University of Oxford

Cognitive Dissonance in Social Networks

Alexander Campbell

Brasenose College

A thesis submitted in partial fulfilment of the requirements for
the degree of Master of Philosophy in Economics

Word count: approx. 24,000

May 2007

Department of Economics, Manor Road, Oxford

Acknowledgements

I would like to acknowledge the contribution of my advisors Professors Peyton Young and Christopher Wallace for their supervision and input throughout this process. In addition, I must thank my undergraduate advisor Professor Jim Engle-Warnick of McGill University for opening up a whole new area of economics to me and saving my economics education. Lastly, I would like to thank my editor and moral support Jessica Prince.

Abstract

The theory of cognitive dissonance, originally developed by Leon Festinger, explains how individuals experience significant discomfort upon holding two psychologically inconsistent belief or attitude. This paper takes two essential insights from cognitive dissonance and presents an an analytically tractable model of cognitive dissonance in order to investigate how political discourse occurs in social networks. The first insight is that political disagreement between individuals leads to dissonance, and as a result of the urge to reduce dissonance, individuals alter their political ideology. The second insight, is given the fact that disagreement causes dissonance, individuals may choose to avoid interacting with people they know they will disagree with.

The primary aims of this paper are to demonstrate how a behavioral system ruled by these two forces from cognitive dissonance can fit neatly into the accepted analytical framework of evolutionary game theory, and also, to use both analytic mathematics and agent-oriented simulations to examine what the behavior of that system says about the interaction between ideological choice and social segregation.

These two methodologies generate several mutually affirming results. The first is both an evolutionary model and a simulation based model of cognitive dissonance yield a strong push toward local convergence in the short term, and global convergence in the long term. This is a result of the stochastic stability of homogeneous political states as compared to segregated heterogenous states. The second result is that there is a relationship between the social segregation of a group and the political segregation of the society, this paper takes an approach which provides a mechanism by which this plays out across

groups. The third result is that when a system reaches convergence, it tends to spend more time in the center of the political spectrum than the extremes. Finally, that increasing the number of agents in a system significantly enhances the systems ability to sustain distinct ideological social groups.

Contents

Chapter 1

Introduction

This paper has two central and complimentary goals, both concerned with the synthesis of ideas across disciplinary boundaries. The first is the adoption of cognitive dissonance, a model of behavior from social psychology, into a tractable framework eligible for analysis using components of the economics toolkit, in this case evolutionary game theory. The second goal is applying the insights developed from this exercise to real problems in political science, with a focus on the development of political consensus, persistence of ideological segregation, and the integration of distinct groups into society.

In accordance with these goals, this paper can be viewed as an argument. Starting with a basis in psychology, working through techniques developed by economics of modeling human behavior and interaction, and culminating in results applicable to the real world. The complexity of this formulation requires a significant devotion to justifying the links within and between the steps involved, and the format of this paper acts accordingly.

What are these steps? First; there is an explanation and justification for the use of cognitive dissonance in general and then specifically as applied to ideological choices in a social environment. Second; this literature is developed into a form on which modeling can take place. This includes both mathematical analysis, and agent-oriented simulations of the evolution of ideology within a social network. Third, using the insights gained from both the pure modeling and the simulation data, we can make inferences for a system guided by dissonance and note where

and how it weighs on the real world. Finally, we take these insights as a basis for preliminary policy recommendations.

Chapter 2

The Theory of Cognitive Dissonance

2.1 Festinger's Theory of Dissonance

What is cognitive dissonance?

Cognitive Dissonance is theory of behavior developed by social psychologist Leon Festinger. According to his theory, cognitive dissonance is a state of tension that occurs when individuals hold two psychologically inconsistent cognitions (beliefs, ideas, attitudes, opinions).(17) Two elements are held to be dissonant when the opposite of one follows from another. Dissonance is caused when one is exposed to new information or takes an action which either introduces a new cognition that in turn clashes with previous cognitions, or alters old cognitions in such a way as to contradict other cognitions.

The magnitude of dissonance is a function of two things: the relevance and importance of the cognitions. The relevance criterion is concerned with the connectiveness of the elements. For illustration purposes, consider two things that are rather far removed from one another: the statistical knowledge of baseball and one's views on the minimum wage. These two cognitions are completely irrelevant to each other. The veracity of one of these ideas has no bearing whatsoever on the other, and thus there is very little potential for dissonance between the two. If you learn new information about baseball, this means nothing for your views on the minimum wage.

The magnitude of dissonance is further effected by the importance of the cognitions in question. Two elements can be equally contradictory, but we suffer more due to contradiction when the issues are more important to us. Take for example, the behavior of a politician's support base. Any case where a politician makes a decision that conflicts with the ideological beliefs of his or her supporters triggers dissonance on the part of these supporters. In helping a politician get elected, an individual places trust in that politician to make decisions she would agree with while in office. When a politician goes against a supporter, the cognitions 'I helped her get access to power' and 'she went against my beliefs on issue X' are in some conflict. It may not be the case that disagreeing over one issue is enough to cause tremendous dissonance, but as the importance of the issue increases, the magnitude of the resulting dissonance does as well. When George Bush nominated Harriet Miers for the Supreme Court, there were rumblings among even the most strident Bush supporters regarding her relative lack of experience and the apparent cronyism of the executive's legal counsel being nominated for such an important role. Here, the cognition 'George Bush is a good and competent President' conflicted with the cognition 'a good President will find the best person for the job.' Given the ardent nature of Bush's support base in the US, even these initial rumblings were signs that his supporters were in the throes of dissonance. However, what eventually sunk Miers' nomination was not questions regarding her qualification, but media reports on her previous statements in support of the *Roe v. Wade* decision. Bush's religious base was outraged that he could possible nominate someone who would uphold *Roe.* Here, the cognition 'I will support President Bush' conflicted with 'the next Supreme Court nominee should seek to strike down *Roe*', and as we shall speak about below, the second cognition won.

In the real world, the causation may be much more complicated, however, the point of the example is clear. As the contradiction involves cognitions which are more central and important to the individual, the resulting pressure to reduce that dissonance is increased. Also, we can predict which cognition will be altered by the relative strength of each cognition. *The Reduction of Dissonance*

The simple way in which individuals reduce dissonance is by endeavoring to change the ideas or the environment which causes the dissonance in the first place. Broadly speaking, this can occur in two ways: either by eliminating the contradiction between the cognitions, or by attempting to attenuate the magnitude of dissonance through decreasing the relevance and importance of relevant cognitions. To adapt a well known aphorism, if you can't beat 'em, ignore 'em.

The elimination of a contradiction occurs by altering dissonant cognitions or introducing new consonant cognitions in order to bridge a gap. An example here is helpful. Let us say that I have a friend, Steve. Steve and I go way back. On the other hand, I recently learned that Steve ran out on his family, leaving them high and dry. If I like to think that my friends are good, honest and kind people, then learning about Steve's behavior causes myself considerable cognitive dissonance. Clearly there is a contradiction between the cognitions, "all of my friends are good people," "Steve is my friend" and "Steve ran out on his family." As long as my definition of good person excludes this type of behavior, then these three cognitions are contradictory.

What are the options available to me? The first is to alter the cognition "Steve is my friend". If Steve is no longer included in the set of people I consider to be my friends, and the people for whom the first cognition applies, then there is no contradiction. Steve is a guy who used to be my friend, but did something I find appalling and therefore we are no longer friends.

The second option is to alter the cognition, "all of my friends are good people". Removing this cognition from my criteria for friendship eliminates the dissonance. I can then befriend whoever fulfills my other criteria for friendship; that is, that they "enjoy my company" and "I enjoy their company." This says nothing about the difficulty of altering the cognition, which as we will see, is entirely another matter.

The third option would be some combination of the two. That is, change both of the cognitions slightly so as to bring them more in line. This would entail broadening the criteria for friendship to be less draconian as to the character of my associates, and also shifting Steve down a couple rungs in the friendship hierarchy as a result of his actions.

The fourth options is, in a sense, cheating. That is, to re-evaluate the last cognition, the statement of fact. In order to reduce the dissonance, I may reinterpret the third cognition regarding Steve's actions or question the validity of the information. When anyone speaks of how individuals can be predisposed to 'colour' new information in a certain way without realizing it, they are referencing the role that cognitive dissonance has in the biased interpretations of events and information.

This was one of the central advances in the development of the theory of dissonance, one which has been so widely accepted in the psychological community as to filter into the public wisdom. If it is possible that individuals, and even entire institutions such as media outlets, government bureaus, and think tanks can be systematically biased in their interpretations of new information, then cognitive dissonance provides a powerful and compelling story of the mechanism by which bias operates. In 1979, Lord, Ross and Lepper published a well known paper on the topic of biased assimilation, however, it was done completely under the umbrella of cognitive psychology and not dissonance theory[1]. Their results were completely in agreement with the original theory, but classified under a different heading. Both Lord et al and Festinger both argue that individuals interpret empirical evidence in a biased manner when it relates to strongly held opinions.(28) The difference is that Festinger's theory of dissonance provides a motivation, whereas cognitive psychology removes motivation from the equation. Another classic example of this phenomena is the reception smokers gave to new information from the surgeon general regarding the health risks associated with smoking. Kassarjian and Cohen found that people who smoked were significantly less likely to allow initial reports of the dangers of smoker change their attitudes(24).

Back to our example of Steve, discarding the accuracy of the information may be difficult in this case. It would seem to be easily verifiable. Therefore, all that I may be left with is rationalization. The concept of rationalization is implicitly tied to dissonance theory.(7) Rationalization is an attempt to attenuate the connection between cognitions without directly altering the cognitions. In our example, an explanation of behavior which did not categorize Steve as a bad person as a result of his actions would suffice. Explaining his behavior as a function of unusual stress, labeling it as a one-off extraordinary event, or justifying his actions on the basis that the family was unhappy anyway, would all be attempts to put distance between the third cognition and the first two. If the first two cognitions

[1] the distinction between the two is unimportant for our purposes

are particularly resistant to alteration, and the validity of the information is unquestionable, then all that is left is rationalization. This is an example in which I would be diminishing the relevance of the cognitions in order to relieve the pressure of dissonance.

Finally, I can make a conscious or unconscious decision to render all of these cognitions less important to me. It may be that I have great resistance to writing off Steve, and it may be unreasonable for me to say that I don't care about the character of my friends, the information may be irrefutable and no alternative explanation for his behavior exists. Ultimately, I might just say "I don't care, I know what he did, and it was awful, but he has been my friend for a long time and the friendship means too much to me."

If it seems that the combinations of all of these options is rather complicated, that is because it is so. What goes on behind the curtains of the mind is extremely subtle and in no way as straight forward as I have explained here. The central concept to take away from this foray into social psychology is that dissonance is caused by contradictions between cognitions, that cognitions can be any idea that is given credence by individuals, and that the pressure resulting from this dissonance can be so great as to challenge long-held beliefs, distort the interpretation of events, or change behavior.

An Extension to the Theory

The original theory of dissonance outlines how the relevance between cognitions and the importance of cognitions determines the magnitude of the discomfort resulting from contradictions between them. An important step in this reasoning is that there are very few cognitions that are not intrinsically tied to how we view ourselves. In a sense, by re-orienting dissonance theory to focus on the self image, Elliot Aronson identified a group of cognitions which almost always qualify under the two criteria outlined by Festinger. Aronson suggested that as the self-concept was so important to the individual and relevant to so many things, the preservation of a specific type of self image has tremendous implications for how we experience and diminishing dissonance.(5) Individuals are said to strive for three things:

1. To preserve a constant, stable, predictable sense of self.

2. To preserve a competent sense of self.
3. To preserve a morally good sense of self.

In this way, Aronson explains that dissonance is most acute when it calls into question our fundamental quality as a person. When Festinger speaks of lying as creating dissonance between the cognition "I said 'X' " and "I believe 'not X'," Aronson's version of dissonance speaks directly to the ramifications of lying on our self image. It is not that these two cognitions simply contradict, it is that the act of lying threatens one's self image a moral, truthful, decent person. After all, do decent, reasonable people go around lying?

Not only does the contradiction cause discomfort, but the idea "I have said something I do not believe and it could have bad consequences for people" is also dissonant with a self-concept that holds that I am an honest and good person. As Aronson states, I am pressured to reduce dissonance when I have done something or have been informed of some information that astonishes me, makes me feel stupid, or makes me feel guilty.(6)

2.2 Empirical Support for Dissonance Theory

The argument that any theory can and should be applied to broad social phenomena requires a great deal of justification. In effect, by suggesting that this particular theory should be applied in this way, one is saying that other theories in the same area should not be so applied; it is not enough to simply say that this theory looks like it applies. So what makes a good theory for this purpose, and why does cognitive dissonance qualify? The first criterion is *validity*. Does the theory have an internally consistent logic? Does it provide plausible explanations, and are there any obvious empirical results or verified competing theories which violate these predictions? The previous section partially dealt with the criteria of validity, and more discussion of this will follow. The second criterion is *potency*. Again, in order to argue that dissonance should be applied to something as important as the choice of one's political ideology, the reader must be convinced that dissonance is conceptually powerful enough to have a real and measurable effect on these political ideologies. Presumably, choice of political ideology is a powerful cognition and one especially resistant to change. Before explaining how dissonance applies to this choice, it is crucial to demonstrate that dissonance is sufficiently powerful to alter these significant cognitions. This section provides the empirical justification for the claim of potency, while buttressing the initial claim of validity.

In the initial presentation of the theory, Festinger provides evidence supporting the claim that individuals deal with dissonance by changing behavior and opinions. Festinger finds evidence, albeit in a non-rigorous fashion, that: a) following a decision, individuals actively seek out corroborating information and discount or misperceive contradictory information (pgs 83, 176), and b) that following public compliance, forced or otherwise, there is a subsequent change of private opinion over and above what would be expected in the absence of dissonance (pg 122). Furthermore, he finds that in both of these cases, the magnitude of the resulting behavioral shift is proportional to the magnitude of the dissonance. This evidence is consistent with the claim that the relevance between dissonant cognitions, and the importance of those cognitions to the individual are both proportional to the magnitude of the resulting change in behavior or opinions.

This so-called forced compliance effect has been verified by numerous studies, using more rigorous methodologies. Festinger and Carlsmith find that not only is there convergence between private opinion and overt, forced behavior, but that the magnitude of the opinion change is *negatively* related to the pressure used to force that behavior. (18) This is important. Festinger and Carlsmith compelled subjects who had just completed a rather monotonous experiment to convey the opposite opinion to subjects who were about to endure the experiment. They find that paying subjects \$1 as opposed to \$20 results in them reporting, post-hoc, a more enjoyable experience. People who are paid very little to lie about how much they enjoyed a task retrospectively change their opinions about how much they actually did enjoy it. These results were further affirmed by numerous studies(21)(19).

Through a dissonance lens, this makes a great deal of sense. Subjects who are paid \$20 can justify the lie on a monetary basis more than those paid \$1. Nobody likes to lie, but if you are going to lie, you might as well be well paid for it. If you are not paid very much at all, in order to justify lying, you are more likely to convince yourself that you were not telling a lie. Insofar as we can trust the results of survey data, underpaying subjects for lying actually altered their subjective recollections of the task.

Note that this result could also be interpreted as changing the *reported* experience, for fear of being judged by the experimenter as one who lies with little coercion. This interpretation suggests that the individual's subjective recollection is actually unchanged, but that subjects are willing to bias their report in order to save face. This issue is present in much of the literature in which there is a personal interview. However, this interpretation does not contradict dissonance theory, in fact, it just suggests there are other routes through which it operates.

This result was supported by Aronson and Mills in their experiments on barriers to group entry. (8) They find that there is a positive correlation between the difficulty of obtaining membership to a group and the reported quality of the group after after entry. This provides a compelling explanation for the persistence of hazing, the practice of subjecting newcomers to severe initiation rituals in sports teams, fraternities and social clubs. Dissonance suggests that not only do individuals retrospectively alter their perception of initiation rituals in a positive manner, but the act of hazing can cultivate loyalty among newcomers to the group. If Josephine endures a tremendously painful and embarrassing initiation ritual, she now has a personal stake in the group being worthwhile. After all, only a fool would suffer to enter a group which in unrewarding, and only a greater fool would suffer to join and then immediately leave.

This retrospective change in judgement is a common theme in the cognitive dissonance literature. Condensed, it explains that after an irrevocable decision is made, in order to justify the action, people may either change their recollection of the subsequent consequential experiences, or alter their opinions so as to reconcile their self-image with the action.

These results strongly contradict the theory of catharsis, a contemporary theory of dissonance, which holds that injuring another person reduces the hostility held towards them. Several experiments in the 1960s demonstrated that after injuring another person, individuals subsequently alter their opinions to be even more negative. This move towards greater hostility justified the irrevocable act of hurting the other person. (15) (20) While catharsis predicts that hurting a person reduces hostility by "getting it out of your system," dissonance predicts the exact opposite. The knowledge that I hurt another person is a pretty unassailable fact, and as far as cognitions go, it is rather hard to alter. In order to reduce dissonance caused by a positive view of the self and injuring another, dissonance theory predicts that an even more negative view of the victim is called for, in order to justify extreme action.

Ten years after the publication of Festinger's theory, Cummings et al published a review of 23 studies on the relationship between cognitive dissonance and consumer choice. These studies had a particular focus on attitude change as a result of the introduction of new information and/or binding consumption decisions. They find that "the evidence in favor of dissonance theory in the consumer behavior literature looks good."(14)

Adverse Consequences

The principle challenge to the Aronson-Festinger model of dissonance came in the form of Cooper and Fazio's "New Look" theory. The New Look perspective argues that the primary driver of dissonance is the distress from being personally responsible for an adverse outcome.(13) However, Thibodeau and Aronson(31) as well as Aronson(6) sufficiently rebuke this attempt to shift the grounds of the theory. They provide numerous examples of studies in which either there is an attitude change, even when the adverse consequences in question are revealed to not have occurred, or the consequences in question are actually *positive*.

The latter is of particular interest because it flies directly in the face of predictions from the New Look perspective. Here, the focus is yet again upon the link between public action and private opinions. However, these studies are not plagued by the problems of survey data. While Festinger relies on reported opinion change, Dickerson et al find that after asking subjects to express the benefits of water conservation, these people then use less water in the future.(16) This is solid evidence that dissonance can change more than opinions; it can alter actual behavior as well. Stone et al asked university students to openly advocate safe sex, and then reminded the students of their failure to use condoms in the past. They find that students who are the most hypocritical, ie: those experiencing the most dissonance, are more likely to buy condoms and buy in greater amounts on average than either those not experiencing dissonance or those in the control group.(30) Again, according to the New Look theory, dissonance is only spurred on by unintended adverse consequences. If I publicly advocate water consumption or safe-sex, and other people change their behavior as a result, while I do not, then this is a net positive result, and any unintended consequences can only be construed as positive.

