# 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 Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.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 9, 2014

# Acknowledgments

I would like to thank my supervisor Julian Garcia, as well as my friends, family and colleagues for their support over this past year. Special thanks to Candace Nazareth and Melissa Hall for their support and helpful critique of work over this year.

<div align="right">Bradon Thomas Hall</div>

*Monash University*
*November 2014*

# Chapter 1

# Introduction

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 getting help, and some cost to giving help. For example, consider two predators sharing the food they capture in their separate hunts equally. 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 getting help. On the other hand, there is a cost to helping 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 (there is incentive to reciprocate). 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 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 that studies 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 (van Veelen et al., 2012). 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. In a repeated game, the history is the series of moves that have occured so far in an interaction between two players. Based on history of actions, the transitions in a FSA are followed. How the individual behaves depends on what state it ends up on after following this history. If it ends on a state that is a final state, it cooperates. If it ends on a state that is not a final state, it defects. An example is shown
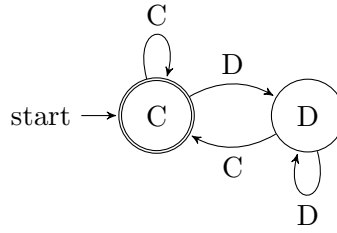
Figure 1.1: Tit-For-Tat as a FSA

in figure **??**. This strategy, called Tit-For-Tat (TFT) starts where the arrow indicates; which is an accepting state (indicated by two circles).

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

I have produced a simulation that can simulate the evolution of strategies in the Prisoner's Dilemma using a previously unexplored representation for those strategies. The results of these simulations are used to answer several questions:

- How does changing the strategy space affect the cooperation that is seen? Results are compared to a previous simulation that used an FSA representation.

- theoretical results hold with the larger strategy space? In certain regions of the graph, a bound can be placed on the behvaiour seen.

- Do successful non-regular strategies evolve?

The research contribution of my work is focused somewhat on the last question. By asking this, I investigate the impact of the assumptions previous work has put on the capability of the strategies studied.

## 1.2  Thesis Structure

I will next discuss Game Theory as general background knowledge. Building from this, I will discuss specific previous work into studying behaviour in the Repeated Prisoner's Dilemma.

# Chapter 2

# Research Context

My thesis is building of work in Game Theory in general, as well as specific work into the Repeated Prisoner's Dilemma. Concepts from work in Game Theory will be discussed, as well as previous work into the simulated evolution of strategies for the repeated Prisoner's Dilemma.

## 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 | $\frac{b-c}{2}$, $\boldsymbol{\frac{b-c}{2}}$ | $b$,**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. If $c > b$, the Dove has the highest expected payoff.

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 $B$. Alternatively stated, a Nash Equilibrium is found when a strategy is the best reply to itself. 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.

In the Hawk-Dove game for $c > b$, analysing the payoff for a Pure strategy (which always plays a stated strategy; in this one-shot version of the game, either Hawk or Dove) shows that neither is Nash. When playing against a Hawk (H), you would do better to switch to Dove and avoid the cost. When playing against a Dove (D), you would do better to switch to Hawk and gain the benefit $b$:

$$\pi(H, H) \ngeq \pi(D, H)$$
$$\pi(D, D) \ngeq \pi(H, D)$$

If instead of playing with a set strategy, we can change to a scenario where two strategies probabilistically select a move. So, $p_1$ plays Hawk with probability $p$ and Dove with probability $p - 1$. Strategies that make decisions probabilistically are Mixed Strategies. In this case, the expected payoff must be calculated based on probabilities of each move. Consider some strategy $p_1$ playing itself:

$$\pi(p_1, p_1) = \pi_{H,H} * p_{H,H} + \pi_{D,D} * p_{D,D} + \pi_{D,H} * p_{D,H} + \pi_{H,D} * p_{H,D} \tag{2.2}$$
$$\pi(p_1, p_1) = \frac{b-c}{2} * p^2 + \frac{b}{2} * (1-p)^2 + 0 * (1-p)p + b * p(1-p)$$

In a similar strategy, we can find the payoffs for all combinations of $p_1$ and $p_2$. In the example of the Hawk-Dove game, a Nash Equilibria does exist for mixed strategies [1]. Pure strategies are the focus of this thesis, but the above serves as an introduction to updating the payoff matrix for a new scenario.

