'history' to base decisions on, when no actual history is available. 6

bits are required since both the opponents and their own moves need to be encoded. To encode the strategy, 64 bits are used. Three moves are remembered, with 4 possibilities- so 4^3. A total of 70 bits were used by Axelrod's simulation, allowing for moves to be determined based on the last 3 results. That is, it is a full-memory simulation of length 3. This allows for any of 2^{70} different strategies to appear- analysis using a matrix of payoff of every strategy against every other strategy would be a square matrix with 2^{70} elements on each side. This is why explicit analysis selects notable examples from the population, and demonstrates why simulation will be useful.

Strategies were evolved using Genetic Algorithms. Strategies are generated in a population, and then they compete with 8 strategies from the original tournaments to determine their fitness. A new population is determined by reproducing from the population, with fitness determining what strategies reproduce. They then compete with the 8 representatives again. Reproduction involves a small chance of random mutation, and a crossover from each parent. The string representing the strategy is the chromosome, and the new child is produced from parts of each parent's chromosome. A population of 20 strategies competed in games 151 moves long, competing with 8 other members of the population in each round, for 50 generations. Technology of the time limited the size of simulations; today we have the luxury of being able to throw far more computing power at the problem, allowing for much longer (and with larger population) simulations. The initial population was random. The strategies that evolved mostly resembled Tit-For-Tat, and Axelrod identified general patterns for strategies. They continued cooperating after mutual cooperation had been established, they responded to opponent defection with defection, they returned to cooperation after it was restored by the opponent following a defection, and when stuck in a pattern of mutual defection they do not attempt to escape and allow themselves to be exploited.

Strategies that performed better than Tit-For-Tat did emerge. These developed ways to exploit one of the 8 representatives, having a higher fitness against that, at a cost of a lower fitness against some of the other 8 representatives. If the net result is a gain over Tit-For-Tat, the strategy performs better. This only applies to the environment of this simulation. In an environment where the competing strategies is not fixed, for example when the competition is picked from the population, those exploited strategies might be expected to die off and remove the advantage the exploiter gains.

Axelrod also performed simulations in which reproduction only involved the chromosome of one parent; reproduction was asexual, involving no crossover, so new strategies only appeared from random mutations. Again, Tit-For-Tat-like strategies evolved. However, strategies that exploited one of the 8 representatives did not appear as often. Reproduction mechanism affects results of this simulation- in this case, crossover explored the strategies differently to only mutating, and as will be discussed further how strategies mutate from one to another is important.

2.4.3 Fogel's Automata

? studied the evolution of strategies using Finite State Automata (FSA). These are abstract computing machines with a number of states, and a number of transitions (**?**). Each state can be labelled an accept state, and one state is labelled the start state. Based on some input sequence, transitions are followed from one state to the next. When the machine has read all input, the final state may be an accept or reject state. So, it can compute whether a sequence meets or does not meet a criterion.

The sequence of {C,D} choices (the history) in a game of Prisoner's Dilemma lends itself to this method; view the strategy as a language defined by a FSA. When a history is a word in that language (when its FSA ends in an accept state) the strategy cooperates, when it is not it defects. Fogel used FSA with a maximum of 8 states, to simplify analysis

of evolved strategies. The strategies were full-memory, knowing both their opponent's history of moves and their own. A population of size 50 to 1000 was used, 151 games were played for each interaction, and there were at most 200 generations. Number of generations and repetitions of experiments were limited by hardware. As in Axelrod, new strategies are created by mutations of parents, and what strategies reproduce are decided by fitness.

Results were similar to those found by (**?**)- strategies developed that would establish mutual cooperation if possible. Fogel suggests initial sequences of symbols form patterns, which other machines can recognise to establish cooperation- a 'handshake'. Interestingly, size of the population was not observed to affect the evolution of cooperative behaviour, nor the time taken for it to evolve. The best machines produced in the 50 population example did have lower fitness compared to best machines from larger populations, though any effect of population size appeared to diminish after sufficient population size was reached.