There is evidence that the severity of the adverse consequences does have a bearing on the magnitude of the resulting opinion change, but really, there is nothing in the Cooper-Fazio New Look theory which cannot already be represented in the Festinger-Aronson theory. In fact, the New Look perspective significantly reduces the scope and parsimony of dissonance.

Sunk Costs

Can dissonance work in the other direction? That is, can it not only retrospectively change beliefs due to actions, but also change future actions as a result of previous beliefs and actions? It would appear that there is no reason why not. Here, we have stumbled upon a link between dissonance and sunk costs. In introductory economics lectures, after explaining the concept of a sunk cost, the lecturer then goes on explain how we act in an all-you-can-eat establishment and how this violates the concept of sunk costs. Why should the amount one paid to eat have any effect on the quantity of consumed food? Shouldn't we all just eat exactly to our satiation point? Cognitive dissonance provides an answer to this age old question. People stuff themselves to exhaustion in an all-you-can-eat pizza joint because it would be tremendously foolish to spend $10 on all-you-can-eat pizza and then only eat $8 worth of pizza. So even if I only *want* $8 of pizza, you can bet that I will eat more than $10 worth of pizza.

2.3 Economics and Cognitive Dissonance

The manner in which economics assimilates a theory like cognitive dissonance can explain a great deal about how the field is progressing, in terms of Stigler's quest as the 'Imperial Science'.(29) The first barrier to adoption of a theory is the reasonable reticence of economists to adopt neophyte theories from untrusted disciplines. It would take too much time and energy not being spent on economics, to be constantly apprised of the content of other disciplines. Especially when many hot topics turn out to be passing or unfounded. In this way, barriers between fields can function as screening processes. Eventually, the ground-breaking research cross barriers. The ability to recognize and incorporate robust findings from psychology in particular has been at the center of the recent surge in experimental and behavioral economics. Simple insights into how people view relative losses and gains, and the way that humans judge probabilities led to prospect theory,(23) by some measures one of the largest impact papers in economic history.(26) A cursory Google scholar search reveals that Kahneman and Tversky, two *psychologists*, might actually have the most cited paper in economics history. Their relatively simple paper completely devastated one of the landmark tools of the economist's toolkit: expected utility theory.

At the same time, prospect theory might also hold the record for the most-cited, least-applied work in history of economics. How is this possible? Here we find the second barrier to incorporating external theories into economics: tractability. The subtle nuances brought by Kahneman and Tversky to decision theory are simply too difficult to model. So economists, wanting to present their understanding of the field, cite the paper, explain how they cannot use it in their particular situation, and proceed on with their work as if nothing had ever happened. That is how one can have the most cited paper in economics without anyone being able to actually apply the contribution. Obviously, this is not the fault of Kahneman and Tversky, but rather a systematic problem in the integration of knowledge from different disciplines. Even within the social sciences, disciplines often use vastly different methods to answer questions, therefore uniting research across fields is of considerable difficulty.

This exposition is important when considering the impact, or rather the lack of impact, of cognitive dissonance on the economics community. The first barrier, that of reasonable ignorance, clearly applied to Festinger's theory early on. The second barrier, that of tractability, probably also contributed as well to the significant delay in adoption which occurred after Festinger first published his theory. Searching amongst three of the most prominent journals in economics over the past 40 years, *The Journal of Political Economics*, *The American Economic Review*, and *Econometrica*, the first published paper in which cognitive dissonance is explicitly addressed is Akerlof's 1982 paper(4), auspiciously titled "The Economics of Cognitive Dissonance".[1] Akerlof provides three propositions for incorporating dissonance theory into economics:

1. Individuals not only have preferences over states of the world, but also over their beliefs about the state of the world.

2. Individuals have some control over their beliefs; not only are individuals given choice over beliefs, given information, they can also manipulate their own beliefs by selecting sources of information likely to conform to "desired beliefs."

3. Beliefs, once chosen, persist over time.

There are two essential lessons from Akerlof's paper, the first being that it is possible to adapt the standard rational model to incorporate cognitive dissonance and in so doing, better predict and explain human behavior. The second is that an individual's preferences over beliefs are formed in part due to the environment of that individual. The operative example from the paper is an employers' beliefs about the riskiness of a job. Akerlof demonstrates that given an *a priori* choice to work in an unsafe environment, and a subsequent revelation of a lack of safety provisions, workers have incentives to alter their beliefs about the safety of the job. Here, the two cognitions in conflict are preservation of the self, and working in a job which places the self at significant risk. The world in which workers believe, in spite of information to the contrary, that the job is safer, is preferred by workers to the world in which they worry about the risks that they are facing daily. Note that it is not the actual threat of injury which changes the beliefs in this model, it is simply the discomfort resulting from accurately measuring the risk that provides incentives for workers to alter their beliefs. While it does not

[1] Previous to Akerlof's paper, there are three other papers that reference have Festinger's book without specifically mentioning the phrase in the body of the paper. Searching the economics field on JSTOR, there are under 10 papers in the whole field previous to Akerlof's that mention the theory as applicable to a relevant issue. None of those papers address a workable model for dissonance or place it at the center of their claim.

predict traditionally rational behavior, in that workers are ignoring information, workers are better off with altered beliefs, especially those committed to the job. Hence, this can be considered a rational decision, even if the engine of rationality is subconscious dissonance reduction.

This speaks to one of the central assumptions of this paper: while dissonance is reduced by subconscious actions, it can still be considered rational and hence, modeled as such. A frequent layperson's critique of economics is that utility maximization seems rather farfetched if you impose a conscious rational choice framework on individuals. When I go to the supermarket, I am obviously not computing the relative price and marginal utilities of all of the thousands of goods. However, we judge this particular choice of modeling by evaluating the way in which it models behavior and on the accuracy of those behavioral predictions. Any model with both a plausible motivation and accurate predictions is considered a good model. If we can say that rational utility maximization represents a subconscious process, it seems a clear extension, to claim that a proven subconscious process can be modeled in a rational choice fashion.[1]

In a subsequent paper, Akerlof examines another area of interest for this paper, the operation of conformity through social distance. In *Social Distance and Social Decisions*, Akerlof looks into the connection between conformity and social distance.(3) He provides a quadratic utility function of the form:

$$U = -ax^2 + bx + c - d(x - \bar{x})^2 \tag{2.1}$$

Equation 2.1 explains that agents lose utility $-d(x - \bar{x})^2$ due to failing to conform. The intrinsic value to the agent is $-ax^2 + bx + c$, this results in a unique equilibria for all agents of $x = b/2a$. In some ways, this parallels one of the manners in which this paper models cognitive dissonance.

Konow examines the implications of cognitive dissonance for experimental results in bargaining games.(27) In dictator games, one player is given the power to decide the allocation of a pot of money between herself and another player. As opposed to other bargaining games in which both players have some power to determine the allocation (either through an implicit of explicit power of rejection),

[1] with the obvious proviso that the model predicts the same type of behavior as the process

in dictator games one player is totally in control. The absence of uncertainty or strategic considerations on behalf of the "dictator" player implies that proposals by dictators reveal their direct preferences. Konow hypothesizes that players experience dissonance through the contradiction between cognitions relating to the selfish desire for money and the wish to fairly divide the money. Based upon the standard predictions of dissonance, we should expect to see two manifestations of dissonance reducing behavior: dictators making more generous proposals and a belief that it is somehow fair for dictators to take more than the 'fair amount'.

———-

2.4 Political Discourse in Social Networks

Up to this point, the discussion has been concerned with dissonance occurring in a social world, but not as a result of social interaction. There is no reason to presume that cognitions arising from social interactions function differently than any other cognition derived from the non-social world. Clearly, cognitive dissonance is relevant to interactions with friends and neighbors This paper takes the view that via Aronson's specification of dissonance, we can see how dissonance from social interactions occurs and is mediated.

In order to connect the discomfort of social disagreement and cognitive dissonance, we will work with the old adage of not bringing up contentious issues at dinner with polite society. The dissonance lens provides two explanations for this norm. The first is that individuals do not like being confronted with alternative views on issues that they hold particularly dear. As previously mentioned, simply the introduction of new information or perspectives into our cognitive space can cause considerable mental consternation as a result of dissonance with our own views.

The second dissonance explanation is considerably more subtle, and yet a more developed view of this phenomena. If the purpose of a dinner party is to become better acquainted with new people, or even to strengthen the bonds between old friends, disagreement over important issues directly combats that goal. This is the relationship between liking a person and agreeing with them

This process occurs through two avenues. The first mechanism by which political disagreement causes individual dissonance is the act of being confronted with alternative viewpoints and political beliefs. This causes dissonance by introducing new claims which contradict previously held cognitions. The old adage of what is proper to discuss at a dinner party applies here.

The key contention here being that if cognitions relating to the self-image are particularly strong, then that the implications of who we choose to befriend and interact with has relevance to this self-image. In a sense, through a dissonance perspective, we extend some of the same personal moral requirements to our associates that we apply to ourselves. If I am an upright, honest and decent person, then clearly the people I choose to spend time with should be of a similar mold.

An alternate explanation for the convergence of opinions in social clusters is that common attitudes are spread in a population through the effects of conformity. Abelson provides a model of convergence through conformity in a mathematical context. Abelson takes a model with a strong assumption that two individuals will converge in attitudes upon interaction. Applying this assumption, Abelson demonstrates how universal agreement is inevitable for any population where there is at least one member of the population who is in connected, directly or indirectly, to everyone else in the network.(2) Abelson defines this type of network as a compact network, and this condition is the linchpin of his analysis. *Cognitive Dissonance and Political Choice*

Every member of society arrives at their particular ideological preferences by coming into contact with the diversity of views on a broad range of topics in our environment and then choosing their position from amongst the available political categorizations. In other words, the source of justification for individual political and ideological preferences is in some sense derivative of external stimuli. This can be accurately described as a deterministic perspective on human behavior. It stands to reason that if all behavior can be attributed to some combination of genetic and environmental factors, then political preferences are potentially the result of genetic predispositions to particular ideological arguments presented to individuals by their environment. For example, we are not born with a preconceived belief on the subject of the marginal tax rate. As a result, it must be that political ideology in some important sense a direct function of environmental factors.

Admittedly this is an extremely determinative view of human development. In short, we may have genetic predispositions, but our environment matters more. Much as everyone would like to believe they have excellent internally and philosophically consistent reasons for their particular political ideology, this just cannot be the case. Political ideology has to be, in large part, a product of environment. Any other contention just does not hold water when brought to bear on the wealth of human social and political history. it just does not hold water when compared to the vast reservoir of political history. History is littered with and societies that engaged in practices that are now seen as abhorrent: slavery, segregation, the legal status of women, the exploitation of the poor and the legacy of indifference to the suffering of others, etc. If it were true that our beliefs were completely self-determined, and intrinsic to our inherent characteristics, if politics was purely in the genes so to speak, given that we now find these practices to be immoral, then sustaining these institutions would have been impossible.

Chapter 3

The Model

The previous section spent much argumentative energy justifying the use of cognitive dissonance, and with good cause, as the rest of the argument is based upon the legitimacy of this theory. There was an emphasis on explaining how interactions with other people can cause dissonance, and the behavioral consequences of this dissonance. Again, this is a justification of the application of dissonance theory to this particular problem.

Before delving into the mathematics of a system bound by cognitive dissonance, and departing from the norm, an important objection must be addressed. This objection also serves as the bridge in the argument between proving the applicability of dissonance theory to social-political questions and the justification for the choice of modeling framework.

Recall the two central, relevant insights from the previous section: first, that individuals moderate their opinions so as to reduce conflict and dissonance with people that they like and respect; and second, that individuals seek out others who are close to them in political or ideological space to interact with in preference to those that they disagree with. On some level, the two insights above seem relatively facile; there is no profound gain in the understanding of behavior in stating that individuals seek out those with whom they have things in common. On the other hand, this suffers from a problem common to many predictive behavioral models; that is salient examples of events or individuals which operate in direct opposition to the predictions of the theory. There are numerous, easily recalled instances of individuals who seek disagreement or seem impervious to belief changes in the face of persistent dissonance. For example, those who seek politically charged conversations with acquaintances with whom they disagree; the first year undergraduate looking to pick a fight over politics. Those that have lived for years in an environment hostile to their ideological perspective without changing in the slightest; such as the Republicans in San Fransisco or the abortion rights activists in evangelical Colorado. Those who make a significant break with the political tradition of their environment or social network; the John Stuart Mills. Finally, those that seem to be capable of constant resistance to cognitive dissonance while living a life fraught with hypocrisy and ideological inconsistency.

There are a few responses to this objection. The first is that the reason these examples are salient is that they strongly stand out against a backdrop of relative adherence to the principles of cognitive dissonance. Further, the presence of salient examples has been consistently shown to lead to the systematic bias in perception resulting from mental heuristics.(32) As a result of such bias, easily recalled events or examples, even those that occur with low probability, can lead to vast misperceptions of frequency or significance. The classic examples here are the obsessions of the media and general public with the catastrophic losses that occur with tiny probabilities: the fear of flying as compared to the fear of driving; perceptions of the relative danger to small children of having a pool in the backyard as compared to having a handgun. This does not invalidate the theory, rather it demonstrates a lack of uniformity in its application across society.

Further, just as there is innumerable variety in human appearance and appetite, there can be variation in the measure to which cognitive dissonance has relevance to individuals. The legitimacy of cognitive dissonance is not wholly dependent upon its applicability to every person, but rather its predictive power of aggregate behavior.

If the reader finds this explanation unsatisfactory, there exists another defense in drawing parallels to the the general practice of economic modeling. In modeling any behavior, it is traditional practice in economics to provide some sort of objective function upon which agents optimize. This objective function is sometimes labeled a utility function. In the same way that we know that everyone in society does not have the same utility function, we understand that cognitive dissonance impacts various individuals with different strength. This paper will apply objective functions, but most of the dynamics of the system do not require a specified function for the dynamics of the system to hold.

The final response to the salient objection is to surrender and the to assimilate the critique into the modeling techniques of the paper. Strong assumptions regarding the homogeneity of individual preferences, by way of objective functions *are* indeed dangerous to the parsimony of results. However, the use of evolutionary game theory in examining social phenomena can address this objection directly in the modeling framework. Evolutionary game theory imposes relatively low demands on the rationality and foresight of individuals. Given the applicability of this type methodology to the problem at hand, this paper adopts evolutionary game theory in mathematically analyzing a social system bound by cognitive dissonance.

Briefly, evolutionary game theory looks at strategic interactions in a dynamic environment subject to stochastic perturbations. Traditional neoclassical game theory tends to deal with hyper-rational actors and seeks strict deterministic conclusions regarding equilibria. There is a significant repeated game literature, but it is primarily focused on the quest for the Holy Grail of game theory: acquiring a mechanism by which to consistently find unique equilibria. Evolutionary game theory adapts the traditional neoclassical environment to encapsulate a world with persistent stochastic shocks, and often populates that world with agents that lack perfect rationality and perfect information.

The wedding of stochastic shocks to relatively 'dumb' agents in evolutionary game theory can end up selecting the same type of equilibria as hyper-rational agent models. At the same time, it allows us to examine fundamentally different types of questions: how fast does the system reach something resembling equilibria, how robust are these results to variation in the game's parameters, and which equilibria are we more likely to see happen over time? Accordingly, evolutionary game theory presents an apt framework to examine complex social environments.

This section proceeds firstly by introducing the general form of a model which is amenable to a cognitive dissonance framework. Then, we will adapt this model to include the insights captured by dissonance theory, and finally, we will use tools from evolutionary game theory and social network simulations to see if we can find any results that make interesting predictions about human social-political interactions.

3.1 The General Model

The setup of the model borrows heavily from the outline provided in *Individual Strategy and Social Structure* by H. Peyton Young.(34) This model departs from that in the Young text in specifying how agents select who to interact with and the way in which decisions are made as a result of those interactions. As we shall see, using this basis, the model becomes exceedingly complex to work with using traditional methods at any level of sophistication. At the point at which the model becomes almost impossibly complex, we will attempt to move from analytic results to simulation-driven results.

Importantly, both the analytic and simulation models takes place within the same general game. Agents exist on a grid, one agent is chosen to act at a time, it chooses a neighbor to interact with according to some rule, and, upon encountering a neighbor with a different political ideology, it suffers dissonance. The agent may, as a result, change its political ideology so as to reduce dissonance.

This type of system finds its closest parallel in Young's 'Local Interaction' model, in which agents exist on a grid and interact exclusively with a fixed group of local neighbors. In the grid model, agents are situated at the vertex of a graph Γ. The set of m vertices, or 'nodes,' will be denoted by V, and the set of directed edges by E. Each edge $\{i,\ j\}$ has a positive weight w_{ij}, where the weight of the edge is the 'importance' of the interaction. Vertices i and j are neighbors if i and j are linked by an edge, where $\{i, j\} \in E$. The set of all neighbors of a given vertex is denoted by N_i. We assume that Γ has no isolated vertices, ie: $N_i \neq \emptyset$ for all i. In other words, every agent has at least one neighbor.

The *state* of the process is a vector $\mathbf{x} \in X_0^m$ such that $x_i \in X_0$ is player i's current action for each $i \in V$. Denote the set of states by Ξ. In the general game, the payoff to player i through an interaction with player j is given by the function $u_i(x_i, x_j)$. This function does not always need to be specified. In games in which the payoff is specified, the total payoff to i for a given round is provided by the weighted sum of interactions with all of x's neighbors:

$$v_i(\mathbf{x}) = \sum_{j \in N_i} w_{ij} u(x_i, x_j) \tag{3.1}$$

This equation defines a payoff function of an m-person game played each period, which Young calls a *spatial game*. State $\mathbf{x}$ is a *Nash Equilibrium* of the spatial game if for every i, and every $x' \in X_0$,the following condition holds.

$$\sum_{j \in N_i} w_{ij} u(x_i, x_j) \geq \sum_{j \in N_i} w_{ij} u(x', x_j) \tag{3.2}$$

3.2 Spatial Games as Cognitive Dissonance Games

In order to represent cognitive dissonance, we have to specify what actions are available to players, how they choose who to interact with, and the link between the two, in order to represent a dissonance reducing process.

In this version, $x \in [0, \theta]$, where x represents political ideology on a political spectrum ranging from 0 to θ, and θ is a setup parameter of the game used to expand or contract the available 'political space'. Interaction between agents with different values of x causes dissonance in the manner explained in the first chapter. This dissonance effects behavior in two ways; it may result in a change of the choice of x, and in anticipation of dissonance, agents attempt to interact more frequently with other agents possessing values of x closer to their own. This is sometimes called the 'importance' of the relationship to a particular agent and it has a clear parallel to the edges in the preceding section.

However, whereas in the spatial game, edge $\{i, j\}$ represents the non-directional weight of the $i - j$ interaction to both i and j, in the cognitive dissonance version of the game, it represents the *directional* importance of the $i - j$ interaction to i. Usually this importance represents the probability that i, when called upon to act, chooses j to interact with, out of the set of possible neighbors N_i.