### 2.1.2   Evolutionary Stable Strategies

In an environment where strategies the strategies reproduce based on payoff from the game (fitness), an extra condition must be added to see what is the best performing strategy.

---

[1]In this case, a strategy that has a p of $b/c$. See (?)

In addition to not gaining a benefit from switching strategies in the immediate game, an Evolutionary Stable Strategy must survive against a population. For example, in the Hawk Dove game given earlier, a single Dove entering a population of Hawks can do quite well. That is, the population of individuals all playing Hawk are beaten by a single Dove invader (and likewise).

For a strategy to be Evolutionarily Stable it must do better than any invader. This is an extension to Nash:

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

In this example, I will call T an invader and x a candidate for ESS. The weak Nash condition is used in this definition. The extra condition considers that any invader will at some point play itself. If the weak Nash condition is met, the invader does as well as the candidate against other strategies. If the second condition is instead equal, it performs equivalently and each invader individual will have the same chance of reproducing as every candidate individual. The candidate does not prevent invasion. If the second condition is met, the invader cannot take over because sooner or later it plays itself, at which point it does worse than the candidate. The candidate is able to prevent invasion, and is an Evolutionary Stable Strategy.

### 2.1.3 Replicator Dynamics

How strategies perform can be analysed with the Replicator equation. The average fitness ($\phi(x)$) of a population ($x$, where $x_i$ is the number of strategy $i$ present) is found by summing total fitness of all members ($f_i(x)$, determined from payoffs) and dividing by the population size ($N$), with $n$ unique members:

$$\phi(x) = \frac{1}{N} \sum_{i=1}^{n} x_i f_i(x) \tag{2.3}$$

The rate of change of the number of $i$ in a population is dependent on the proportion of $i$ in the population, and the fitness in relation to the rest of the population:

$$\dot{x}_i = \frac{x_i}{N}[f_i(x) - \phi(x)] \tag{2.4}$$

Dividing by N normalises the vector containing the number of each strategy type in a population, giving the proportion. Given a proportion of a population, the rate of change can be found. We can visualize how the strategies perform by plotting a phase diagram.

Take the example of the Hawk-Dove game. We'll use three strategies, so they can be plotted on a 2-d simplex. Let $b = 2$ and $c = 4$. The third strategy is a mixed strategy that plays either way with a fifty percent chance.

|  | **Hawk** | **Dove** | **1/2** |
|---|---|---|---|
| Hawk | 0, **0** | 0, **0** | 0, **0** |
| Dove | 0, **0** | 0, **0** | 0, **0** |
| 1/2 | 0, **0** | 0, **0** | 0, **0** |

Table 2.2: Payoff Matrix Hawk, Dove, 1/2

## 2.2   The Prisoner's Dilemma

The Prisoner's Dilemma is a simple model that has been widely used to study cooperation in repeated games (Axelrod, 1997). In the one-shot version of the Prisoner's Dilemma, cooperation is not the expected outcome. In this game, two players play a single game, choosing between two options; cooperate or defect. Players choose simultaneously, and have no information on the opponents choice before their own decision is made. If they both cooperate they receive a score of R (Reward for cooperation). If they both defect, they receive a score of P (Punishment for defection). If one player defects, and their opponent cooperates, the defector is rewarded with T (Temptation to defect). If a player cooperates, and their opponent defects, the cooperator receives a payoff of S (Sucker's punishment). The hierarchy of payoffs goes $T > R > P > S$. Table 2.3 shows the payoff matrix for the Prisoner's Dilemma.

|            | **Cooperate** | **Defect** |
|------------|:-------------:|:----------:|
| Cooperate  | R, **R**      | S, **T**   |
| Defect     | T, **S**      | P, **P**   |