When payoff for exploiting the opponent is increased, it is expected that mutual cooperation should decrease. For example, if the payoff for alternating Exploiter (payoff T), Exploited (payoff S) is larger than continuous mutual cooperation, this behaviour should be selected for. Indeed, transitions to long-term mutual cooperation is not seen in the structure of the best performing strategies in these circumstances. Fogel identified the impact of the payoffs in the PD matrix; when mutual cooperation results in the highest overall payoff it can be expected to evolve, when the payoff from alternating defection and cooperation exceeds the payoff for cooperating, mutual cooperation does not evolve. When either alternating or mutually cooperating has equal reward, initial population state determines the outcome.

This approach limits the strategies that can evolve to those representable by Automata with 8 nodes. It also uses fixed length games; with an expanded strategy space, cooperation would not be expected with this simulation.

2.4.4 Miller's Automata

? used FSA, of a size of 16 states, to perform Prisoner's Dilemma simulations. Each FSA is represented by a string of bits. To define each state, 9 bits are used. One bit defines what to do if the history input to the machine ends at the state- or, equivalently defines if it is an accept state. Four bits define what state to transition to in response to a cooperation, and the final 4 define what to do in response to a defection (16 states, so 4 bits are required to identify the target state). Four bits at the start of the string define the initial state, followed by 16 sets of 9 bits, one set for each state. Since this is a total length of 148 bits, 2^{148} strategies are possible (ignoring the fact many will be actually equivalent, just different expressions of the same strategy).

Evolution follows a similar process to previously described research. Based on performance in the game (playing every player including itself in a sequence of 150 games), some strategies in the population are selected for reproduction. From a population of 30, 20 are selected from to reproduce. These 20 also survive to the next generation. Reproduction involves a crossover and a mutation process. In crossover, a sequence from both parents is taken and combined. In mutation, a bit is flipped. The 10 individuals that result from crossover are added to the population, replacing the 10 least fit. Initial populations were random.

Simulations were run with two levels of noise included. For both noise levels, the number of states reduced from the initial value. Number of states is calculated from the minimal representation of the FSA- not the actual equivalent FSA that is in the simulation. More noise resulted in fewer states. One interpretation of this result is that establishing a pattern of behaviour between individuals requires a simple 'message' to communicate if the environment is noisy. In the simulation, defection was reciprocated at a high rate,

and cooperation was reciprocated at a lower rate- but still reciprocated. Higher noise also negatively impacted rates of cooperation.

Miller's work provides a clear technique for defining a FSA, a simple mutation method, and various features of FSA that may be worth studying- like the number of states, and number of terminal states (both transitions at this state are to itself). However, in limiting the number of states, the strategy space is limited. Games are also fixed length; if the number of states were unbounded, this simulation would eventually collapse to a population of full defectors (**?**). Probabilistic length games would avoid this issue.

A summary of the works regarding Direct Reciprocity examined in this review is shown in Table **??**. The representations column indicates the means by which individual players are modelled. A strategy describes how the individual player will play the game, in response to a history of previous movements (a sequence of {C,D}, Cooperate and Defect). The representation chosen determines the types of strategies that can evolve, the number of strategies, and the set of strategies they can represent. The importance of the representation will be developed later in this review. In the repeated Prisoner's Dilemma, there are an infinite number of strategies. An individual representation will allow a subset of these strategies- they will have a strategy space. The strategy space is listed, along with how the games are repeated (fixed length, probabilistic length), and number of generations simulated. In the notes column, I indicate that **?** is a study of how strategies perform at the repeated Prisoner's Dilemma- with no exploration of strategies aside from those added initially. I indicate **?** used both repetition of the game, and Assortment of the population.

2.4.5 Direct Reciprocity in Structured Populations

REWORD ALL OF ME **?** investigated the role Assortment can play in circumstances in which Direct Reciprocity does not enable cooperation- when games are one shot. In previously discussed research, the probability of encountering an individual in a population was uniform (the population is well-mixed). **?** instead considers a scenario where the probability of encountering a cooperator, given the individual is a cooperator, is p. The probability of encountering a defector, given the individual is a defector, is q. The population consists of just cooperators and defectors. The likelihood of meeting an alike player is therefore increased (for p, q > 0). The assortivity index is a function of q and p, where x denotes the proportion of the population that are cooperators:

$$a(x) = p(x) - q(x)$$

They also found the difference (D) between the payoff of the cooperator and the payoff of the defector. In this equation, b is the benefit recieved from someone cooperating with you, and c is the cost you pay to cooperate (so R=b-c, S=-c, T=b, P=0 in the PD game matrix).

$$D(x) = a(x)b - c \tag{2.2}$$

They find when the reward for cooperation times the assortivity index is greater than the cost for helping, cooperators will do better than defectors. When it is less than, defectors will do better. This is Hamilton's rule- which indicates whether an individual shares enough genes for helping them to be an increase in fitness for the potential helper- but with an Assortment parameter replacing relatedness.