In the traditional spatial game, the $\{i, j\}$ edge is symmetric and unchanging, so $w_{ij} = w_{ji}$. This is *not* true for a cognitive dissonance game. While the values of w_{ij} may be equal to w_{ji} by chance, they are representative of two different elements in the game and any similarity in value is not necessary or even expected to occur.

Traditional Spatial Game Weights

$$\begin{tikzcd} I \arrow[r, leftrightarrow, "w_{ij}"] & J \end{tikzcd}$$

Cognitive Dissonance Weights

$$\begin{tikzcd} I \arrow[r, shift left, "w_{ij}"] & J \arrow[l, shift left, "w_{ji}"] \end{tikzcd}$$

When, as a result of dissonance reduction, i reduces the importance of the $i-j$ interaction, this amounts to a unilateral change in w_{ij}; changes in w_{ij} occur independently of w_{ji} which remains unaltered. Note that $\sum_{j \in N_i} w_{ij} = 1$. That is, the sum of all of the weights of i's interaction with its neighbor is equal to 1. We also specify, for now, that in cognitive dissonance games i can only interact with, or 'see', agents immediately bordering it. So the set of neighbors N_i is simply the set of agents who immediately border agent i on the grid.

All cognitive dissonance games have some variant on the following order of moves per round:

1. An i is randomly selected, with all $i \in V$ equally likely to be selected.

2. i allocates a weighting of the importance of each of the bi-lateral relationships amongst her immediate neighbors.

3. i interacts with her neighbors, either bilaterally or as a group and as a result she changes her value of x in order to reduce dissonance.

4. The state of the game $\mathbf{x}$ is updated and the next round begins.

Given that there are two types of variables assigned to each agent, a political ideology choice and the allocation of importance across the set of neighbors, it might seem that the state of the game should capture both elements. However, as allocations of importance are generated each round, and simply derived from x_i

and $x_k \in N_i$ by applying some allocation mechanism, all of the relevant information for determining the state of the game is provided in the list of agent's political ideologies. Accordingly, the state is defined as $\mathbf{x}$, where $\mathbf{x} = x_0, x_1, ..., x_m$. As we shall see, it is the function form of the individual steps which drive the process; different processes lead to various different kinds of equilibria and *very* different kinds of behavior in the evolution of the game.

3.3 The Decision Rule

Central to the cognitive dissonance game is the manner in which agents choose values for x when interacting with neighbors (See step 3 above). In all forms of the game, we make the assumption that the experience of dissonance and the resulting choice of political identification in order to reduce dissonance occurs in one round. It is modeled in two steps, but they both occur in one time period. The alternative would be a system where i experiences dissonance, logs it, and then later uses that record as an input into decision making. This does not add to the workings of the model, while clogging up the gears. Compacting the allocation of importance and the choice of political ideology into one round allows for all of the relevant information of a given state to be restricted to the list of agent's political ideologies, otherwise known as state $\mathbf{x}$.

This paper will use two methods consistent with the theory of dissonance which allow for dissonance from social disagreement to be directly incorporated into the choice function. In the first method, labeled *neighborhood interaction*, agent's weight the importance of each neighbor relationship and make a decision as to their political identification in response to the weighted average of their surrounding neighborhood. In this model, it is as if individual agents experience dissonance as a result of distance from the weighted average political ideology of its local community. The second method, labeled *bilateral interaction* involves agents choosing a neighbor from some distribution and interacting directly with that neighbor. They interact with their environment as if they are bumping into people in their neighborhood, at a rate defined by a distribution created from the weighted allocation of importance of each of the bilateral interactions to i.

Neighborhood Interaction

Neighborhood interaction permits i to evaluate the distance between itself and the weighted average of her neighborhood. A neighbors political identification is given a weight according to an allocation function defined later on. As opposed to the bilateral choice model, this interaction is inherently unilateral in nature, i in a sense takes in the surrounding landscape, and without specifically choosing a neighbor to interact with, optimizes over that community. These steps are as follows:

1. An i is randomly selected, with all $i \in V$ equally likely to be selected.
2. i allocates distribution of importance amongst its neighbors dependent on their values of x according to a specific allocation function.
3. i calculates a weighted average of its neighbors according to these weights.
4. i experiences dissonance resulting from distance from the community as defined by the weighted average, and changes political identification accordingly.
5. The state is updated with i and only i's change in political ideology x'_i and the loop starts again.

Dissonance from Disagreement

In this specification it is helpful to specify a function for the dissonance arising from particular states of the world. In the bilateral choice model, that is not always the case. The first step is to establish a functional form of the dissonance from disagreement with neighbors.[1] The dissonance resulting from ideological distance between the agent under consideration i and a given neighbor j is given by

$$u_i({}_ix_i, x_j) = -(x_i - x_j)^\alpha \tag{3.3}$$

[1] This paper considers dissonance to be functionally equivalent to negative utility, and these terms will be used interchangeably throughout this chapter.

If the distance function assumes a quadratic form, which it will for much of this section, we have the following.

$$u_i(x_i, x_j) = -(x_i - x_j)^2 \tag{3.4}$$

Now that we have a function for the utility from disagreement dissonance, we can insert 3.4 directly into 3.1 to get a function for the total utility to player i from interacting (disagreeing) with all of its neighbors.

$$\begin{aligned} v_i(\mathbf{x}) &= \sum_{j \in N_i} w_{ij} u_i(x_i, x_j) \\ &= \sum_{j \in N_i} w_{ij}[-(x_i - x_j)^2] \\ &= -\sum_{j \in N_i} w_{ij}(x_i - x_j)^2 \end{aligned} \tag{3.5}$$

It is helpful to think of equation 3.5 as the *level* of utility from state $\mathbf{x}$. At this point it is simple to characterize the Nash Equilibrium as the state $\mathbf{x}$ such that, for each i,

$$x_i = \arg\max_{x_i} v_i(\mathbf{x})$$

Solving for x_i yields[1],

$$x_i = \sum_{j \in N_i} w_{ij} x_j \tag{3.6}$$

In mathematical terms, this says what we already understand intuitively. i has no incentive to change x_i when x_i is equal to the weighted value of i's neighbors. Any change in x_i from that spot results in a level of utility lower than when $x_i = \sum_{j \in N_i} w_{ij} x_j$. This is a standard result of a quadratic distance loss function.

Dissonance from Opinion Change

In the previous subsection, individuals are modeled as choosing the utility maximizing(loss minimizing) point on the political spectrum relative to the weighted average of their neighbors. However, this is not the end of the story. From the literature on dissonance theory, it is clear that individuals are reticent to change their opinions. To briefly restate this discussion, assume that i has to

[1]

$$\begin{aligned} x_i &= \arg\max_{x_i} v_i(\mathbf{x}) \\ &= \arg\max_{x_i} - \sum_{j \in N_i} w_{ij}(x_i - x_j)^2 \end{aligned}$$

Taking the first order condition with respect to x_i, and applying the fact that, $\sum_{j \in N_i} w_{ij} = 1$ and thus $\sum_{j \in N_i} w_{ij} x_i = x_i$, yields

$$\begin{aligned} -2 \sum_{j \in N_i} w_{ij}(x_i - x_j) &= 0 \\ \sum_{j \in N_i} w_{ij}(x_i - x_j) &= 0 \qquad \text{(diving by constants)} \end{aligned}$$

Noting that the equation is additive separable and moving one half of the equation to the other side yields,

$$\sum_{j \in N_i} w_{ij} x_i = \sum_{j \in N_i} w_{ij} x_j$$

As w_i are taken as constant in this optimization, we can apply, $\sum_{j \in N_i} w_{ij} = 1$ and thus $\sum_{j \in N_i} w_{ij} x_i = x_i$ such that,

$$x_i = \sum_{j \in N_i} w_{ij} x_j,$$

have some previous reasoning for the initial choice of x. While, the choice of x may be assumed to be random for simulation and analytic purposes, in the real world, individuals have good reasons for choosing a particular place on the political spectrum. The phrase 'good reasons' is another way of saying that there are relatively important cognitions which corresponds to any agent's current choice of political identification. Just as the individual in the previous section minimized the loss from disagreement with neighbors, we have to assume that the initial position of x minimized the dissonance with respect to other cognitions.

Festinger states that any individual's current beliefs are those which are least dissonant with the other cognitions held by that individual. Applying this to our model, we can say that an agent's current x is the x which is most consonant with the various pre-existing cognitions held by the agent. Thus, any change in political identification causes some dissonance with all of the previously held cognitions which led to the previous choice of x. In short, change is costly. One does not move from one side of the political spectrum to the other overnight; the obvious betrayal of our old selves would be too costly. As a result, change happens gradually.

When modeling cognitive dissonance; it is essential to remember this point. There is in fact a balance between the urge to reduce tension with those around you and the pressure to stick to your initial position.

This second pressure can be modeled in a similar way to the first, where dissonance is the distance between the new chosen value, x_i', and the previous state value, x_i

$$u_i(x_i', x_i) = -(x_i' - x_i)^{\beta}. \tag{3.7}$$

Once again, focusing on quadratic loss,

$$u_i(x_i', x_i) = -(x_i' - x_i)^2 \tag{3.8}$$

The previous subsection, on dissonance from disagreement, established that if the only thing individuals worried about was disagreement with their neighbors, then they would pick their political position equal to the weighted average of their neighbors. That is the highest *level* of utility they can obtain. However, they cannot jump to that point immediately because to do so would impose significant loss in terms of moving away from their reference position.

A quick note should be made on the temporal nature of this problem. In combining these two costs, agents do not consider future periods: there is perfect myopia. There is no discounting of the future level of dissonance from disagreement and thus there is no tradeoff between moving today and being in a better position tomorrow and for every day thereafter. This is not a dynamic optimization problem. Agents only consider the dissonance they incur today as result of neighborhood disagreement and opinion change. This is in part because agents have no guarantee of where everyone else will be tomorrow, and also to aid in the modeling.

Thus, it is helpful to consider equations 3.7 and 3.8 as a payment that agents have to pay in order to agree with their neighbors. This payment corresponds to the discomfort of knowing that they are now in disagreement with their former selves. This payment is a one-off fee for changing position, it can be thought of as a flow, whereas the level of disagreement is a stock. Re-label equation 3.8 as the opinion change fee, c;

$$c_i(\mathbf{x}', \mathbf{x}) = u_i(x_i', x_i) = -(x_i' - x_i)^2 \tag{3.9}$$

The integration of the two considerations, the dissonance from disagreement and the dissonance from opinion change, will give us our *Neighborhood Interaction* choice rule. As both functions are negative and concave in x_i, the best i can do is the maximum of the total utility which is also the minimum of the two dissonance functions combined. In a more general form, the two functions can be combined in a linear fashion, which later allows for different weightings of the two elements:

$$\max_{x_i'} \omega[v_i(\mathbf{x}')] + (1 - \omega)[c_i(\mathbf{x}', \mathbf{x})] \tag{3.10}$$

For the rest of this section we assume $\omega = 0.5$, that is they take equal weight:

$$\max_{x_i'} v_i(\mathbf{x}') + c_i(\mathbf{x}', \mathbf{x}) \tag{3.11}$$

Expanding equation 3.11 and taking the first order condition yields the following result:[1]

$$x_i' = \frac{x_i + \sum_{j \in N_i} w_{ij} x_j}{2} \tag{3.12}$$

This provides the standard conclusion that, when agents care equally about dissonance from disagreement and dissonance from changing opinion, the loss minimizing or utility maximizing choice of x_i' is equal to half of the distance between the old position on the political scale and the weighted average of the neighbors. Realistically, this probably places too much weight on reducing dissonance for disagreement, and results in changes of political disagreement that are too extreme. This can be modified such that agents care relatively more about dissonance from opinion change, by setting $\omega > 0.5$.

It is also important to note that while this section has mainly referred to the w_{ij} variable as the directional importance of the $i - j$ interaction to i, the distribution of w_{ik} for $k \in N_i$ can and is considered a probability distribution of interaction. Instead of actually interacting with neighbors, i is optimizing over the expected value of its neighbors' political identification. It can be shown that this form, and the form in the neighborhood model, are functionally equivalent. This is important in considering the next model and for ease of terminology throughout the paper.

[1]

$$\begin{aligned}
\frac{d}{dx_i}[v_i(\mathbf{x}') + c_i(\mathbf{x}', \mathbf{x})] &= 0 \\
\frac{d}{dx_i}[-(\sum_{j \in N_i} w_{ij}(x_i' - x_j)^2) + (-(x_i' - x_i)^2)] &= 0 \\
\sum_{j \in N_i} w_{ij}(x_i' - x_j + (x_i' - x_i)) &= 0 \\
\sum_{j \in N_i} w_{ij} x_i' + x_i' &= x_i + \sum_{j \in N_i} w_{ij} x_j \\
x_i' &= \frac{x_i + \sum_{j \in N_i} w_{ij} x_j}{2}
\end{aligned}$$

Bilateral Interaction

The second specification considers *bilateral* interaction between a randomly selected agent, i, and one of its neighbors, j. The bilateral choice model specifies that i choose a neighbor by drawing from some predefined distribution and interacts with that neighbor j bilaterally but suffering dissonance unilaterally. Here, edge $\{i, j\}$ represents not only the relative importance to i of the $i-j$ interaction but also the probability that i, when selected in step 1, will choose j to interact with in step 2. The specification that $\sum_{j \in N_i} w_{ij} = 1$ explicitly states that i, when selected, will choose with probability 1 to interact with one of its neighbors.

The bilateral version of the cognitive dissonance game does not require a specific payoff function; it suffices that in any situation in an $i - j$ interaction where i is called upon to reduce dissonance, any $x_i \neq x_j$ will provide enough dissonance to compel i to choose a x'_i, such that $|x'_i - x_j| \leq |x_i - x_j|$, but also constrained such that $|x'_i - x_i| <= 1$. In other words, x has to be somewhere between 0 and θ, but in discrete increments along the natural number line, and i can only move up or down at most one unit in trying to get closer to j. When $x_i = x_j$ then obviously no movement is necessary.

The bilateral choice model of dissonance expands the steps of the game in the following manner:

1. An i is randomly selected, with all $i \in V$ equally likely to be selected.
2. i determines a distribution to apply to her neighbors according to some allocation mechanism, dependent on their values of x.
3. i draws from that distribution a neighbor with whom to interact, j.
4. "Interaction" occurs between i and j, and i moves one step closer to j if necessary.
5. The state is updated with i and only i's change in political ideology x'_i and the loop starts again.

This form of modeling is both simpler in form than that in section **??**, and, under certain settings, provides similar predictions about how agents interact.

As a result of this tractability, the second method will be the primary workhorse model of this paper. The potential for further research exists in attempting to integrate neighborhood interaction into the analytic framework introduced later in this paper. However, in providing a different means by which the system can acquire similar convergent tendencies, it supports the 'dumber' but more tractible bilateral choice framework.

Mechanisms for Allocating Importance/Probability

In step 2 of the original cognitive dissonance formulation, and in both the neighborhood choice and bilateral interaction games, how i allocates the weights of importance/probability amongst its neighbors is left undefined. This section provides two variants on how this can be done, both consistent with the principles outlined in the cognitive dissonance literature. This section speaks exclusively of probability weights for simplicity of communication, as it has been shown above that they are functionally equivalent representations in either game.

The probability weights w_{ik} for $k \in N_i$ are dependent on the relative distance of x_k to x_i However, suffice it to say, that one of the central tenets of cognitive dissonance will always apply. That is, that individual i will prefer to interact with neighbors with relatively close x values, and given the choice between two neighbors with different values of x it will always be more likely (ie: with $p > 0.5$) that i will interact with the neighbor with a value of x, closer to her own (in absolute values). Two methods are provided below for assigning those probability weights:

1. *Closest Neighbor*: A simple indicator function in which i conducts a local search for the neighbor with closest x and assigns the probability of interacting with that neighbor equal to one and all others 0.

$$w_{ij} = I(x_i, xj, x_{-j \in N_i}) \tag{3.13}$$

$$w_{ij} = \begin{cases} 1 & \text{if } |x_i - x_j| < |x_i - x_{-j}| \ \forall \text{j} \in N_i \\ 1/n & \text{if there are } n-1 \text{ other neighbors equally close} \\ 0 & \text{if } \exists x_{-j \in N_i} \text{ where } |x_i - x_{-j}| < |x_i - x_j| \end{cases}$$

2. *Quantal Choice*: A multinomial logit mechanism, or "quantal mechanism" which gives neighbors with closer values of x higher 'importance', along with a higher probability of interaction (w_{ij}). In this manner, neighbors who are, in political terms, far away are not very important, and not often interacted with, but there still exists a small chance that an interaction will occur.

$$w_{ij} = \frac{e^{-|x_i - x_j|}}{\sum_{k \in N_i} e^{-|x_i - x_k|}} \tag{3.14}$$

3.4 A Simple Game

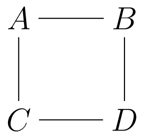
The previous section explored two kinds of choices agents make: the choice of who to interact with and the choice of what to do in response to an interaction that causes dissonance. This section puts those elements together in a simple version of the game.

To start, we will use the most simplified version of a cognitive dissonance game. A bilateral interaction with a closest neighbor allocation mechanism. While the various permutations of game types may seem rather complicated, this version highlights a simple formulation of the game. An agent i is randomly selected, it conducts a local search among its neighbors for the agent with the closest political identification, finding agent j, and interacts with that neighbor. If $x_i \neq x_j$, agent i chooses to move one-step closer to j.

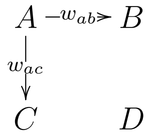
To further simplify, we can set $\theta = 1$. That is, there are only two choices of political identification, say 0 = Democrat and 1 = Republican (to use a prominent example involving two political parties). As $\theta = 1$, the political spectrum is truncated to a binary range and there are only two potential values for political identification. In any bilateral interaction with disagreement and hence dissonance, the actor called upon to choose political identification will choose to flip sides to remove the dissonance. Clearly this is an unreasonable assumption for a general case model, but it is only a parameter constraint. We can later demonstrate how expanding the political space to a more reasonable range does not alter the fundamental understanding of the behavior of the system (although it will greatly change outcomes).

For analytical purposes, if any given i has two or more neighbors who are equally close, we say that i picks one randomly to interact with. Understanding that it is i and only i that will be called upon to change x as a result of bilateral interaction, the resulting decision will be the same regardless of which one of the closest neighbors it picks to interact with. This helps reduce the complexity and the potential for mistakes in the coding for the simulations.

The first geometric arrangement under scrutiny will be a box neighborhood. In the box neighborhood, there are four players and every player has two neighbors, a horizontal neighbor and a vertical neighbor. There are no diagonal connections.



To use the example above, $N_A = [B, C]$, recalling that N_i is a mathematical representation of the set of neighbors available to a given i. That is, A can only interact with, or 'see', B and C. When choosing weights for neighbors, A is only choosing the weights for w_{ab} and w_{ac}, and $w_{ab} + w_{ac} = 1$.