Table 2.3: Payoff Matrix for the Prisoner's Dilemma

Assuming both players are aware of the payoff table and both players act to maximise their personal payoff, defection is the strategy both players choose. If the opponent defects, the best move is to defect too to avoid paying the cost. If the opponent cooperates, the best move is to defect to avoid the cost, and gain the benefit. That is, the player can only lessen the payoff by cooperating- mutual defection is a Nash equilibrium. The dilemma arises from the fact that mutual cooperation would be a better result for both players than the equilibrium of mutual defection. This game captures the essence of the problem of cooperation.

### 2.2.1   Stuck in the Dilemma

When the Prisoner's Dilemma is played multiple times, all parties defecting is no longer the only successful strategy. For the repeated Prisoner's Dilemma, the number of games played is not known by players in advance. If both players knew how many rounds were to be played, the best move in the last round is defection. The same logic causes the second last round to be a defection, and the problem reduces to the Nash equilibrium of the one-shot version. A way to accomplish having an unknown number of rounds is to make future encounters probabilistic, the chance of another encounter between two players is the continuation probability $\delta$.

García and Traulsen (2012) calculate the payoff for strategy A from repeated games with strategy B as

$$\Pi_{AB} = (1 - \delta) \sum_{t=0}^{\infty} \delta^i \pi_{AB}^i \tag{2.5}$$

where $\delta$ is the continuation probability, and $\pi^i$ is the payoff from the $i$th round. The sum represents the diminishing likelihood of another game between the two players. The term $(1 - \delta)$ is for convenience, normalising the payoffs.

Two simple strategies possible in the repeated version are Tit-For-Tat (TFT) and Suspicious Tit-For-Tat (STFT). Tit-For-Tat starts cooperating on the first round, then reciprocates the opponents last move. Suspicious Tit-For-Tat defects on the first round, then reciprocates the opponents last move. If ALLC plays TFT, in all rounds they are cooperative. If TFT plays STFT they alternate defect and cooperate (Figure 2.1). This

behaviour is an example from another model for the evolution of cooperation; direct reciprocity.

| TFT | C | D | D | ... |
|------|---|---|---|-----|
| ALLD | D | D | D | ... |

| TFT | C | D | C | ... |
|------|---|---|---|-----|
| STFT | D | C | D | ... |

| ALLC | C | C | C | ... |
|------|---|---|---|-----|
| STFT | D | C | C | ... |

Figure 2.1: Simple strategies playing the Prisoner's Dilemma

The previous combination of strategies can be examined analytically. Equation 2.5 indicates a natural representation for the payoff of strategies as a matrix. Comparisons can then be made between strategies, as done above but quantitatively. Further, it can be determined what strategies will be selected for (Imhof et al., 2005). For example, a small number of ALLD can quickly dominate ALLC since ALLD gains a better payoff.

### 2.2.2  Assortment

A solution in order to allow evolution of cooperative behaviour is to add structure to the population. This also better reflects most situations that can be modelled with evolutionary simulations (Eshel and Cavalli-Sforza, 1982). In nature, interaction between members of a population is not likely to be randomly distributed; interaction is more likely between kin, or social peers. Structure can be added by changing the probability of interactions- a subset of a population is more likely to interact with that subset.

The expected payoff for a strategy in both a structured and unstructured environment can be found analytically (van Veelen et al., 2012). Consider a strategy space consisting of only always defect and always cooperate (ALLD, ALLC). The potential payoff ($\Pi$) for each strategy can be calculated by the probability of meeting an ALLC opponent times the payoff in that instance plus the probability of meeting an ALLD opponent times the payoff in that instance. Where $N_{TYPE}$ is the number of a type in a population, and $N_{TOTAL}$ is the total population size ($N_{TOTAL} = N_{ALLC} + N_{ALLD}$), the payoffs are:

$$\Pi_{ALLC} = \frac{N_{ALLC} - 1}{N_{TOTAL} - 1} \cdot (R) + \frac{N_{ALLD}}{N_{TOTAL} - 1} \cdot (S)$$