Consider the payoffs in Table 2.2. Take a simple Assortment scenario, with a single-shot game. With probability r, a player plays with an 'alike' player, without sampling from the population. With probability $1 - r$, the player plays with a player sampled from the population. The population consists of two strategies: ALLC, and ALLD.

	Cooperate	**Defect**
Cooperate	R = 4, **R = 4**	S = 0, **T = 5**
Defect	T = 5, **S = 0**	P = 1, **P = 1**

Table 2.2: Payoff Matrix for the Prisoner's Dilemma, with example values

When sampling from the population (or equivalently, when there is no structure), the payoffs Π are:

$$\Pi_{ALLC} = \frac{N_{ALLC} - 1}{N_{TOTAL} - 1} * (R = 4) + (S = 0)$$

$$\Pi_{ALLD} = \frac{N_{ALLC}}{N_{TOTAL} - 1} * (T = 5) + \frac{N_{ALLD} - 1}{N_{TOTAL} - 1} * (P = 1)$$

Since the Temptation payoff is always greater than the Reward payoff and the Punishment payoff is always greater than the Sucker's payoff, in an unstructured population ALLD has a higher expected payoff.

When structure is added, the expected payoff is:

$$\Pi^{(r)}_{ALLC} = (1 - r)\Pi_{ALLC} + r * (R = 4)$$
$$\Pi^{(r)}_{ALLD} = (1 - r)\Pi_{ALLD} + r * (P = 1)$$

Substituting the example game, with 50 ALLC players and 50 ALLD players:

$$\Pi^{(r)}_{ALLC} = (1 - r)\frac{4 * 49}{99} + r * (4) = \frac{196}{99} + \frac{200}{99}r$$
$$\Pi^{(r)}_{ALLD} = (1 - r)\frac{299}{99} + r * (1) = \frac{299}{99} - \frac{200}{99}r$$

So if $r > 103/400$, ALLC has a higher fitness. That is, in a sufficiently structured population, cooperation can become favorable even in the one shot version.

Assortment is known to enable the success of cooperative behaviour (**?**). Next, I will discuss when games are both repeated, and there is Assortment of the population.

The behaviour that evolves at with various levels of probability of a game continuing and population structure was discussed in **?**. It was investigated in detail with a simple model with both simulation and analysis by **?**. The method I will use in my project is based on this paper. Both general analytic results were found, and simulation results that may be representation-specific (FSA with half-memory were used in the simulations in this paper).

The Prisoner's Dilemma game played was defined as follows:

	Cooperate	**Defect**
Cooperate	R = b-c, **b-c**	-c, **b**
Defect	b, **-c**	0, **0**

Table 2.3: Payoff Matrix for van Veelen et. al.

The analysis focused on identifying behaviour as both continuation probability (so, number of times the game is played) and Assortment are varied. Assortment is the likelihood of meeting an alike player: there is r chance that the other player is the same as the current player, and $1 - r$ chance that they other player is selected at random from the population (**?**). Several regions of predictable behaviour exist in a graph of continuation probability vs assortment. This graph is reproduced in Figure **??**. The graph is for

b/c=2, different ratios will change the curves separating regions- but the results will be qualitatively the same.

FIGURE OF PNAS

In Region I, ALLD is an equilibrium- determined by comparing the payoff of ALLD to a mutant that cooperates at least once. It is not the only equilibrium in the region, but equilibria strategies will play Defect against themselves and ALLD. In this region, cooperation is not expected; no cooperating strategy is an equilibrium.

In Region II, a variety of strategies are equilibria; for example, ALLD and TFT. So, both extremes of behaviour may be seen in a population under these circumstances.

In Region III, there are again a variety of strategies that are equilibria, but ALLD and other complete defectors are no longer equilibria. Cooperative behaviour of varying types and degrees is expected to be common most of the time (equilibria can be escaped, for a short time).