We can already learn a lot about the dynamics of a social system defined by the cognitive dissonance reducing behavior expounded above. In the box model, there are only $2^4 = 16$ potential states, $\mathbf{x}$ in the state space X. That is, there are four nodes and each node can be either 0 or 1. Within these sixteen potential states there are six 'types', which correspond to different possible strategically distinct arrangements of agents. These six different arrangements are enumerated as follows:

(I) All zeros:

$\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right)$

(II) Three zeros:
$\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right)$
(III) Two zeros (segregation):
$\left(\begin{smallmatrix}1&1\\0&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1\\0&1\end{smallmatrix}\right)$
(IV) Two zeros (diagonal):
$\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$
(V) One zero:
$\left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right)$
(VI) No zeros:
$\left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$

Given the simplicity of the model, we can immediately enumerate the static Nash Equilibria of this game. From this, we can generate a first guess as to the stable states of the game. There are only two forms of static Nash Equilibria: perfect homogeneity and perfect segregation. Perfect homogeneity occurs at I and VI, with all zeroes and no zeroes, respectively. Perfect segregation occurs only at III, with two zeroes and two ones, connected. We can find these equilibria directly from the definition of a Nash Equilibrium. From Nash, we know that any situation in which at least one player can make themselves better off by changing strategy will not be stable. Obviously any state in which players change voluntarily cannot be a Nash; this is relatively straight forward. In the bilateral interaction model, agents have an incentive to change their political identification whenever they encounter another agent with different political identification. Through the closest-neighbor allocation mechanism, in a sense agents already pre-screen their interactions. As a result, any given agent will only interact with someone with a different political identification when they cannot find at least one neighbor who shares the same political identification. Put another way, a necessary condition for stability is that all agents have at least one neighbor whom they share a political identification with, and all agents only place positive interaction weight on agents with whom they share a political identification.

Recalling equation **??**, we have that state $\mathbf{x}$ is a *Nash Equilibrium* of the spatial game if for every i, and every $x' \in X_0$,

$$\sum_{k \in N_i} w_{ik} u(x_i, x_k) \geq \sum_{k \in N_i} w_{ik} u(x', x_k) \tag{3.15}$$

We have not specified a functional form of the utility function for bilateral interaction, but we know how any functional form has to behave. That is, any distance between x_i and x_k provides enough dissonance to compel the agent in question to change their political identification to move closer to k. We can thus provide an extremely simple utility function which does just that: $u_i(x_i, x_k) = -|x_i - x_k|$ and inserting it into eq. **??**, we have;

$$\begin{aligned} -\sum_{k \in N_i} w_{ik}|x_i - x_k| &\geq -\sum_{k \in N_i} w_{ik}|x'_i - x_k| \\ \sum_{k \in N_i} w_{ik}|x_i - x_k| &\leq \sum_{k \in N_i} w_{ik}|x'_i - x_k| \end{aligned}$$

Recall the closest neighbor weighting allocation function. Where agents find the closest neighbor $j \in N_i$ and set $w_{ij} = 1andw_{i-j} = 0$, applying this to the above equation we have;

$$\begin{aligned} \sum_{k \in N_i \bigcap \neq j} w_{ik}|x_i - x_k| + w_{ij}|x_i - x_j| &\leq \sum_{k \in N_i \bigcap \neq j} w_{ik}|x'_i - x_k| + w_{ij}|x'_i - x_j| \\ \sum_{k \in N_i \bigcap \neq j} 0 \bullet |x_i - x_k| + 1 \bullet |x_i - x_k| &\leq \sum_{k \in N_i \bigcap \neq j} 0 \bullet |x'_i - x_k| + 1 \bullet |x'_i - x_k| \\ |x_i - x_j| &\leq |x'_i - x_j| \end{aligned}$$

This occurs if and only if $x_i = x_j$. Put another way, the best any one agent can do is have the same political identification as the selected neighbor. Any situation in which an agent can make themselves better off by moving closer to their neighbor cannot be a Nash equilibria. Agents will *always* choose to change x when they interact with another agent possessing a different value of x and this *only* occurs when an agent cannot find a neighbor with the same value of x. The stable situation is when every agent can find a neighbor with the same value of x, in the absence of stochastic shocks. This gives us a necessary condition for examining the first order stability of states in a stochastic environment. As we shall see later in the simulation section, this result of perfect segregation defined by every agent having at least one neighbor of similar political ideology becomes very important as we increase the potential political space and/or the height and width of the grid to include larger numbers of agents.

Using this one step of logic, we have a simple criteria for defining a first guess of equilibrium states. In any stable unperturbed state, all agents must have at least one neighbor with which they have a common political ideology, and agents only interact with neighbors who have the same political ideology as themselves. This is a generic form of stable state, and we can easily conceptualize two sub-classes which are our two forms, states in which all agents share the same political identification (perfect homogeneity) and states where there are at least two distinct groups of self-contained homogeneity (perfect segregation). While this result holds for any grid, applying it to the 2x2 simple grid game, we can categorize the enumerated states. Of all of the types of states enumerated above, only (I), (III) and (VI) fall within this criterion.

3.5 Applications of Markov Chain Theory

Even before stochastic errors are introduced into the model, a lot can be said about the behavior of this system. We have specified a type of game where the state of the game at any point is defined as the political ideology of all players on the grid. Recall the state is defined as $\mathbf{x}$, where $\mathbf{x} \in X_0^m$ and X_0^m is the set of all potential starting states of the game. The behavior of this game is neatly described as a process of moving between various states over time with predefined probabilities according to the specifications of the game. This type of system is highly amenable to analysis using Markov Chain Theory.

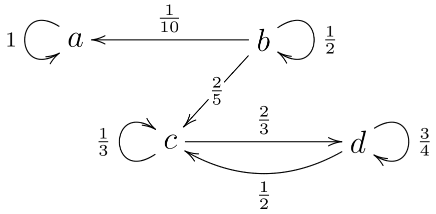
In order to take the next step, we can draw from the relevant bits of Chapter 3.3 of Young (1998).[1] We have already defined the state space as X, and the

[1] The terminology and definition of terms will be adapted from this text.

current state of the system being $\mathbf{x}$. From Markov Chain Theory, we can define a discrete-time Markov process on this state X which specifies the probability of transiting from any state in X, $\mathbf{x}$ to any other state $\mathbf{x}'$, and any time t as $P_{\mathbf{xx}'}(t)$. This is the transition probability of moving to state $\mathbf{x}'$ at time $t+1$ conditional on being in state $\mathbf{x}$. In the cognitive dissonance theory, these transition probabilities are independent of t and thus the process is time homogeneous ($P_{\mathbf{xx}'} = P_{\mathbf{xx}'}(t)$).

We say that a state $\mathbf{x}'$ is *accessible* from state $\mathbf{x}$ if there is a positive probability of moving from $\mathbf{x}$ to $\mathbf{x}'$ in a finite number of periods. This is written as $\mathbf{x} \to \mathbf{x}'$. States $\mathbf{x}$ and $\mathbf{x}'$ *communicate* if you can get from one to the other for both, written as $\mathbf{x} \sim \mathbf{x}'$. These communicating states are said to partition the space into equivalence classes, which in turn are called *communication classes.* Further, a communication class has a another subdivision called a *recurrent class.* Recurrence classes include states which can communicate with each other but cannot access states outside of the recurrent class. That is, once you fall into a recurrent class, you cannot get out; only cycle inside, if at all. A singleton recurrent class is called a *absorbing* state.

After this deluge of terminology, it may be helpful to look at a trivial non-relevant example in order to clarify these matters before going forward.



In this example system, we can see that it is possible to go from c to d and vice versa, and that the system can go from b to either a or c. However, it is not possible to get to b from any other state, nor is it possible to get from $c-d$ to a or vice versa. Applying the terminology from the previous paragraph, we have the following: a is accessible from b but they do not communicate. c and d communicate and are a communication class together. b is a transient state, and a is an absorbing state. In the long run, the chance that the process is in any given state is dependent on the initial state of the system. Thus we say that this process is *path dependent* and *nonergodic.*

Now, returning to the game in question with our new terminology in hand, it helps at this point to look at the transition probabilities of our simple system without stochastic shocks. Recall that each node is selected at random with equal probability, and each agent selected looks for the neighbor with the closest value to interact with. Change only occurs in this simple model when agents are forced to interact with agents of a different type, that is a 0 has to interact with a 1 or vice versa. In the absence of shocks, this only occurs when both neighbors are of the other type. Immediately we can see this only occurs with states of type (II), (IV) and (V). The transition probabilities are as follows:

$$\begin{tikzcd} I \arrow[loop left, "1"] \end{tikzcd}$$

This says that with probability one, in the absence of perturbations, state I, or $\begin{smallmatrix}0&0\\0&0\end{smallmatrix}$, remains in state I. This agrees with our earlier work that suggests that state I is a Nash Equilibria. I is an absorbing state in this version as it does not communicate with any other state.

$$\begin{tikzcd} I & II \arrow[l, "1/4"'] \arrow[loop right, "3/4"] \end{tikzcd}$$

This explains that state II, one of, $\begin{smallmatrix}1&0\\0&0\end{smallmatrix}$,$\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$, $\begin{smallmatrix}0&0\\1&0\end{smallmatrix}$, $\begin{smallmatrix}0&0\\0&1\end{smallmatrix}$ will move to state I with probability $\frac{1}{4}$ and stay the same with probability $\frac{3}{4}$. In the case that the odd node out is selected, and is forced to interact with a player with whom it disagrees, it will always choose to reduce dissonance by changing from 1 to 0 in ideology. The odd node out is selected to act with probability $\frac{1}{4}$. Every other node has at least one neighbor with an $x = 0$ and thus there will be no change with probability $\frac{3}{4}$. Also, this diagram shows us that I is accessible from II.

$$\begin{tikzcd} III \arrow[loop left, "1"] \end{tikzcd}$$

State III will never move to a different state as a result of perfect segregation. It is a singleton recurrent class, but is not accessible from any other states.

$$\begin{tikzcd} II & IV \arrow[l, "1/2"'] \arrow[r, "1/2"] & VI \end{tikzcd}$$

State IV always results in a change to a different state as every node has two neighbors of different political identification, resulting in a move to a state of either three 1s or three 0s. With probability $\frac{1}{2}$ a node with value 1 will be selected and the process will transition to state II, and similarly, with probability $\frac{1}{2}$ a node with value 0 will be selected and the process will transition to state V. State IV is not accessible from any other state and transitions to other states with positive probability and thus is a transient state. However, it is possible at this point to go from IV→ II→ I. Thus I is also accessible from state IV.

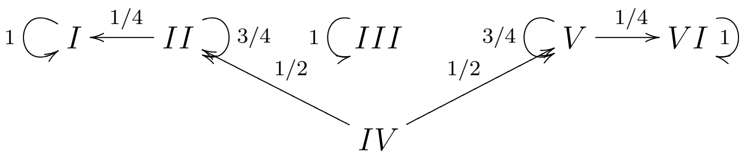
$$\begin{tikzcd} V \arrow[loop left, "3/4"] \arrow[r, "1/4"] & VI \end{tikzcd}$$

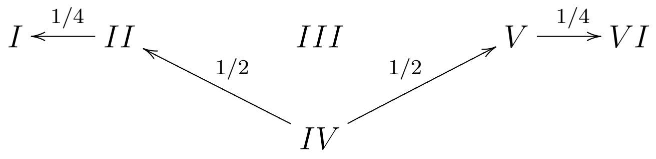
Mirror image of the dynamics of state type II, this time transitioning to state VI.

$$\begin{tikzcd} VI \arrow[loop right, "1"] \end{tikzcd}$$

Mirror image of the dynamics of state type I. Another absorbing state accessible from V and IV.

Combining all the individual diagrams above into one diagram, we have the following two diagrams. For ease of use, in the second diagram, the probabilities of staying in the same state have been removed. The probability of staying in the same state is just one minus the probability of moving to another state, then removing those arrows and keeping this in mind.





The dynamics for all 16 states are rather more complicated, but every state of a given type has exactly the same dynamics, although states within the same types will lead to different states in other types as well. As we will primarily be concerned with the most efficient way of moving from one recurrent class to another, this simplification does not detract at all from our understanding of the system.

From this simple analysis we have two immediate conclusions. States I, III and VI are singleton recurrence classes, but III is inaccessible from any other state. That is, you cannot go from any of them to any other state. States II, IV and V are transient states, not in any recurrent class, meaning the long run limit distribution of the system places zero weights on the state being in any of those states. Following from the literature we will label states I, III and VI as recurrence classes E_1, E_2 and E_3, respectively.

In this game, as stated above, the long run location of the system depends on the initial conditions; that is, it is path dependent. In this system, with no stochastic errors, the process partitions the state space X into different communication classes. If the state begins in the recurrent class III, it will remain there in perpetuity. If the system begins at state II in time $t = 0$, at time period $t > 0$ the system will be in state I with probability $1 - (3/4)^t$. The only other options is for the state to remain in II with probability $(3/4)^t$. As $t \to \infty$, the probability that the process is in II becomes vanishingly small and the probability that the process is in state I approaches 1. Similarly for state V transitioning to state

VI. The only real uncertainty in the entire model is where a process starting in state IV will end up. A process beginning in state IV will, in the first turn, always transition to either state II or state V. Then, in accordance with the system dynamics defined above, the process will eventually fall into their respective singleton recurrent classes: I or VI.

In order to generalize this analysis, we say that given an initial state $\mathbf{x}^0$, for time periods $t > 0$, let $\mu^t(\mathbf{x}|\mathbf{x}^0)$ be the relative frequency with which state z is visited in the first t periods. As $t \rightarrow \infty$ converges 'almost surely' to a probability distribution $\mu^t(\mathbf{x}|\mathbf{x}^0)$, labeled the asymptotic frequency distribution of the process conditional on $\mathbf{x}^0$(25). As Young states, we can say that the process selects states on which $\mu^t(\mathbf{x}|\mathbf{x}^0)$ puts positive probability. As the system is path dependent, the resulting singleton recurrent class that the states selected to have positive probability are wholly dependent on the initial state $\mathbf{x}^0$, this system is nonergodic. It is also an uninteresting result. In order to learn something substantial about this system, we need to introduce stochastic errors. *Introducing Stochastic Concepts*

The most important distinction between traditional game theory and the evolutionary sort is the placement of this type of reasoning in a dynamic, shifting environment. Evolutionary game theory is not simply another mechanism for equilibrium selection, but rather a different approach to evaluating interactions that puts more focus on the how questions. How long does it take to get to equilibrium and how vulnerable are particular states to random fluctuations? Intuitively, we understand that an equilibrium in which one error made by a single agent has the capability to unravel the entire system is not a particularly 'good' equilibrium. For our purposes we look at that issue in terms of robustness. We have already sketched a brief outline of what a perfectly stable cognitive dissonance game looks like. A bilateral choice game with a closest neighbor allocation mechanism, with no errors, upon reaching a Nash Equilibrium of perfect homogeneity or perfect segregation, is entirely stable. No agents can benefit from changing their political identification, and as a result, the allocation weights remain constant.

On the other hand, consider a system bound by the same rules, but one in which agents make error with high probability. Depending on the setting of the frequency of errors, this system is random at best and potentially worse than random. That is, when errors are specifically non-optimal choices, then agents making errors with higher than 0.5 probability will be worse than random in terms of getting to stable states. If such a system happens to be perfectly homogeneous or perfectly segregated, it will immediately fall out of that equilibrium on the next turn as agents are always choosing from the non-optimal range of choices, where the optimal maintains the current state.

For this paper, the interesting part is what we learn by adding this stochastic element to the model. Firstly, how robust are equilibria to error, and how important is decision error as compared to allocation error in determining robustness? Secondly, does adding error lead to specific types of equilibria and how does this play out over the evolution of the system? That is, does a stochastic element favor homogeneous states or segregated states? This becomes of much greater concern as we expand the state space and the political ideology space.

In order to introduce the concept of stochastic stability, we can alter choice rules to include some 'errors' in judgement. As decisions are made in two steps in any cognitive dissonance model, there are two corresponding types of error: errors in deciding who to interact with(labeled neighbor errors) and errors in deciding what to do as a result of that interaction(labeled choice errors). Note that the quantal allocation mechanism already has probabilistic error built into it and as such, does not need updating for this section.

Agents 'misjudge' their neighbor's political identifications with probability δ and subsequently randomly choose a neighbor who is *not* the closest to them politically. A neighbor error excludes all neighbors who are of the same ideology of the closest. So in the case of four neighbors, two of which are the same and closest to the agent in question. In the case of a neighbor error, the agent will choose randomly one of the two neighbors who do not have the closest ideology to interact with. In the analytic version of the model, all agents have only two neighbors, meaning they choose the closest neighbor with probability $(1-\delta)$ and they choose the wrong neighbor with probability δ. Later, in the simulations, players have up to four neighbors.

Agents also make errors in choosing political ideology with probability ε. That is, they randomly choose a political identification from the set of available choices, excluding the correct choice, with probability ε and they follow their correct choice as defined by choice function with probability $(1-\varepsilon)$.

This entirely changes the dynamics of the system. Whereas without errors, the state space was divided into separate communication classes, in this model, it is possible to transition from any state $\mathbf{x}$ to any other state $\mathbf{x}^0$ with positive probability in a finite number of periods. In this process, the entire state space X is one communication class, and the process is said to be *irreducible.* Adapting similar notation from the previous section, here are the transition probabilities: $P_{\mathbf{x}\mathbf{x}'}$ for the states in question (for clarity, we leave out the transition probabilities of remaining in the same state, as it clouds the diagram and will be put in later):

State I:

$$\begin{tikzcd} I \arrow[r, "\varepsilon"] & II \end{tikzcd}$$

This says that with probability ε, state I transitions to state II. This is simply the summation of the probability any one node, selected with probability 1/4 will make a political identification error $4 * (\frac{1}{4}\varepsilon)$.

State II:

$$\begin{tikzcd} I & II \arrow[l, "\frac{1}{4}(1-\varepsilon)"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"] \arrow[dr, "\frac{1}{4}\varepsilon"'] & III \\ & & IV \end{tikzcd}$$

We can get these transition probabilities as follows. Using an example state 'IIe' $= \begin{smallmatrix}1&0\\0&0\end{smallmatrix}$, and labeling the individual nodes $= \begin{smallmatrix}1&0\\0&0\end{smallmatrix} = \begin{smallmatrix}a&b\\c&d\end{smallmatrix}$, where in this example state $a=1, b=0, c=0$ and $d=0$. In order to move from state IIe to state I$=\begin{smallmatrix}0&0\\0&0\end{smallmatrix}$, we simply need to find the ways that a can go from a 1 to a 0. This occurs when a is selected, which occurs with probability $\frac{1}{4}$, and a does *not* make a choice error, which occurs with probability $(1-\varepsilon)$. When a is selected to act, the existence of neighbor errors is irrelevant, as a is surrounded by players of a different ideology and thus will always interact with a player of type 1. Thus the total probability of moving from state II to state I in this case is $\frac{(1-\varepsilon)}{4}$. Note, these transition probabilities are the same for *any* state of type II. So the other states of type II, $\begin{smallmatrix}0&1\\0&0\end{smallmatrix}$, $\begin{smallmatrix}0&0\\1&0\end{smallmatrix}$ and $\begin{smallmatrix}0&0\\0&1\end{smallmatrix}$, all are just as likely to transition to the one state with four 0s, state I. This is how we can justify the diagrammatical simplification.