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

In the case of an unstructured population, since $T > R$ and $P > S$, $\Pi_D$ always has a better payoff (excluding the case when $N_{ALLD} = 0$). Instead take the case when the population is structured, and chance of meeting an alike player is r (structure parameter):

$$\Pi^{(r)}_{ALLC} = (1 - r)\Pi_{ALLC} + rR$$

$$\Pi^{(r)}_{ALLD} = (1 - r)\Pi_{ALLD} + rP$$

Sufficiently higher probability of meeting an alike player allows cooperativity to have the greater payoff. Trivially, if probability of meeting like is 1, $\Pi^{(S)}_{ALLC} = R > \Pi^{(S)}_{ALLD} = P$, so cooperation has higher fitness. Substitution of values shows it holds for other cases. For example, if T=3, R=2, P=1, S=0, and the population is 50 ALLC and 50 ALLD:

$$\Pi^{(S)}_{ALLC} = (1 - r)(2\frac{49}{99}) + r * 2 = \frac{98}{99} + \frac{198}{99}r - \frac{98}{99}r = \frac{1}{99}(98 + 100r)$$

$$\Pi^{(S)}_{ALLD} = (1 - r)(3 * \frac{50}{99} + \frac{49}{99}) + r * 1 = \frac{199}{99} - \frac{199}{99}r + \frac{99}{99}r = \frac{1}{99}(199 - 100r)$$

So ALLC has higher fitness if $r > 101/200$. These cases demonstrate that structure can benefit cooperation.

## 2.3   Simulating Strategies

Previously discussed methods of analysing strategies for the repeated Prisoner's Dilemma have some limitations. A notable one being that the specific strategies examined must be chosen some how. If we know or can guess what strategies will be important, this is an effective tool. One way to find what strategies will be important is through simulating an evolutionary environment.

Instead of picking the strategy, we pick a representation method of some sort. The goal of this representation is to take a History $\mathcal{H}$ of previous moves in an interaction, and map it to the action $\mathcal{A}$ the strategy decides to use:

$$f : \mathcal{H} \to \mathcal{A} \tag{2.6}$$

In the case of the simulation presented in this thesis, $\mathcal{H}$ contains previous moves of an opponent– so a sequence of $\{C, D\}$. If instead the full history is recorded – a combination of both what the opponent and the strategy itself did– a sequence of pairs of $\{C, D\}$ or a sequence of results $\{R, T, S, P\}$ would be recorded. **?** and Fogel (1993) are two examples of work using this 'full-memory' representation that will be discussed in 2.4.

In addition to the method to map from a History to an Action, a mutation scheme is also needed; to take an existing strategy and create a new one. This Genetic Programming approach will allow the creation of strategies, without the researcher needing to design or pick any.

### 2.3.1   Representations

Previous work that will be discussed has used primarily a Finite State Automata model for strategies. A simple model, that of the Lookup Table as well as FSA models will be discussed.

**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.

For example, consider a representation that consists of a binary string of length 3 (García and Traulsen, 2012). The first digit represents the move performed in the initial game, when no history exists between two players. A 0 represents a choice to cooperate, a 1 represents a choice to defect. The next two digits determine the choice made, based on the previous choice made by the opponent and discarding all previous history. The second digit in the three digit string describes what the strategy does in response to a cooperation. The final digit describes what the strategy does in response to a defection. Table 2.4 describes some of the possible strategies.

| Strategy | Description | Representation |
|----------|-------------|----------------|
| ALLD | Always Defect | 111 |
| ALLC | Always Cooperate | 000 |
| TFT | Cooperate, then repeat opponents last move | 001 |
| STFT | Defect, then repeat opponents last move | 101 |

Table 2.4: Binary String representation for strategies

The length 3 binary string method can explore a set of $2^3 = 8$ strategies. Increasing the size of the string can allow consideration of larger sets of history. If all bar the last

2 moves are discarded, and so decisions are made based on the last 2 moves, $2^7 = 32$ strategies can be represented.