In Region IV, ALLC is an equilibrium. All other equilibria always cooperate when playing against ALLC or themselves. Fully cooperative behaviour is expected to dominate. More regions are defined, but these are the ones of primary concern.

These general regions are expected to hold regardless of representation (I and II are particularly well defined, and not expected to vary). The simulations van Veelen et. al. conducted were of FSA with an unbounded number of states and half-memory (of unbounded length). The simulation is initialized with simple individuals: ALLD. Every member plays one other member of the population for a number of rounds determined stochastically, based on the continuation probability. With probability r a member will play against an identical strategy. With probability $1 - r$ a member will play against another player from the population.

This payoff was used to calculate the probability for the individual to reproduce into the next generation. Stochastically, members of the population are selected for the new generation, so unfit individuals tend to be removed. During reproduction, mutation could also occur, resulting in new strategies.

Results of the simulation were consistent with the analysis. A colourmap indicating the amount of cooperation in the simulation for varied continuation probabilities and assortment parameters was produced, with behaviour matching that predicted in each region. In general, both increased assortment and increased continuation probability increases cooperation (but there are exceptions).

Indirect invasions are observed in the simulation (**?**). Indirect invasions are when a new strategy enters the population not by performing well against the currently dominant strategy, but by 'springboarding' off another strategy. For example, consider that TFT is resistant to ALLD, and will only get exploited on the first move. ALLC is a neutral mutant of TFT (it plays against TFT the same way TFT plays against itself), and in a finite population of mostly TFT, ALLC can become more common by neutral drift. ALLD can exploit ALLC, and so when this neutral drift occurs, ALLD can use it to invade the population. This was summarised as: "unconditional cooperation is therefore cooperation's worst enemy" (**?**). Indirect invasions were observed establishing cooperative behaviour in a defecting population too.

The results of the analysis should be valid regardless of representation. The simulation built upon previous work, investigating the interplay between reciprocity and population structure, and using unbounded rather than bounded FSAs allowing an infinite strategy space. However, the set of all possible strategies for the Prisoner's Dilemma is larger than this space.

Other representations have their own strategy space. The choice of representation will exclude some, and enable some that were not possible in other representations. Representation also affects the chance of a strategy appearing, or the route by which it appears, since the mutation process may differ.

The model described in **?** was chosen as the basis of my project. Both varied assortment and varied chance of a game continuing are modelled. Broad behaviour (such as whether strategies that always cooperate can dominate, or can be somewhat successful) can be predicted for ranges of assortment and continuation probability. This provides bounds for what behaviour is expected. Detailed behaviour can be discovered through simulation. The impact of variations in the simulation- such as changing how strategies are represented- can be compared quantitatively. By using both assortment and continuation probability, conditions not explored in previous literature (Region III for example in Figure **??**) can be investigated.

Chapter 3

Research Method

3.1 Representation

3.1.1 Pushdown Automata

Pushdown Automata can be briefly described as Finite State Automata that also interact with a stack. During each transition instead of simply reading from the history and checking what transitions are valid, the machine (optionally) reads the next point in the history, and what item is on the top of the stack. Valid transitions are chosen based on what is read, and what is on the top of the stack (**?**).

Each transition can push a character to the stack.

More formally, a PDA consists of:

1. A finite, set of states $S \neq \emptyset$
2. An input alphabet, $\Sigma \neq \emptyset$
3. A stack alphabet, Γ
4. The possible states of the stack, Γ'
5. A transition function $\delta : S \times \Sigma \to \mathcal{P}(S, \Gamma')$
6. A set of final/accepting states $F \subseteq S$
7. The initial state $q_0 \in S$
8. The stack marker $Z_0 \in \Sigma$

Initially, the stack contains only the stack marker ($\Gamma'_0 = \{Z_0\}$).

A PDA is Non-Deterministic if the power set $\mathcal{P}(S, \Gamma')$ can have more than one element. It is Deterministic if $\mathcal{P}(S, \Gamma')$ has at most one element. That is, if at most one transition exists for any possible state and stack state. The language or set of strategies representable by a DPDA is a subset of strategies represented by PDAs. The strategies represented by FSAs is a subset of both. FSAs can represent Regular strategies, DPDAs can represent Deterministic Context-Free strategies, and PDAs can represent Context-Free strategies.