In order to move to a state III from state IIe we can do the same operation but in the other direction. The relevant states of type III that our example state IIe has the possibility of transiting the state to are as follows: $\begin{smallmatrix}1&1\\0&0\end{smallmatrix}$ or $\begin{smallmatrix}1&0\\1&0\end{smallmatrix}$.

A move to a state of type III can only occur if and when either b or c converts from type 0 to type 1. Nodes b or c are selected to act with a cumulative probability $\frac{1}{2}$, corresponding to twice the probability that any one node is selected. There are two ways in which b or c can change from 0 to 1.

1. b or c makes an error in choosing who to interact with and choose a to interact with instead of d (neighbor error). This occurs with probability δ. Now, given that they are are interacting with a neighbor who has a different political identification, the optimal choice of political variable is for them to change from 0 to 1. This occurs with probability $1 - \varepsilon$. Note that it is possible for b or c to choose to interact with a but make an error in choice, and stay at 0!

2. b or c does not make a neighbor error, which occurs with probability $(1 - \delta)$, but they do make a choice error and switch from 0 to 1, in spite of interacting with d, occurring with probability ε.

Simplifying $\frac{1}{2}(1-\delta)\varepsilon + \frac{1}{2}\delta(1-\varepsilon)$ leaves us with $\frac{1}{2}(\varepsilon+\delta) - \delta\varepsilon$ as the transition probability of moving from II to III.

A third possibility is moving from state IIe to state IV= $\begin{smallmatrix}1&0\\0&1\end{smallmatrix}$. This requires player d to be selected, and then make a choice error and change from 0 to 1, which occurs in combination with probability $\frac{1}{4}\epsilon$.

Finally, the probability of remaining in state IIe is just one minus the the sum of the probabilities of moving away from IIe. This is $\frac{3}{4} - \frac{1}{2}(\varepsilon + \delta) + \delta\varepsilon$. Again, each transition probability is the same regardless of which state of type II occurs due to symmetry.

State III:

$$\begin{tikzcd} II & III \arrow[l, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"] & V \end{tikzcd}$$

The transition probabilities to get to states II and V follow a logic similar to that above. Using the example $\begin{smallmatrix}1&1\\0&0\end{smallmatrix} = \begin{smallmatrix}a&b\\c&d\end{smallmatrix}$, to get from III to II we need either a or b to make a*either* a neighbor error or a choice error. To get to state V we need c or d to do the same. The process remains in state III with probability one minus the chance of moving to II or V. Simplified, this is $1-[\delta+\varepsilon-2\delta\varepsilon]$. State IV:

$$\begin{tikzcd}
II & IV \arrow[l, "\frac{1}{2}(1-\varepsilon)"'] \arrow[r, "\frac{1}{2}(1-\varepsilon)"] & VI
\end{tikzcd}$$

In state IV every player, if chosen, would prefer to change x. Accordingly the process remains in state IV with probability: $1-2*[\frac{1}{2}(1-\varepsilon)]=\varepsilon$.

The diagrams for states V and VI are mirror images of states II and I, respectively. State V:

$$\begin{tikzcd}
III & & V \arrow[ll, "\frac{1}{2}(\epsilon+\delta)-\delta\varepsilon"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"] \arrow[dl, "\frac{1}{4}\varepsilon"] & VI \\
& IV & &
\end{tikzcd}$$

With $V \to V$ again being $(\frac{3}{4}-\frac{1}{2}(\varepsilon+\delta)+\delta\varepsilon)$. State VI:

$$\begin{tikzcd}
V & VI \arrow[l, "\varepsilon"']
\end{tikzcd}$$

The process remains in state V with probability $(1-\varepsilon)$.

Finally, combining them into one large diagram:

$$\begin{tikzcd}
I \arrow[r, "\varepsilon"'] & II \arrow[l, bend right, "\frac{1}{4}(1-\varepsilon)"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"'] \arrow[dr, "\frac{1}{4}\varepsilon"] & III \arrow[l, bend right, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"'] & V \arrow[l, bend right, "\frac{1}{4}(1-\varepsilon)"'] \arrow[r, "\frac{1}{4}(1-\varepsilon)"'] \arrow[dl, bend left, "\frac{1}{4}\varepsilon"] & VI \arrow[l, bend right, "\varepsilon"'] \\
& & IV \arrow[ul, bend left, "\frac{1}{2}(1-\varepsilon)"] \arrow[ur, "\frac{1}{2}(1-\varepsilon)"] & &
\end{tikzcd}$$

Put into one large transition matrix, we have the following:

$$P = \left[\begin{array}{cccccc} 1-\varepsilon & \varepsilon & 0 & 0 & 0 & 0 \\ \frac{1}{4}(1-\varepsilon) & \frac{3}{4}-\frac{1}{2}(\varepsilon+\delta)+\varepsilon\delta & \frac{1}{2}(\varepsilon+\delta)-\varepsilon\delta & \frac{1}{4}\varepsilon & 0 & 0 \\ 0 & \frac{1}{2}\delta(1-\varepsilon)+\frac{1}{2}(1-\delta)\varepsilon & 1-(\delta+\varepsilon-2\delta\varepsilon) & 0 & \frac{1}{2}\delta(1-\varepsilon)+\frac{1}{2}(1-\delta)\varepsilon & 0 \\ 0 & \frac{1}{2}(1-\varepsilon) & 0 & \varepsilon & \frac{1}{2}(1-\varepsilon) & 0 \\ 0 & 0 & \frac{1}{2}(\varepsilon+\delta)-\varepsilon\delta & \frac{1}{4}\varepsilon & \frac{3}{4}-\frac{1}{2}(\varepsilon+\delta)+\varepsilon\delta & \frac{1}{4}(1-\varepsilon) \\ 0 & 0 & 0 & 0 & \varepsilon & 1-\varepsilon \end{array}\right.$$

We can calculate the inverse of the first matrix , as (with $d=\delta, e=\epsilon$):

$$P^{-1} = \left[\begin{array}{cccccc} -1/4\,\frac{s(8)}{s(1)} & \frac{s(2)e}{s(1)} & \frac{(2\,ed-d-e)e}{s(3)} & -1/2\,\frac{(1-e-d+2\,ed)e}{s(3)} & \frac{s(4)e}{s(1)} & -1/4\,\frac{s(4)e}{s(1)} \\ -1/4\,\frac{s(2)}{s(5)} & \frac{s(2)}{s(5)} & \frac{2\,ed-d-e}{s(6)} & -1/2\,\frac{1-e-d+2\,ed}{s(6)} & \frac{s(4)}{s(5)} & -1/4\,\frac{s(4)}{s(5)} \\ -1/4\,\frac{2\,ed-d-e}{s(6)} & \frac{2\,ed-d-e}{s(6)} & \frac{1-e-d+2\,ed}{s(6)} & -1/2\,\frac{2\,ed-d-e}{s(6)} & \frac{2\,ed-d-e}{s(6)} & -1/4\,\frac{2\,ed-d-e}{s(6)} \\ -1/4\,s\,(7) & s\,(7) & \frac{2\,e^2d-3\,ed+d-e^2+e}{s(6)e} & -1/2\,\frac{2\,e^2d-11\,ed+5\,d-e^2+6\,e-3}{s(6)e} & s\,(7) & -1/4\,s\,(7) \\ -1/4\,\frac{s(4)}{s(5)} & \frac{s(4)}{s(5)} & \frac{2\,ed-d-e}{s(6)} & -1/2\,\frac{1-e-d+2\,ed}{s(6)} & \frac{s(2)}{s(5)} & -1/4\,\frac{s(2)}{s(5)} \\ -1/4\,\frac{s(4)e}{s(1)} & \frac{s(4)e}{s(1)} & \frac{(2\,ed-d-e)e}{s(3)} & -1/2\,\frac{(1-e-d+2\,ed)e}{s(3)} & \frac{s(2)e}{s(1)} & -1/4\,\frac{s(8)}{s(1)} \end{array}\right]$$

with subexpressions:

$$s = \left[\begin{array}{c} -20\,e^3d+16\,e^3d^2+6\,e^3+46\,e^2d-32\,e^2d^2-15\,e^2+20\,ed^2-34\,ed+12\,e-4\,d^2-3+8\,d \\ -10\,e^2d+8\,e^2d^2+3\,e^2-8\,ed^2+23\,ed-10\,e+2\,d^2-9\,d+5 \\ 2\,d-6\,ed+4\,e^2d-2\,e^2+3\,e-1 \\ -10\,e^2d+8\,e^2d^2+3\,e^2-8\,ed^2+7\,ed-2\,e+2\,d^2-d+1 \\ -20\,e^2d+16\,e^2d^2+6\,e^2+26\,ed-16\,ed^2-9\,e+4\,d^2+3-8\,d \\ 4\,ed-2\,d-2\,e+1 \\ \frac{2\,e^2d-3\,ed+d-e^2+2\,e-1}{(4\,ed-2\,d-2\,e+1)e} \\ -10\,e^3d+8\,e^3d^2+3\,e^3-57\,e^2d+56\,e^2d^2+14\,e^2-62\,ed^2+95\,ed-31\,e+16\,d^2+12-32\,d \end{array}\right]$$

This can be simplified by assuming that all powers of either variable greater than one go to zero, i.e. $e^nd^m \to 0$ where $n+m \geq 2$. Doing this on the subexpressions gives:

$$s_2 = \left[\begin{array}{c} 12\,e-3+8\,d \\ -10\,e-9\,d+5 \\ 2\,d+3\,e-1 \\ -2\,e-d+1 \\ -9\,e+3-8\,d \\ -2\,d-2\,e+1 \\ -3\,d+\frac{d}{e}-e+2-e^{-1} \\ -31\,e+12-32\,d \end{array}\right]$$

$$\begin{bmatrix}
\frac{31\,e-12+32\,d}{48\,e-12+32\,d} & \frac{-10\,e^2-9\,ed+5\,e}{12\,e-3+8\,d} & \frac{2\,e^2d-e^2-ed}{2\,d+3\,e-1} & \frac{-e+e^2+ed-2\,e^2d}{4\,d+6\,e-2} & \frac{-2\,e^2-ed+e}{12\,e-3+8\,d} & \frac{2\,e^2+ed-e}{48\,e-12+32\,d} \\
\frac{-10\,e-9\,d+5}{36\,e-12+32\,d} & \frac{10\,e+9\,d-5}{9\,e-3+8\,d} & \frac{-2\,ed+e+d}{2\,d+2\,e-1} & \frac{1-e-d+2\,ed}{4\,d+4\,e-2} & \frac{2\,e+d-1}{9\,e-3+8\,d} & \frac{-2\,e-d+1}{36\,e-12+32\,d} \\
\frac{2\,ed-e-d}{8\,d+8\,e-4} & \frac{-2\,ed+e+d}{2\,d+2\,e-1} & \frac{-1+e+d-2\,ed}{2\,d+2\,e-1} & \frac{2\,ed-e-d}{4\,d+4\,e-2} & \frac{-2\,ed+e+d}{2\,d+2\,e-1} & \frac{2\,ed-e-d}{8\,d+8\,e-4} \\
1/4\,\frac{3\,ed-d+e^2-2\,e+1}{e} & \frac{-3\,ed+d-e^2+2\,e-1}{e} & \frac{-2\,e^2d+e^2+3\,ed-e-d}{2\,ed+2\,e^2-e} & \frac{2\,e^2d-e^2-11\,ed+6\,e+5\,d-3}{4\,ed+4\,e^2-2\,e} & \frac{-3\,ed+d-e^2+2\,e-1}{e} & 1/4\,\frac{3\,ed-d+e^2-2\,e+1}{e} \\
\frac{-2\,e-d+1}{36\,e-12+32\,d} & \frac{2\,e+d-1}{9\,e-3+8\,d} & \frac{-2\,ed+e+d}{2\,d+2\,e-1} & \frac{1-e-d+2\,ed}{4\,d+4\,e-2} & \frac{10\,e+9\,d-5}{9\,e-3+8\,d} & \frac{-10\,e-9\,d+5}{36\,e-12+32\,d} \\
\frac{2\,e^2+ed-e}{48\,e-12+32\,d} & \frac{-2\,e^2-ed+e}{12\,e-3+8\,d} & \frac{2\,e^2d-e^2-ed}{2\,d+3\,e-1} & \frac{-e+e^2+ed-2\,e^2d}{4\,d+6\,e-2} & \frac{-10\,e^2-9\,ed+5\,e}{12\,e-3+8\,d} & \frac{31\,e-12+32\,d}{48\,e-12+32\,d}
\end{bmatrix}$$

Figure 3.1: Simplified inverted transition matrix

Substituting this into the original matrix leaves Figure 3.1 as the fully inverted transition matrix, Finally taking the limit of this matrix as $\delta, \varepsilon \rightarrow 0$ leaves the following matrix, which is supposed to represent the probability distribution of being in the relevant states:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-5/12 & 5/3 & 0 & -1/2 & 1/3 & -1/12 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\infty & -\infty & 2 & -\infty & -\infty & \infty \\
-1/12 & 1/3 & 0 & -1/2 & 5/3 & -5/12 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Figure 3.2: Probability distribution matrix

This result is unsatisfactory, and may be in part a result of the numerous simplifications required to derive this analytic result. Either way, we will now turn to a mechanism for finding stable states that requires a much less complicated method.

3.6 Stochastic Stability

The central idea behind stochastic stability will be introduced firstly with a mental picture, and then, in the chance this is not clear enough, using mathematics.

Simply put, stochastic stability analysis looks at how difficult it is to move between different states. It explains that the states that are the most difficult to get out of, and the easiest to get into, are the states that we should expect systems with stochastic variation to place positive probability on in the long run. Perhaps an illustration here is helpful. Picture a ball rolling along a sloping field. The field has various hills and valleys, peaks and troughs. Standard equilibrium, that of the first order concept, simply finds all of the points in this field that have a flat gradient. That is, if we place the ball perfectly on a given spot, it would not move. A rough parallel here is Nash equilibrium. If we find ourself in a given state, do any of the players have any incentive to change their behavior or position? The next step is to look at local asymptotic stability, or the second order condition in calculus. I would consider that maybe a couple of those flat points are actually at the top of hills. Thus it is extremely unlikely that any ball will come to rest on them naturally and even if a ball does find itself directly at that point, a short gust of wind may move it just enough such that it begins the long slow roll down the slope. This is an example of a equilibrium point which lacks asymptotic stability.

Stochastic stability takes the insight a step further. Let us say that the ball is constantly being buffeted by rather strong gusts of wind, enough to move it to the top of tremendous peaks, although not that often. Stochastic stability asks, if this ball is randomly being shifted around, which areas of the field can we expect the ball to spend most of its time in? It turns out that there is a analytic framework directly applicable to this question. This framework asks not only where the equilibria points are, and not only whether they are locally stable, but how strong they are in keeping the ball in the area. How deep are the wells, and how high a peak does the ball need to climb to find a different area considered stable?

Now in our system, any state where every player can find at least one neighbor with the same ideology qualifies as an equilibrium. Stochastic stability provides the tools for deciding which of these states should we expect to see most often when the process runs for an extremely long time.

We have already seen that when the choice error rate is positive, or both the neighbor and choice error is positive, we can move from any state to any other state. Thus, the system is irreducible or ergodic. In order to define the behavior of the system we need to go a step further and to do this, we need firstly to better outline the standard literature on stochastic processes and Markov Chains.

The non-perturbed version of our game outlined specific Markov process on the finite state space X. Adding stochastic errors in both the neighbor and choice process modifies the initial transition matrix to that seen above, from the initial process $P_{0\mathbf{xx}'}$ to $P_{\delta\varepsilon\mathbf{xx}'}$. This transition matrix is said to be a perturbed Markov process. We can further specify it as a *regular perturbed Markov process* if it qualifies under certain criteria. The first is that $P_{\delta\varepsilon\mathbf{xx}'} > 0$ for all values of $\varepsilon > 0$, which has already been shown (δ can add to the noise, but it is not enough to make the process irreducible). Further, we require that $P_{\delta\varepsilon\mathbf{xx}'}$ converge to $P_{0\mathbf{xx}'}$ for all $\mathbf{x}, \mathbf{x}' \in X$, specifically at an exponential rate. More specifically,

$$\lim_{\delta,\varepsilon \to 0} P^{\delta\varepsilon}_{\mathbf{xx}'} = P^{0}_{\mathbf{xx}'}, \tag{3.16}$$

and for some $\varepsilon > 0$,

$$0 < \lim_{\delta,\varepsilon \to 0} P^{\delta\varepsilon}_{\mathbf{xx}'}/\varepsilon^{r(\mathbf{xx}')} < \infty \tag{3.17}$$

Where $r(\mathbf{xx}') > 0$ and is uniquely defined. This $r(\mathbf{xx}')$ is called the *resistance* of moving $\mathbf{x}$ to $\mathbf{x}$. A helpful way to think of resistance is as a representation of the number of errors needed to transition between states. Where two states are in a communication class in the non-perturbed Markov process, then $r(\mathbf{xx}') = 0$, that is, we need 0 errors to get from one to the other. Note, this condition holds even when the probability of moving from $\mathbf{x} \to \mathbf{x}$ is extremely small, even in the non-perturbed process. Envision a system in which it is extremely unlikely to jump from state a to state b, say $\frac{1}{128}$. Now compare that to the same system where there is a resistance of 2 to get from a to c. In this system it takes two error jumps of probability ε to get from a to c, and the joint probability of both of those errors occurring is ε^2. As ε approaches 0, that probability gets astronomically small. Even if $\varepsilon = \frac{1}{128}$, the same as the probability of moving from $a \rightarrow b$, the chance of going from $a \rightarrow c$ is $\frac{1}{16384}$. For systems in which the resistance between states is rather high, of which there are examples later, resistance conceptually captures the difficulty of moving between states in a stochastic environment.