Part of what makes this a good example representation is that it allows for several example mutation schemes. We could vary the strategies by flipping a bit in the string that represents the strategy. Alternative methods could be used; we could use a 'mutation kernel' that indexes all strategies possible. For a 3-bit lookup table, this would require an 8x8 matrix, that gives the probability of mutating from any of the 8 states to any other of the 8 states. The mutation kernel could assume uniform mutations; that any one state is reachable from any other state in a single mutation. Each term in the matrix could also have some other justification; it could be based on the number bit flips to get from one strategy to another. This matters; García and Traulsen (2012) showed the results from uniform or bitwise kernels differed; cooperative strategies were less common with the bitwise kernel.

Knowing that mutation schemes matter, how should we go about picking one? If we are simulating some biologic process, we could base our mutation scheme off that. In this thesis, strategies in general will be represented. One approach in design for a mutation scheme could be to aim for minimal mutations, for every possible mutation.

Possible mutation schemes for lookup tables are not related to what I have discussed. Other examples could include multiple flips, or inverting the entire string. Crossover could be used; selecting a new strategy with parts from the lookup table of 2 parents.

**Finite State Automata**

A common method used to represent strategies is 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 $\delta : S \times \Sigma \to S$
4. A set of final/accepting states $F \subseteq S$
5. The initial state $q_0 \in S$

An example is shown in Figure **??**. In this example, there are 2 states; $S = \{D, C\}$. The set of accepting states contains one state; $F = \{C\}$. The input alphabet is $\Sigma = \{C, D\}$ and $C$ is the initial state. There are four transitions; $C$ to $C$ when C is read, $D$ to $C$ when C is read, $C$ to $D$ when D is read, and $D$ to $D$ when D is read.
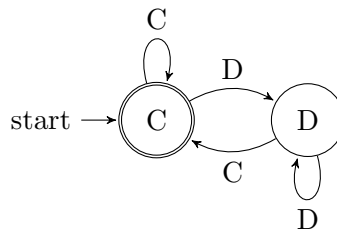


Figure 2.2: Tit-For-Tat as a FSA

It is common that the transition function $\delta$ is a total function in Prisoner's Dilemma's- for every possible input (of a symbol from the input alphabet $\Sigma$) 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 (**?**).

A number of mutation schemes for Finite State Automata could also be used. These could involve adding states, deleting states, changing a transition, changing if a state is accepting, and changing the initial state. This graph based view is used in (van Veelen et al., 2012).

Alternatively, we could set the number of states to a constant or upper limit and represent the FSA by a string; for example a 4 state machine could be given as in Table **??**: The FSA given in **??** is the equivalent of TFT, given earlier. Each column gives what

| 0C | 0D | 1C | 1D | 2C | 2D | 3C | 3D |
|----|----|----|----|----|----|----|----|
| 00 | 01 | 00 | 01 | 10 | 10 | 11 | 11 |

Figure 2.3: String representation for a 4 state FSA

to do if you are at a certain state, and read a certain input. So, at state 0 you transfer to state 0 (00) if you read C. You transfer to state 1 (01) if you read D. So, a FSA can be represented by a string (in this case 0001000110101111) or number. Mutation schemes discussed in Lookup Tables can also be applied to FSA.

Variants on these methods are used in the work discussed.

## 2.4   Related Work

Related simulation work focuses mostly on direct reciprocity; repeating the game being the basis for cooperation to evolve. I will discuss seminal work into strategies for the Prisoner's Dilemma, giving examples of approaches used. Finally, I will discuss the work I have extended.

### 2.4.1   Axelrod's Tournaments

Axelrod and Hamilton (1981) 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, Boyd and Lorberbaum (1987) showed this might not be the case; Tit-For-Tat is not (and indeed no strategy is) evolutionary stable in the repeated dilemma game (Boyd and Lorberbaum, 1987; van Veelen and García, 2010).