FIGURE OF LANGUAGE SPACE

Initially, I pursued simulating Non-Deterministic Pushdown Automata. This is a more complex task than Deterministic Pushdown Automata, due to needing to maintain a set of valid configurations, since any number of valid configurations can exist for a given input. This results in more complicated software to create, and more computation needed to determine if one of the configurations is an accepting configuration.

Third party software was explored for use with this; specifically **?**. This was designed as a learning tool for Theory of Computation, and for my purpose needed to be significantly modified. For example, current possible configurations would need to be maintained while waiting for the next input. The alternative already supported- re-checking if the

entirety of the current history instead of just iterating to the next item in the history- was computationally expensive. I made the decision to instead implement my own version of the simpler Deterministic Pushdown Automata., with the intention of expanding representations at a later stage.

3.1.2 Deterministic Pushdown Automata

3.1.3 Mutations

As discussed previously *chap ref*, the mutation scheme matters. As long as it is a reasonable exploration of the strategy space we expect certain behaviour to hold, but much of the dynamics will be dependant on the mutation scheme chosen. Since I intend to extend work done with Finite State Automata, I have based my mutation scheme heavily on a mutation scheme for Finite State Automata.

The first mutation is the Add State mutation. A state is generated, which can be either an accepting state or a non-accepting state. Additionally, an outbound transitions is added with random Read, Pop, Push instructions and a random destination (with no bias towards 'nearby' states, or bias against the new state itself as the destination). One transition is rerouted with the new state as its destination, ensuring the new state is connected to the Automaton.

Alternative mutation schemes could already be noted for just this first mechanism. The requirement that the new state be connected could be dropped- perhaps with justification to biologic systems. Inactive information may exist in a genome. It could also be important- at some later stage it may be the subject of an advantageous mutation.

The connected requirement is made because after mutations, the Automaton is pruned of disconnected states in order to address bloat *a ref here*. Another alternative would be to not add any transitions to the new state- or to add an inbound transition rather than reroute an existing one. The method chosen is used as it ensures connectedness and is a smaller edit, when we define the size of an edit as the number of parameters changed. Four parameters must be added if a transition is inserted (Read, Pop, Push, Destination), whereas the single re-routing is only one parameter change.

Removing a state is the second method of mutation. Outbound transitions from the state are deleted. Inbound states are randomly rerouted. When the re-routing results in an invalid automaton, the transition is deleted instead. Rerouting is chosen over deleting all related transition to minimize the size of the mutation. If the initial state is removed, another randomly chosen state is assigned as the initial state.

Adding a transition involves randomly selecting a source, destination, and Read Pop and Push instructions. All factors are uniformly selected, with the exception of only allowing a Stack interactions with a stack marker to both pop and push; not one or the other.

The remove transition mutation removes a transition. As a result, this may trigger more substantial changes to the Automaton. For example, it may disconnect the Automaton, making part unreachable and causing it to be pruned.

Each parameter that a Transition contains can also change. It's destination state, read condition, pop condition and push instruction can all be changed. These are selected randomly, with the exception of stack markers (with the pop and push condition as before). Transitions that would make the Automaton Non-Deterministic are simply not added, so the Automaton does not change on this mutation event in that case.

Finally, the accepting/non-accepting parameter of a state can be flipped.

Note that this scheme can explore the entirety of the strategy space that can be represented by Deterministic Pushdown Automata which accept by final state.

3.2 Simulation

3.3 Analysis

The primary output of the simulation is the time series. This includes the total payoff at each reported generation, the strategies present at each generation, and the number of each strategy present. Several factors need to be examined. The first is the average payoff; is the population cooperative when continuity probability and assortment is set to a certain number. What strategies are common with theses parameters? The number of times each strategy is found over an entire can be reported based on this time series, at which point the number of each strategy present when the strategy is common can be examined. Important factors to consider is how it enters the population, and how it exits if it does so.

Chapter 4

Results and Analysis

Chapter 5

Conclusions

Appendix A

Source Code Release

The easiest method to get the simulation source code is to import the archive into eclipse.

The easiest method to run the simulation is to use the jar provided.

Please note that the archive file attached to my thesis is the version submitted for assessment.

The code for my simulation will be released on github at github.org/somewhere. I intend to improve upon this work in the future here, perhaps with others interested in the area of research.

References