This is our third level of analysis in terms of stability, and the beginnings of a definition of stochastic stability. The final step is to work towards a unique stationary distribution, traditionally denoted μ^ε. A state is said to be *stochastically stable*(33) if

$$\lim_{\varepsilon \to 0} \mu^\varepsilon(\mathbf{x}) > 0 \tag{3.18}$$

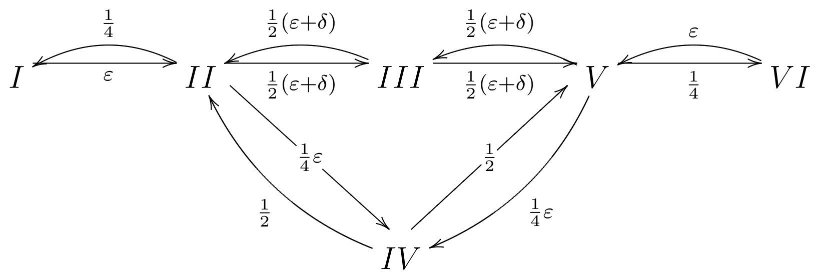
Adapting this definition to the system in question, and noting the case where $\delta = 0$ is identical to that above, yields this condition for stochastic stability,

$$0 < \lim_{\delta, \varepsilon \to 0} \mu^\varepsilon(\mathbf{x}) < 0 \tag{3.19}$$

Working towards the system under consideration, we can say that given the regular perturbed Markov process $P^{\delta\varepsilon}$ the stochastically stable states are chosen from the recurrence classes of the non-perturbed process P^0. Recall the previous section in which the recurrence classes I, III and VI were provided the labels E_1, E_2 and E_3, respectively. Given that the perturbed process is irreducible, for any pair of recurrence classes, there exists at least one path between them by moving from states. This is explicit in the definition of irreducibility. The *resistance* of any path is the sum of the resistances of its edges. For example, in order to get from I to VI, one potential path is the I$\rightarrow II \rightarrow IV \rightarrow V \rightarrow VI.Define\xi$ to be the particular path, where in the example $\xi = (I, II, IV, V, VI)$, beginning at E_1 and ending at E_3. The resistance of this path is the sum of the individual resistances of each state transition in a given path. For the example, $r(\xi) = r(I, II) + r(II, IV) + r(IV, V) + r(V, VI)$. Let $r_{ij} = \min r(\xi)$ for a general $\xi = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q)$. Two quick notations and additions must be made at this point. First, the convention is that if $P^{\delta\varepsilon}_{\mathbf{x}\mathbf{x}'} = P^{\delta\varepsilon}_{\mathbf{x}\mathbf{x}'} = 0$, we say $r(\mathbf{x}, \mathbf{x}') = \infty$. This is never a concern for our model as it will always be irreducible for any $\varepsilon > 0$ but is a useful convention. Secondly, note that by the definition of recurrence classes, it is not possible to move from one recurrence class to any other recurrence class with resistance 0. This would imply that the two states share a communication class, and would violate the definition of a recurrence class.

In order to calculate the stochastically stable states, we create a graph which demonstrates the minimum resistance between any and all pairs of recurrence class in the model. Using this graph to provide a relatively simple representation of the resistance concept.

Working off the state transition diagram, this directed graph is relatively simple to construct. With one caveat, this time we work off a state transition diagram which is simplified further to aid in the calculations of resistances by setting$(1-\varepsilon)$ and $(1-\delta) = 1$. This leaves the fundamental resistance unchanged, but significantly helps the calculations of resistance. It is technically correct to say that transiting from a state with three 0s and one 1 to a state of four 0s occurs with probability $\frac{1}{4}(1-\varepsilon)$, however, leaving the $(1-\varepsilon)$ on the diagram, when $\varepsilon \to 0$ will needlessly complicate the calculation of the resistances. Simplifying the state transition diagram yields the following:



Recall that E_1 is state I, E_2 is state III, and E_3 is state VI. And that the resistance of traveling from, for example, state I to state III is the highest exponent of error read off the conjunctive probability of the 'least resistant' path from I to III. In this case $P^{\varepsilon}_{I,III} = P^{\varepsilon}_{I,II}P^{\varepsilon}_{II,III}$. Accordingly, the resistance of $P^{\varepsilon}_{I,III}$ is equal $r(I, III)$ where $r(I, III) = r(I, II) + r(II, III)$ and $r(\mathbf{xx}')$ measures the resistance of $P^{\varepsilon}_{\mathbf{xx'}}$. The least conjunctive probabilities of traversing between states and thus the least resistant paths are as follows:

$E_1 \rightarrow E_2 : r(I, III) = 2$

$$\varepsilon\frac{(\varepsilon+\delta)}{2} = \frac{\varepsilon^2}{2} + \frac{\delta\varepsilon}{2}$$

$E_1 \rightarrow E_3 : r(I, VI) = 2$

$$\varepsilon(\frac{\varepsilon}{4})(\frac{1}{2})(\frac{1}{4}) = \frac{\varepsilon^2}{32}$$

$E_2 \rightarrow E_1 : r(III, I) = 1$

$$(\frac{\varepsilon+\delta}{2})(\frac{1}{4}) = (\frac{\varepsilon+\delta}{8})$$

$E_2 \rightarrow E_3 : r(III, VI) = 1$

$$(\frac{\varepsilon+\delta}{2})(\frac{1}{4}) = (\frac{\varepsilon+\delta}{8})$$

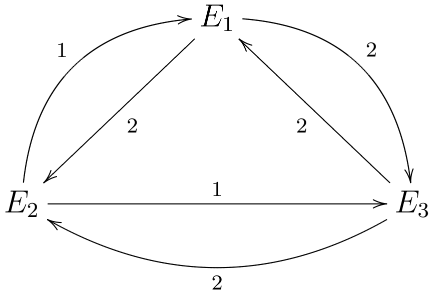
$E_3 \rightarrow E_1 : r(VI, I) = 2$

$$\varepsilon(\frac{\varepsilon}{4})(\frac{1}{2})(\frac{1}{4}) = \frac{\varepsilon^2}{32}$$

$E_3 \rightarrow E_2 : r(VI, III) = 2$

$$\varepsilon\frac{(\varepsilon+\delta)}{2} = \frac{\varepsilon^2}{2} + \frac{\delta\varepsilon}{2}$$

For now, set $\varepsilon = \delta$. Putting the above together into one directed resistance diagram, we have the following::



A resistance *tree* at vertex j is a set of $K-1$ directed edges, where K is the number of vertices on the graph, that show a unique path from every other vertex on the graph to j. In this example, there are three recurrence classes, and as for each class there are three different rooted trees which demonstrate unique paths from all other recurrence classes. They are as follows.

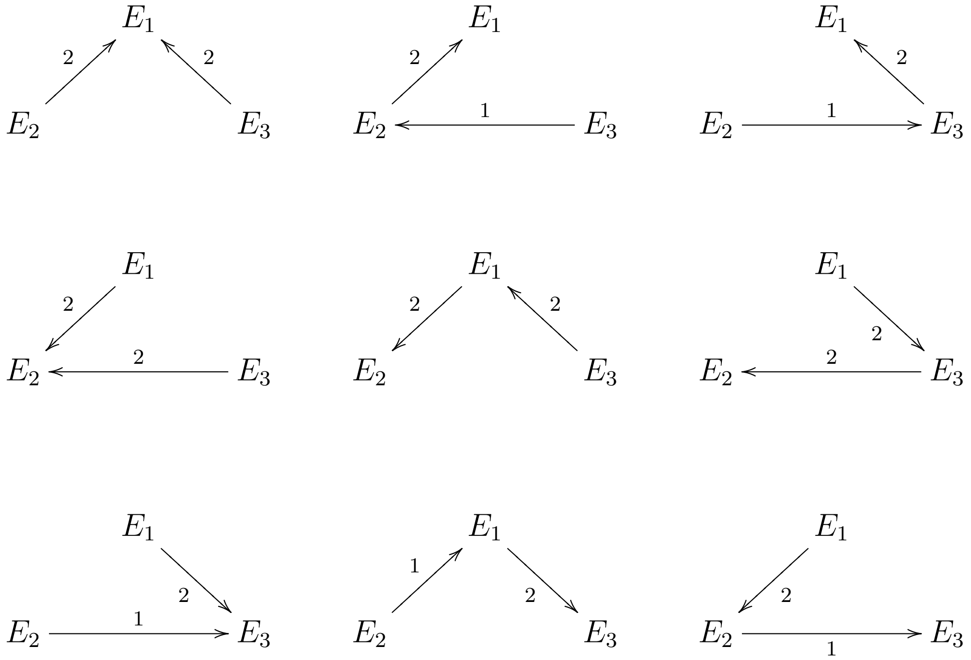


Figure 3.3: Resistance between recurrence classes

The *stochastic potential* of a given recurrence class E_j is defined as γ_j, where

γ_j is the minimum resistance over all trees ending at j. In this example, the stochastic potential for each tree rooted at a given j is the same. That is, due to there being only three recurrent classes and a highly symmetric game, the minimum resistance for each class is the same as the resistance from any tree rooted at that class. This will not be the case for other systems.

We can now apply Young's Theorem, in order to find the stochastically stable states. We can use this theorem to define the stochastically stable states are those in recurrence classes with the minimal potential relative to the other states in the system.

Young,1993 3.6.1 *Let P^ε be a regular perturbed Markov process and let μ^ε be the unique stationary distribution of P^ε for each $\varepsilon > 0$. Then the* $\lim_{\varepsilon\to\infty} \mu^\varepsilon = \mu^0$ *exists, and μ^0 is a stationary distribution of P^0. The stochastically stable states are precisely those states that are contained in the recurrent class(es) of P^0 having minimal stochastic potential.*

From the rooted trees diagram, the solution to this system is immediately apparent. Due to the ease of moving out of state III, labeled E_2, the other two recurrence classes have a lower minimum stochastic potential. Whereas both E_1 and E_3 have a minimum stochastic potential of 3, whereas E_2 has a minimum stochastic potential of 4. Thus, by this method, it is confirmed that perfect homogeneity states can be the long run stochastically stable states of this model and that in long run the process will spend much more time in states I and VI than in any other.

Note that the restriction of $\delta = \varepsilon$ *does* effect the dynamics of this system in an important way. The rate at which δ and ϵ approach 0 is significant. Recall the model where δ and ε are of similar magnitude and approach 0 at the same rate, the transition probability of moving out of state III into either state II or state V is,

$$\begin{tikzcd} II & III \arrow[l, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"'] \arrow[r, "\frac{1}{2}(\varepsilon+\delta)-\delta\varepsilon"] & V \end{tikzcd}$$

Where the total probability of moving out of state III is $(\delta + \varepsilon)$. Also recall the transition probabilities of moving into state III from either II or V.

$$II, V \xrightarrow{\frac{1}{2}(\varepsilon+\delta)} III$$

To demonstrate the importance of different rates of decrease between δ and ε, consider a case where we hold $\delta = 0$, a model with no errors in neighbor choice. Eliminating δ does *not* change the order of the resistance to get into or out of state III. The move directly into or out of state III still has a resistance of order 1. However, this is true in the case where δ approaching 0 at a slower rate or is held constant! The case where δ is held constant and does not trend toward zero eliminates the stability of segregated states in this model, and in general. A positive and constant probability of neighbor error implies that the resistance in moving in or out of state III is now 0, due to the fact that neighbor error in a segregated state collapses the barriers between segregated political ideologies. Thus, this eliminates the segregated states as a recurrent class and increases the relative stability of perfectly homogeneous or convergent states.

3.7 Expanding the Game

A cognitive dissonance game with four players and two potential political ideologies is rather straightforward. This section will demonstrate that by slightly expanding one parameter of this game, the range of potential political ideologies, can alter the dynamics of the system and indeed have tremendous implications for the complexity of working through the system.
The first indication of the consequences of this expansion for the complexity of this process is a simple calculation of the total number of potential states in this game. Given three potential values of x, and four players, we have a system with $3^4 = 81$ potential states. Clearly, in order to make this workable, some of the shortcuts demonstrated in the previous chapter are helpful. Following the method set out in section 3.5, we can enumerate the type of states under consideration; that is, by bunching groups of states with symmetric values and behavior under umbrella categories via specific notation. Types of states will be identified by the number of nodes with values of 0, 1 and 2. For example, under

this identification taxonomy, state 013 represents any state with 1 node of value ‘one’, and 3 nodes of value ‘two’. That is, states $\begin{smallmatrix}1&2\\2&2\end{smallmatrix}$, $\begin{smallmatrix}2&1\\2&2\end{smallmatrix}$, $\begin{smallmatrix}2&2\\1&2\end{smallmatrix}$ and $\begin{smallmatrix}2&2\\2&1\end{smallmatrix}$. As before, we represent segregated states as distinct from disconnected states, this time with the designations ‘s’ and ‘d’, respectively. There are 21 type of states in this model specification. They are presented by their identification, a sample state, and the total number of states in each categorization.

State ID	Sample	No. of states
400	$\begin{smallmatrix}0&0\\0&0\end{smallmatrix}$	1
310	$\begin{smallmatrix}1&0\\0&0\end{smallmatrix}$	4
301	$\begin{smallmatrix}2&0\\0&0\end{smallmatrix}$	4
220s	$\begin{smallmatrix}1&1\\0&0\end{smallmatrix}$	4
220d	$\begin{smallmatrix}1&0\\0&1\end{smallmatrix}$	2
211s	$\begin{smallmatrix}1&2\\0&0\end{smallmatrix}$	8
211d	$\begin{smallmatrix}1&0\\0&2\end{smallmatrix}$	4
202s	$\begin{smallmatrix}2&0\\2&0\end{smallmatrix}$	4
202d	$\begin{smallmatrix}0&2\\2&0\end{smallmatrix}$	2
130	$\begin{smallmatrix}0&1\\1&1\end{smallmatrix}$	4
121s	$\begin{smallmatrix}1&2\\1&0\end{smallmatrix}$	8
121d	$\begin{smallmatrix}2&1\\1&0\end{smallmatrix}$	4
112s	$\begin{smallmatrix}2&2\\0&1\end{smallmatrix}$	8
112d	$\begin{smallmatrix}2&1\\0&2\end{smallmatrix}$	4
103	$\begin{smallmatrix}2&0\\2&2\end{smallmatrix}$	4
040	$\begin{smallmatrix}1&1\\1&1\end{smallmatrix}$	1
031	$\begin{smallmatrix}2&1\\1&1\end{smallmatrix}$	4
022s	$\begin{smallmatrix}1&1\\2&2\end{smallmatrix}$	4
022d	$\begin{smallmatrix}1&2\\2&1\end{smallmatrix}$	2
013	$\begin{smallmatrix}1&2\\2&2\end{smallmatrix}$	4
004	$\begin{smallmatrix}2&2\\2&2\end{smallmatrix}$	2

Figure 3.4: Enumerated States for three variable system

In the same manner as in the previous section, we can build a complete transition probability diagram for the entire system by finding the transition probabilities of each individual state. In the following diagrams, we substitute ϕ for $(\delta + \varepsilon)$ for clarity, in addition to shortening $(1 - \varepsilon)$ to 1 and the like. This is relevant for the transition diagrams like 220d where it looks as if the probabilities sum to greater than one, but where the probability of not making an error is implicit in any transition without ε. Appendix 1 is available for reference as to how to work through the specific state transitions for all 21 types of states.

In order to find the stochastically stable states, we proceed in the same fashion as before. First, combine the individual state transition diagrams into a cohesive picture of the system and second, after identifying the recurrence classes, calculate the stochastic potential of each recurrence class and use that to find the stable states. Combining the individual diagrams into one is rather cumbersome, so diagrammatical shortcuts can be extremely useful here. The combined diagram will follow the style of the example diagram below:

Solid arrows - one way resistance (to the right)

$$\begin{tikzcd} a \arrow[r] & b \end{tikzcd}$$

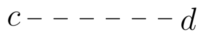
A rightward solid arrow demonstrates lack of resistance in moving from left to right, here $a \rightarrow$. However, in this game, there are no states for which there is no resistance in either direction. As such, the b to a transition only occurs with resistance 1. Resistance 1 transitions include all state to state transitions which only occur as a result of one error. This can be a choice error, or a state which can be left by either a choice error or a neighbor error. Either way, the order of the resistance is one. The direction of the transition is specified by the nature of the combined state transition diagram, in which all rightward transitions represent states where the total sum of the individual nodes has increased, ie: a zero has converted to a one or a one has converted to a two.

Dotted arrows - one way resistance (to the left)

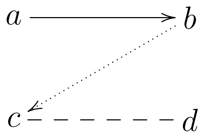
$$\begin{tikzcd} c & b \arrow[l, dotted] \end{tikzcd}$$

This represents the same dynamic as a solid arrow to the right, except for transitions moving to the left. Leftward transitions represent a shift between states where the total sum of individual nodes values has decreased; ones converted to zeros or two converted to ones.

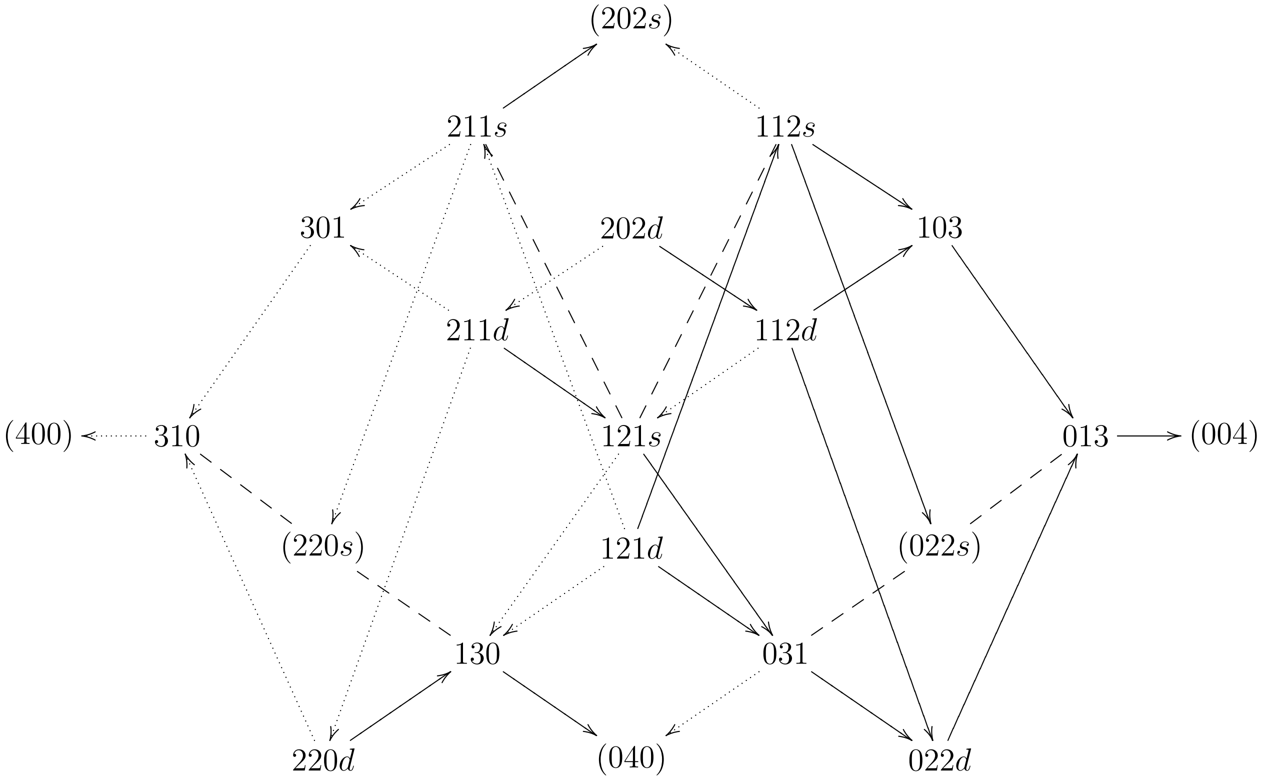
Dashed lines - two way resistance (both directions)



A dashed line between states represents a potential transition in which the resistance in either direction is one.
Putting this example diagram together into one diagram, we have;



The combined diagram represents the entire system. Here, there is no resistance in moving from $a \rightarrow b$ and $b \rightarrow c$. There is a resistance of one in moving from $b \rightarrow a$, $c \rightarrow b$ and between $c \leftrightarrow d$.
Finally combining the individual state transition probability diagrams from our real process, we have the following representation of the system.