## 2.4.2 Evolving Strategies in Simulations

Axelrod (1987) used a simple Genetic Algorithm approach to study the evolution of strategies for the Prisoner's Dilemma (Back et al., 1997). 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 Axelrod (1987).

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

Fogel (1993) 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 (Sipser, 2006). 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 (Axelrod, 1997)- 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.

*Draft of 7:58 pm, November 9, 2014*

### 2.4.4 Miller's Automata

Miller (1996) 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 (Aumann, 1995). Probabilistic length games, as I have used, would avoid this issue.

### 2.4.5 Direct Reciprocity in Structured Populations

**REWORD ALL OF ME** Bergstrom (2003) 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). Bergstrom (2003) 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).

$$D(x) = a(x)b - c \tag{2.7}$$

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 2.5. 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 2.5: Payoff Matrix for the Prisoner's Dilemma, with example values

When sampling from the population (or equivalently, when there is no structure), the payoffs $\Pi$ 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 (Bergstrom, 2003). 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 Axelrod and Hamilton (1981). It was investigated in detail with a simple model with both simulation and analysis by van Veelen et al. (2012). 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 2.6: 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 (Eshel and Cavalli-Sforza, 1982). 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 (García and Traulsen, 2012). 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" (van Veelen et al., 2012). 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 van Veelen et al. (2012) 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

In this section I will discuss the representation and mutation scheme I decided on (Pushdown Automata), the simulation I have created and how analysis was carried out.

## 3.1 Representation

Previous representations discussed were Finite State Automata. The representation I have used in my work is the Pushdown Automata, which can be thought of as a minor extension to Finite State Automata. Other representations may be worth exploring, in regard to the question of how changing the strategy space effects cooperation. These might include other models of computation such as a decision tree model, or FSA with mixed strategies; with transitions with probabilities attached. In this thesis, the focus is on Pushdown Automata.

### 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, $\Gamma$, containing a stack marker $Z_0$, the empty character which indicates no changes to the stack, and other stack symbols
4. The possible states of the stack, $\Gamma'$
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 $q_0 \in S$

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.
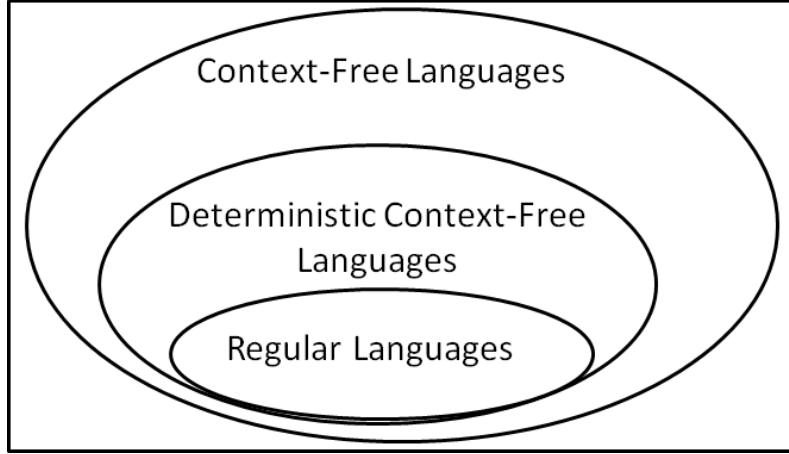


Figure 3.1: Strategy spaces discussed

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.

An example of a Deterministic Pushdown Automata is given in Figure 3.2. It should be familiar– it is a variation on TFT. The character $\lambda$ is the empty character; no change is made to the stack when it is popped or pushed. Following the operation of this Automaton, we start on $q0$. In all circumstances when a $C$ is read, and we are on state $q0$, $a$ is pushed to the stack. If $D$ is read we do one of two things. If the stack is empty, the stack marker $ is on top, so we pop it and move to state $q1$. If the stack has a marker on it, we pop it and remain on $q0$. If we have transferred to $q1$ and read $C$, we push $a$ and transfer back to the accepting/cooperating state. If we read $D$, we remain on the non-accepting state.

This example DPDA counts the number of times the opponent cooperates. If they have cooperated recently, it does not retaliate until defection moves outnumber cooperation moves.

### 3.1.2   Deterministic Pushdown Automata

The Automata used in my simulation are designed from a graph point of view. Each state in a DPDA is an object inside that DPDA, and each transition from a DPDA is an object inside that state. A DPDA is defined in the simulation by its initial state, its states (which include a boolean accepting/non-accepting marker) and its transitions (each belonging to a source state, and having a destination, pop instruction, push instruction and read instruction).
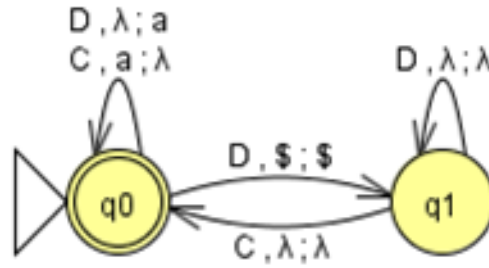
*Draft of 7:58 pm, November 9, 2014*

Figure 3.2: A DPDA modelled off TFT

### 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.

This scheme can explore the entirety of the strategy space that can be represented by Deterministic Pushdown Automata which accept by final state, which includes all Finite State Automata.

## 3.2   Simulation

Simulations were created in Java, extending existing code for simulating repeated games (**?**). The simulations accept a collection of parameters specifying the simulation, and output either a time series or an estimate of the payoff of the population. The time series contains the state of the population in specified generations. It reports the payoff of the population in that generation, the strategies present in the population, and the number of each strategy present.

The Monash Campus Cluster (**?**) was used for running simulations.

## 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 Discussion

# 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

Aumann, R. J. (1995). Backward induction and common knowledge of rationality, *Games and Economic Behavior* **8**(1): 6–19.

Axelrod, R. (1987). The evolution of strategies in the iterated prisoner's dilemma, *Genetic Algorithms and Simulated Annealing* pp. 32–41.

Axelrod, R. (1997). *The Complexity of Cooperation*, Princeton University Press, Princeton.

Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation, *Science* **211**: 1390–1396.

Back, T., Fogel, D. B. and Michalewicz, Z. (1997). *Handbook of evolutionary computation*, IOP Publishing Ltd.

Bergstrom, T. C. (2003). The algebra of assortative encounters and the evolution of cooperation, *International Game Theory Review* **5**(03): 211–228.

Boyd, R. and Lorberbaum, J. P. (1987). No pure strategy is evolutionarily stable in the repeated prisoner's dilemma game, *Nature* **327**: 58–59.

Eshel, I. and Cavalli-Sforza, L. L. (1982). Assortment of encounters and evolution of cooperativeness, *Proceedings of the National Academy of Sciences USA* **79**: 1331 – 1335.

Fogel, D. B. (1993). Evolving behaviors in the iterated prisoner's dilemma, *Evolutionary Computation* **1**(1): 77–97.

García, J. and Traulsen, A. (2012). The structure of mutations and the evolution of cooperation, *PLoS One* **7**: e35287.

Imhof, L. A., Fudenberg, D. and Nowak, M. A. (2005). Evolutionary cycles of cooperation and defection., *Proceedings of the National Academy of Sciences USA* **102**: 10797–10800.

Miller, J. H. (1996). The coevolution of automata in the repeated prisoner's dilemma, *Journal of Economic Behavior & Organization* **29**(1): 87–112.

Sipser, M. (2006). *Introduction to the Theory of Computation, Second Edition*, Course Technology, Boston.

van Veelen, M. and García, J. (2010). In and out of equilibrium: Evolution of strategies in repeated games with discounting, *Technical report*, Tinbergen Institute Discussion Paper.

van Veelen, M., García, J., Rand, D. G. and Nowak, M. A. (2012). Direct reciprocity in structured populations, *Proceedings of the National Academy of Sciences USA* **109**: 9929–9934.