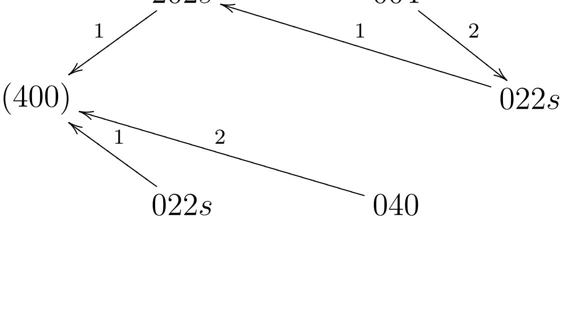


Simplified Resistance diagram for entire system

Working off of this diagram, it is apparent there are 6 recurrence classes for this system: $(400), (220s), (040s), (022s), (004)$ and $(202s)$.

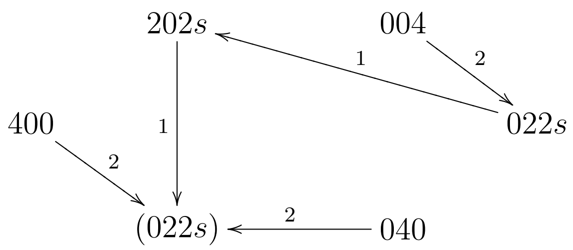
Following the same method as in the game with two potential political ideologies, but jumping right to the least resistant tree, we find the following.

Resistance tree rooted at state 400/004



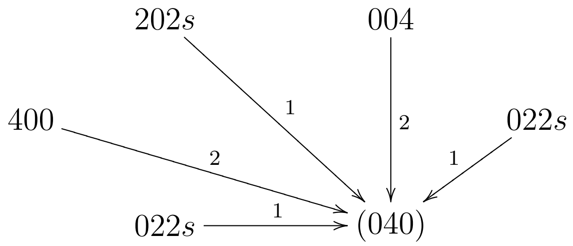
The sum of the branches of the least resistant tree rooted at 400 is 7. This is the stochastic potential of the recurrent classes 400 and 004. Due to symmetry, the same holds for 004.

Resistance tree rooted at state 220s/022s



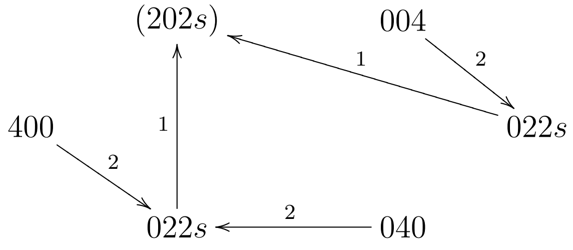
The stochastic potential of the recurrent classes 220s or 022s is 8.

Resistance tree rooted at state 040



The stochastic potential for 040 is 7.

Resistance tree rooted at state 202s



The stochastic potential for 202s is 8.
The conclusion to this system is that the stochastically stable states of a cognitive dissonance game played on a 2x2 grid with three potential values for political ideologies are all states convergent states with perfect homogeneity of preferences. These states have a stochastic resistance of 7 whereas all other states have a minimum of at least 8.

3.8 Preliminary Conclusions

So, what can make of all of this? First, the recurrence classes of the cognitive dissonance game are the same states as those identified with our first-pass Nash Equilibrium approach. Secondly, while segregated states are recurrence classes, their vulnerability to destabilizing stochastic shocks implies that in the long run, we would expect the system to spend more time in homogeneous states than segregated ones. Finally, we quickly run into severe diminishing returns with this analytic approach. Going through the same analytical process with a game involving four potential political ideologies requires the parsing of 256 states. Expanding the height and width of the two ideology game broadens the system to include 512 potential states. A 10x10 system, with 5 potential political ideologies provides $8x10^69$ potential states. At this point, given the tools presented in this paper, this route of analysis seems relatively exhausted. However we can take the commonalities between the two games as potentially yielding conclusions applicable to a broader environment.
The first conclusion is that common to both the two and three ideology setting is a greater stochastic stability of convergent states. A system with stochastic errors and one in which every state is accessible from every other state with some positive probability as $t \rightarrow \partial$ will 'select' convergent states as the stochastically stable recurrent classes.
The second broad conclusion is that the rate at which δ approaches 0 has important implications for the relative stability of segregated states. Specifically, values of δ which do not trend to zero eliminate segregated states as potential candidates for stochastically stable states by converting them into transient states. This will have an important contribution to the next section.

With this in mind, we now turn towards a different method of analysis altogether, agent-oriented simulations.

Chapter 4

Simulations

4.1 The Rationale Behind Simulations

In a fundamental sense, this paper is an argument for incorporating a certain perspective into our thinking on the evolution of political ideology in a dynamic socially connected world. At this point, there are two primary reasons why this argument is well served by the use of computer simulations. The first is the hard rule of diminishing returns encountered in using analytical methods on cognitive dissonance games. Thus far, there are clear stylized conclusions derived from analytic work. At this time however, a more general analytical approach to this problem is unavailable. Simulations provide a perspective that takes us beyond these simple games, and with the capability to look into systems of several orders of magnitude more complex than those found in Chapter 3. Whereas with stochastic analysis, every small step into greater complexity results in an exponential increase in the work required to come to conclusions, with simulations, every configuration of parameters takes exactly the same amount of time to come to a conclusion. It may seem like a black box, but by tying it as closely as possible with the model in the previous section, some legitimacy is restored.

The second primary benefit of using simulations is that they can, if successful, support those stylized conclusions derived from analytic results. The simulation of a system possessing identical characteristics to the analytic games in Chapter 2 affirms our confidence in the results of both mechanisms. Further, in simulating these parallel systems, it not only provides further support to our conclusions, but in a way, it buttresses the justification for using simulations in the first place.

This chapter will proceed with these two benefits in mind. The next section is a brief introduction to the use and application of agent oriented simulations in the social sciences. As the motivation and the methodology in this paper depart from the traditional sociological or political science approach used in these studies, this use of agent oriented simulation in the literature serves more as a qualitative introduction than as a source of theoretical departure for this work. The third section explains the way in which simulations were run. The fourth section addresses the rationale for using simulations, by explaining how and under what conditions the simulation results converge to those of the analytic games. The fifth section broadens the input parameters of cognitive dissonance games and presents some interesting results and supporting conclusions. The final section deals extremely briefly with quantal weights.

4.2 Introduction to Simulations in the Social Sciences

Agent-oriented computer simulations have recently become a very popular means of analyzing social phenomena. Simulations posses the unique combination of being endlessly adaptable to the question at hand, while simultaneously being able to incorporate incredibly complex systems into a modeling framework. Perhaps the strongest case for simulations however, is in their use at dealing with theories of social interaction that have some emergence properties associated with them.

It is this combination of flexibility and complexity, along with the rise of programming languages designed for social simulations, that undoubtedly contribute to the increase in interest in simulation methods. For an excellent overview of the use of simulations in describing political discourse at a society-wide level, see Huckfeldt et al(2004).(22)

One of the most well-known uses of simulations was Robert Axelrod's research into evolutionary algorithms. Axelrod held a competition which called upon economists to submit strategies for an iterated prisoner's dilemma tournament. After pitting algorithms against each other directly, Axelrod embedded the various strategies in populations of adaptable agents playing in an evolutionary environment. His famous result was that the simplest strategy, tit-for-tat feedback, won the evolutionary tournament.(9) This is an excellent example of the potential advantages of simulations. There are many cases where a system in question is too large or too complex to analyze directly, and in these cases, simulations provide a neat tool to derive results in the absence of rigorous methods. This paper deals with one of those complex systems and as such, this is a good point to introduce how the simulations were designed.

4.3 Programming the Simulations

The goal of the simulations is as closely as is possible to replicate the operation of the games in the analytic section. Thus, the coding of simulations attempts to use algorithms that are functionally equivalent to processes in the mathematical games. The simulation environment was coded using REPAST,(12) a Java package designed specifically for use in agent-oriented social simulations. The simulations were built, partially with reference to Andrew Bertie's(11) REPAST implementation of Axelrod's model of cultural dissemination.(10) This current section may be instructive for those interested in the manner in which these algorithms were written and the way in which the simulation program was created. However, it will prove tedious for those uninterested in these methods and thus it is recommended that the casual reader skip this section.

Turning our attention to the manner in which the simulations were programmed; as in the mathematical modeling framework, players exist on a grid and they are connected to those immediately adjacent to them. There is a meta list of players, with their corresponding locations determined by x and y coordinates on the grid. The state of any system is simply the political identification of every player. Players can only interact with direct neighbors, either horizontally or vertically. Players on the edge of the grid have a reduced set of neighbors corresponding to the limited number of players neighboring them. For example, in any game, there will always be four players with only two neighbors, those players located on the corners. In a 2x2 box game, obviously all players are considered corner players and as such, only have two neighbors. Simulations take place in a superclass named *CampbellModel*, the grid takes place in an environment called *Grid*, with the specified instance of a grid in the game labeled *space*, and all method calls from the grid environment use the space name as the reference. Agents are called *CampbellAgents*. This taxonomy is in part a convention of the simulation literature and also played a functional role for the author when writing the code, to aid in distinguishing different pieces of code.

Finding a neighbor

An agent is selected at random, according to this code, getting a random integer corresponding to an agent's place in the agent list.

```
public void execute() {
    // Pick an agent at random and attempt to interact with a neighbor.
    int i = Random.uniform.nextIntFromTo(0, agentList.size()-1);
    CampbellAgent agent = (CampbellAgent) agentList.get(i);
    boolean event = agent.step();   // interact
```

agent.step() passes control to the individual agent, which executes this method,

```
public boolean step() {
        int bit;
        int model;
        CampbellAgent neighbor;
        model = space.getNeighbourhoodType(); //used to check model type

        if ( model == 3 || model == 4){
        //interact with neighbor determined by model type
            neighbor = (CampbellAgent) space.getNeighbour(x,y);
        return (interact (neighbor));
        }
            }
```

Firstly, this method checks which type of model the system takes place in. Model type three corresponds to the closest neighbor allocation mechanism, finding[1] the neighbor with the closest political ideology, and model type four returns a neighbor according to quantal selection mechanism. Taking a look at the relevant code from the getNeighbour function in the space environment:

```
public Object getNeighbour(int x, int y) {
        switch(neighbourhoodType) {
            case VON_NEUMANN:
                ...
            case MOORE:
                ...
            case GLOBAL_UNIFORM:
                ..
            case CLOSEST_NEIGHBOR:
                ...
            case QUANTAL_NEIGHBOR:
                ...
        }
        return null;    // to satisfy compiler.
    }
```

The getNeighbour method takes the variable neighbourhoodType, set at the beginning of the game by choice of model, and passes control to the relevant neighbor finding mechanism. The first three elements in the switch above are inherited methods which use a different range of potential neighbors to pass a random neighbor. Much of the simulation programming relies on the use of

[1] Repast, like Java, heavily utilizes inherited names and method calls, especially on certain important predefined names and methods, which in the case of neighbor, were not always consistently spelled. The spelling of neighbor caries in such methods and variables as neighbourhoodType and getNeighbour. Given this restriction, as much as possible the code uses the ‘neighbour’ spelling whereas this paper endeavors to use neighbor in the text. This was unfortunate, but necessary.

getVonNeumannNeighbors, a method that takes the location of an agent and returns a Vector with a list of the agents directly adjacent to that location. As this Vector can be of a length as little as two for corner agents, and as large as four for agents in the middle of the grid, the code often has to take this into account by placing checks on the number of neighbors. There are two methods used on these Vectors; the first is "neighbor = (CampbellAgent)neighbourhood.get(i);" which finds the object at the location 'i' and *casts* it as an object of type CampbellAgent. At the most basic level Vectors hold Objects, thus informing the program that it is dealing with an object of type CampbellAgent is very important. It allows the program to use the specific methods and variables only applicable to CampbellAgents and not applicable to generic Objects.
Taking a closer look at the *CLOSEST NEIGHBOR* selection code,

```
case CLOSEST_NEIGHBOR: /*This method first checks to see if the
agent makes an error in
*finding closest neighbor If there is no neighbor error, ie,
*errorCheck returns false,     then this simply returns the
*closest neighbor available to the agent in question.  Otherwise
*the method has to find a neighbor  who has a political
*ideology different from that of the closest neighbor.
*Remember that neighbor error means agents specifically chooses a
*non-optimal neighbor to interact with. If there were two
*neighbors with the same ideology close to the player in
*question, then a * neighbor error should return neither!
* */
     neighbourhood = space.getVonNeumannNeighbors(x, y,
                     neighbourhoodExtent, neighbourhoodExtent, false);

    /*This checks first to see all of the agent's neighbors have the
      same political ideology If all neighbors
      do have the same ideology, it matters not which one is returned,
       so choose random.*/
    if (this.checkSameVal(neighbourhood) == true){
        int location = Random.uniform.nextIntFromTo(0,
                                    neighbourhood.size()-1);
        return neighbourhood.get(location);
       }
    else{
       ...
       location = this.getClosestLocation(x, y, neighbourhood);

       //Check to see if error occurs.
       neighborErrorCheck =  this.errorCheck(
                                        errorNeighborProbability);
       if (neighborErrorCheck != true){
          return neighbourhood.get(location);
          }
       else{
       ...}
```

In this method, there are three potential outcomes. One is that all of the neighbors surrounding our agent have the same political ideology, that is, *this.checkSameVal(neighbourhood)* returns true. Then the method returns a random neighbor from the set of neighbors provided by *getVonNeumannNeighbors*. When this is not the case, the program finds the integer location of the closest neighbor by a further method call to *this.getClosestLocation*, setting location as the correct value. The program then checks to see if it should have the agent make an error in choosing which neighbor to interact with. This is done in the standard way, by drawing a random number in the zero to one range and observing if it is less than or equal to the error probability parameter provided by the user, *errorNeighborProbability*. If there is no error, then the program simply returns the object from the 'neighbourhood' vector at the correct location.[1] In the case where there is an error, the program finds the political ideology of the correct

[1] Again, it is important that this method return a generic type Object, hence the focus on getting the location of the closest neighbor and just passing the neighbor directly.

neighbor, and then runs a do-loop, finding a random neighbor who does *not* posses that ideology.

```
if (neighborErrorCheck != true){
   ...
                    }
else{
   rightNeighbor = (CampbellAgent)neighbourhood.get(location);
   rightVal = rightNeighbor.getFirstTrait();
   do{
      wrongLocation = Random.uniform.nextIntFromTo(0,
                                          neighbourhood.size()-1);
      wrongNeighbor = (CampbellAgent)neighbourhood.get(
                                                  wrongLocation);
      wrongVal = wrongNeighbor.getFirstTrait();
      while (rightVal == wrongVal);
      return neighbourhood.get(wrongLocation);
```

The algorithm at the center of the *getClosestLocation* method is just for loop, running through the list of neighbors, and setting the location variable to the location of the neighbor found with the least distance to the political ideology of the agent in question.[1]

```
for (int i = 0; i < size; i++){
            double thisDistance;

            neighbor = (CampbellAgent)neighbourhood.get(i);
            thisDistance = agent.totalDistance(neighbor);
            if (distance <= thisDistance);
            else{
                location = i;
                distance = thisDistance;
                }
```

[1] Note that this algorithm uses a total distance method. This code was prepared with an eye to potentially expanding the number of traits in question, as in the Axelrod cultural model, that is not used in this paper.

```
            }
return location;
```

The *QUANTAL NEIGHBOR* model uses a broadly similar process, excluding neighbor errors, but returning the location of a neighbor as an argument. It does this in two steps,

```
for (int i = 0; i < size; i++)
    {
     neighbor = (CampbellAgent)neighbourhood.get(i);
     distance[i] = Math.abs(cog - neighbor.getFirstTrait());

     weights[i] = Math.exp(-distance[i]);
     sum += weights[i];
} for (int i = 0; i < size; i++){
    weights[i] = weights[i]/sum;
} return weights;
```

The first step involves creating a vector with the quantal weights of each neighbor. In this step, a given weight corresponds to the quantal function from chapter two, that is

$$w_{ij} = \frac{e^{|x_i - x_j|}}{\sum_{k \in N_i} e^{|x_i - x_k|}}$$

Using this weighted vector of probabilities, the method *getWeightedNeighbor* picks a neighbor with weighted probabilities according to their respective weights.

```
 double lowerBound = 0;
 double upperBound = 0;
 double r = Math.random();

 for(int i = 0; i < size; i++){
     upperBound+= weights[i];
     if(r <= upperBound && r > lowerBound){
       location = i;
```

```
        }
      lowerBound = upperBound;
```

Using these two *neighborhoodTypes*, the simulation engine can model the system using both the closest neighbor allocation mechanism and the quantal weighting allocation mechanism. Either of these returns a specific neighbor that the initial agent interacts with. The *interact* method then takes the political ideology of a given neighbor, and finds the positive or negative difference between the agent's trait, *traits.[bitTry]* and the *neighourTrait.* This difference, and the location of the trait in question on the list of potential traits[1] are passes as arguments to a method named *getMove* which provides the specific change in traits the agent chooses.

```
...
 neighbourTrait = neighbour.traits[bitTry];

 double diff, oldTrait;
 diff = traits[bitTry] - neighbourTrait;
 oldTrait = traits[bitTry];
 traits[bitTry] += this.getMove(diff, bitTry);
```

getMove checks to see if a choice error occurs, using the same method employed in checking for neighbor error.

```
double errorChoiceProbability; //used to find out where the
neighbour's value is
         boolean check;
         ...
         check = space.errorCheck(errorChoiceProbability);
         if (check){return this.getWrongMove(diff, featureID);}
         else{return this.getRightMove(diff);}

\subsection{Interacting with a neighbor}
```

[1] For all of the games in this paper, the array of traits is of length one corresponding to the one variable under examiniation, political ideology.

In the case of no choice error, *check* has a value is false, and the method returns the right move, otherwise a wrongMove is returned.

```
public double getRightMove(double diff){
        if (diff > 0){return -1;}
        else{
            if (diff == 0){return 0;}
            else{return 1;}}
} public double getWrongMove(double diff, int featureID){
 ...
rightMove = this.getRightMove(diff); trait = this.traits[featureID];

if (trait == 0){
    do  {wrongMove = Random.uniform.nextIntFromTo(0,1);}
    while(rightMove == wrongMove);}
else{...} return wrongMove; }}
```

The code above requires the use of nested if-loops, in order to ensure that an agent making a wrong move will go outside the ideological bounds of the game as set in the model.
Each time this process completes, a *tick* of time passes, corresponding to the number of rounds the game is allowed to run. While these are the central algorithms involved, there is a significant portion of the code designated to keeping track of the state of the process and recording data for empirical analysis.

Keeping Track of the State of Things

This section briefly explains how the runtime system records the state of the system. In general there are three categories of states, perfect convergence, perfect segregation and out of equilibrium. Perfect convergence occurs when all of the agents in the game have the same political ideology. *sysDistribution* is an array of length ten, which tracks the number of times this occurs and where it occurs. The program sets a variable *convergenceTime* equal to the tick time to record how quickly, a given simulation takes to get to its first convergence. if at all.

```
for (int i=0; i < agentList.size();i++){
    agent = (CampbellAgent) agentList.get(i);
    if (val != agent.getFirstTrait()){
       same = false;
...
    if (convergenceTime ==0){
        convergenceTime = (long)getTickCount();
        }

    x = agent.getFirstTrait();
    sysDistribution[x] +=1;
```

The other two types of states are recorded in an array of length 2 named *sysNonConverge*. Perfect segregation occurs when all agents are not the same but every agent can find at least one neighbor with the same political ideology. This coincides with our initial Nash Equilibrium definition.

```
if (space.agentsOutOfEq()==0){
    sysNonConverge[0]+=1;
    System.out.println("Perfect Segregation");
```

When the state is neither in convergence nor in segregation, we say that the system is out of equilibrium and *sysNonConverge[1]* is increased by 1. The final output of every simulation reads these arrays, converts them to strings, and prints them in a file, along with the model parameters and other relevant information. Note that the program does not keep track of which segregated state the process is in, only that the system is in a segregated state and the number of times during a run that the system was in any segregated state.

Running Simulations

REPAST is designed to be operated, when desired, in multiple simulation runs. Each simulation takes several inputs as parameters and runs to a set time or goal. The first and most important specification is the model type, as outlined above; this defines how the program finds neighbors to interact with. The simulation also takes in values for the probabilities of error in both ideology choices and neighbor choice error. Simulations also allow the user to set the width and height of the simulation world, and by extension, the number of agents. Finally, the user can set the range of potential political ideologies, along with several less important parameters that control how long the simulation runs and if it presents graphical information.

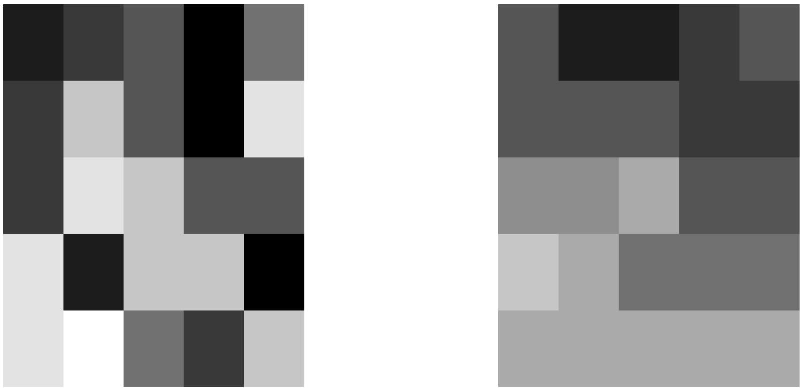
In general, each reported result is from at least 10 'runs', where each run lasted 100,000 rounds.

The key differences in outcomes pertain to:

1. The time it took each simulation to first reach convergence, if at all.
2. The relative time in which a simulation spends in the different types of states.
3. In the simulations where convergence is obtained, the relative stability of convergence.
4. In the simulations where convergence is obtained, the location of convergence relative to the range of potential political ideologies.

The beginning of every simulation assigns a random political ideology to every agent in the game. There is usually a brief number of runs in which the system converges to some form of segregated state. As we will see later, this force is relatively strong and robust to high settings of either error term.

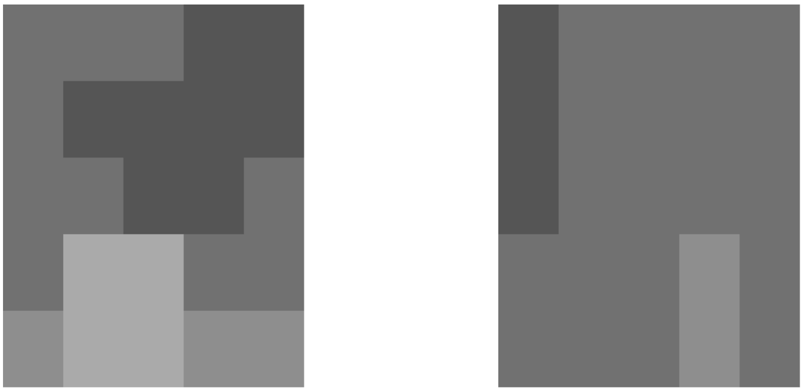
In order to demonstrate this settling period, figures X Y Z below show the evolution of a single run. This game is played on a 5x5 grid with 10 different shades corresponding to 10 different potential political ideologies. This simulation was run with zero choice error and 0.25 neighbor error, in order to speed up convergence. This result is relatively uninteresting as in the absence of stochastic choice variation, the almost random interactions between agents provide strong convergence dynamics to the system.



$(t = 1)$ **(t = 191)**

Figure 4.1: Initial political evolution

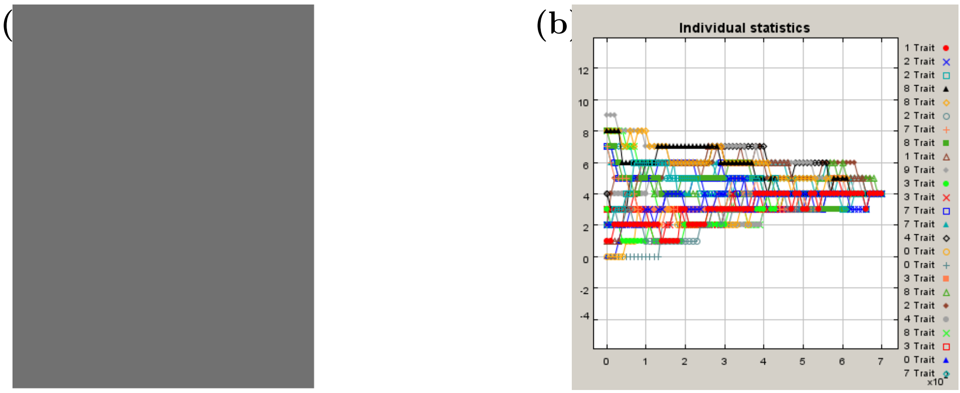
4.1



$(t = 440)$ **(t = 660)**

Figure 4.2: Convergence over time

4.2



(*t* = 700 **convergence)**

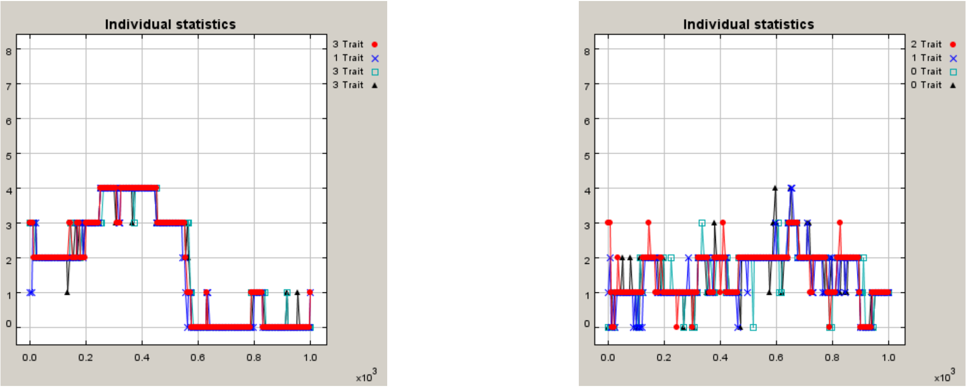
(Individual ideology over time)

Figure 4.3: Convergence and Time Path

4.3

Given that this example was selected to highlight key issues, it is perhaps unsurprising that the system converged within 700 iterations. That a system converges and the speed at which convergence occurs are both interesting considerations for analytical purposes. However, the manner in which systems evolve in the truly long term given stochastic variation is perhaps of greater relevance.

Figure 4.4 shows the individual ideological time path of two different models through the first 1000 ticks of runtime. Both example systems are 2x2 grids with five potential political ideologies, where the change is the level of choice error, ε in the different simulations. The data from which the graphs were generated is updated once every five ticks. The increase in ε is immediately apparent via direct observation of the noise surrounding states.



$\delta = 0.15, \varepsilon = 0.05$

$\delta = 0.1, \varepsilon = 0.1$

Figure 4.4: Convergence and Time Path

4.4

The following provides a sense of those diagrams become hard data; here is an example table of the frequency probabilities of different states derived from these runs. Conv-x is the number of rounds that every agent in the game had political ideology x. OofEq is the number of times the system is in a transient state, or 'Out of Equilibrium'.

δ	ε	Conv-0	Conv-1	Conv-2	Conv-3	Con-4	Conv%	Seg%	OofEq%
0.10	0.05	346	32	116	147	159	*80%*	*6%*	*14%*
0.10	0.10	17	248	319	20	0	*60.4%*	*9.7%*	*29.9%*

Figure 4.5: Example simulations

Clearly these simulations do not provide sufficient sample sizes to derive any conclusions. However, there is one thing demonstrated by examples which directly parallels a key concept from the preceding chapter, that of path dependence. The important thing to take from these examples is that, even upon reaching convergence, the evolution of the process has really only just commenced. The simulation reaching convergence of ideology is only a representation of the process being in a particular state at a particular point in time. As is demonstrated above, even small levels of noise in the system completely alter the long run location of the process. In this sense, simulations confirm that this version of the game is ergodic. As long as the choice error is positive, even setting up a system with all ideologies set the same will not effect the long run average frequency distribution of the system. This was a key concept from chapter and it is satisfying to see it confirmed not only in mathematical terms, but also in a physical representation of the model. With these tools in hand, the simulation engine can now be brought to bear on identical systems to those developed in the modeling chapter in order to affirm their results.

4.4 Simulating a Simple Game

The previous sections have already explained the method for simulating games with the same parameters as those found in the analytic chapter. Briefly describing trivial cases, running simulations with no error will always result in either a homogenous state of all 0s or 1s, or a perfectly segregated state with two 0s and two 1s. Any positive probability on δ or neighbor error will destabilize the segregated states and, in the absence of choice error, lead to a perfectly stable absorbing state. It is only through the introduction of positive probabilities of choice errors that the system acquires the ability to engage in long term stochastically fluid behavior. Note, that while the jump off of zero is very important for δ in some cases, in general the dynamics of the system are driven by the setting of the error rate. As such, for consistency, the rest of this chapter will deal with the cases where $\delta = \varepsilon$. However, it is interesting to note that after this initial jump, the choice of δ seems to have very little if any effect on the long run behavior of the system.

Table 4.5 demonstrates how we can analyse, using simulations, the dynamics of a system as the δ and $\varepsilon \to 0$. It is clear that as these error terms go to 0, the system spends more and more time in convergence states. However, given a large enough sample size, the system can and will spend time in both states. In fact, in every single simulation, the process spent time in both convergence states. This is almost surprising given how low the error rates are set in the extreme cases. This highlights the importance of long simulation times, as with extremely low error rates, setting the runtime of the simulation to too low a sample size can result in behavior which appears to be path dependent.

δ	ε	State I(Hom)%	State VI(Hom)%	TotConv%	State III(Seg)%	OofEq%
0.0001	0.0001	50.06%	49.88%	99.94%	0.02%	0.04%
0.001	0.001	49.55%	49.85%	99.41%	0.19%	0.40%
0.01	0.01	47.03%	47.23%	94.26%	1.91%	3.83%
0.05	0.05	37.91%	38.01%	75.92%	8.00%	16.08%
0.10	0.10	29.65%	29.77%	59.42%	13.31%	27.27%
0.15	0.15	23.90%	23.77%	47.67%	16.99%	35.34%
0.25	0.25	15.94%	16.22%	32.16%	21.38%	46.46%
0.40	0.40	9.24%	9.17%	18.41%	24.40%	57.19%
0.75	0.75	1.77%	1.78%	3.55%	21.59%	74.86%

Figure 4.6: Convergence as a function of δ, ε

There are two important results from this batch of simulations. The first affirms the stochastic stability of states I and VI from chapter 2. The second is an observation of how strong the pull towards convergence is, even in an environment with extremely high error rates. Errors in this system are not just random choices, they are specifically sub-optimal choices, and as such, it is impressive that with error rates exceeding 25%, the system spends any time in convergent states at all.

Three ideologies

Setting up the model to run with three potential political ideologies is quite easy with simulations. Apart from the considerable computational time it takes to run simulations, once the simulations are coded it is just a matter of setting up the relevant parameter files. As mentioned previously, in setting low error parameters, one must be wary of not providing the system with enough time to demonstrate the true distributional dynamics. This may seem a rather absurd consideration when the base runtime used in this paper is 100,001, but considering the how the value of the conjunctive probability of successive errors changes as these error parameters approach zero, it is clearly a relevant issue. The following figure is a table representing the relative frequency distributions observed through a similar run of simulations to that of the third section. That is, the simulations occur on a 2x2 box grid and each simulation is a batch of 25 runs of 20001 ticks. The parameters for neighbor error, δ, and choice error, ε, were reduced to extremely low levels, and in doing this we observe the changes in the relative frequency of states of interest. Recall that state 400 is the recurrent class representing a state with four 0s. State 040 represents a state with four 1s, and state 004 represents a state with four 2s. It is important to note here, that while the rest of this paper used the term convergence in a way connotation a political consensus among the players in the game or a homogenous state, there is another sense of convergence which now becomes important.

The results of figure 4.4 are rather dramatic. In the same way as the mathematical chapter showed that the only stochastically stable states in a cognitive dissonance game are the convergent states, this simulation method demonstrates the exact same thing. As the error parameters approach zero, the system spends more and more time in convergent states, to the point where it spends almost no time whatsoever in either segregated states or transient states. We can see also that this system does not behave in exactly the same way as a system with only two potential political ideologies. This system spends a significantly greater time in the state 040, corresponding to four 1s. This is a further element of 'convergence', and a new result. It is a result that is completely general in this game to all systems where the range of political ideologies is greater than two. In each case, the probability of being in middle states is higher than that of being further out, as in a unimodal distribution. The distribution among the homogeneous states is not uniform in this case, but there is nothing in the resistance analysis that contradicts this. In fact, if one goes through the trouble of going to calculate the specific resistance in terms of probability, it is possible that an effect on the order of over 2:1 could be shown to reveal a slight preference from states with homogeneous ideologies close to the center of the political spectrum.

There is one problem with the figure above, which is the effects of a limited sample size are apparent. As previously mentioned, when the sample size of observations is limited by the runtime of the simulation, which in this case was only 20,000, it is quite easy to get asymmetric distributions. For error ranges near or above 0.01, the resulting distributions are within the bounds of acceptable symmetry for a random sample of that size relative to the stochastic component. However, it is apparent this is not the case for the case where the error term is set to 0.0001.

δ, ε	State 400%	State 040%	State 004%	TotConv%	Seg%	OofEq%
0.0001	29.49%	51.82%	18.59%	99.90%	0.03%	0.07%
0.0010	22.63%	56.15%	20.64%	99.42%	0.20%	0.38%
0.0100	21.48%	52.14%	20.57%	94.20%	1.92%	3.88%
0.1000	10.59%	35.63%	10.82%	57.05%	13.79%	29.16%
0.2500	4.95%	17.47%	5.05%	27.46%	19.28%	53.26%

Figure 4.7: State frequency distribution with three ideologies

Expanding the Parameters

In the previous chapter, the limits of analytic methods on this type of problem became readily apparent as soon as the game was expanded to more than two political ideologies. This framework allows us to play with the game settings in order to examine systems beyond the reach of our previous means. The following results came about by simulating various systems for 10 runs of 100,000 ticks. The error parameters were set at δ,$\varepsilon = 0.0001$ for reasons that will become readily apparent. The model settings were expanded in two dimensions, the range of political ideologies, from two to ten (labeled PolRange for brevity in the tabel), and the number of players, determined by expanding the height and width of the grid from 2x2 to 15x15.

	Grid Size			
PolRange	2x2	4x4	10x10	15x15
2	99.406%	93.26%	44.19%	4.00%
3	99.390%	93.67%	40.25%	10.00%
5	99.387%	93.81%	38.31%	5.79%
10	99.387%	93.73%	28.85%	5.30%

Figure 4.8: The effect of expanding parameters on convergence frequency

Table 4.7 shows clearly how expanding the number of agents in the game reduces the amount of time the game spends in convergent state. It is important to note that this is an entirely new result. Nothing in the analytic section suggested anything like this occurring.

Perhaps as interesting, is that the range of political ideologies has no perceptible effect on the frequency with which the state reached homogeneity. Presumably this is due to the convergence in political space to the center which was observed earlier. At this point we have two new insights to how cognitive dissonance games work that result directly from simulation data and not from mathematical analysis.

Of interest is also the relevant amount of time each state spent in the segregated states and out of equilibrium states:

The first thing that is immediately apparent from figure 4.9 is the tremendous jump in time the process spends in out of equilibrium states in response to increasing the number of agents in the model. This is a direct implication of the state spending less time in homogeneous states, but it is still good to see it bearing out in the data.
Also, figure 4.8 appears to reveal a slight temporary shift of probability into segregated states as a result of the increase in the number of agents. The effect is subtle enough to potentially be a sample size fluke, but could speak to the stability of segregated states in large populations. Intuitively, we might think there would be a greater trend towards segregated states in environments with lots of individuals. Before global convergence occurs, there is plenty of time for individuals to engage in locally stable segregated environments. This is a result common to some of the simulation literature, such as Abelson's 1979 paper titled *Social Opinion and Opinion Clusters*, in which he creates a model of opinion change that resulted i agreement within groups and disagreement between groups.(1) This result also contradicts the Axelrod's findings in his work on cultural diffusion. He found, using computer simulations, that the number of stable homogenous regions decreased as the number of agents grew,

Grid Size				
PolRange	2x2	4x4	10x10	15x15
2	0.19%	4.19%	28.91%	22.98%
3	0.20%	3.96%	28.61%	23.89%
5	0.21%	3.72%	30.26%	26.16%
10	0.19%	3.79%	33.73%	21.29%

Figure 4.9: The effect of expanding parameters on segregation frequency

Grid Size				
PolRange	2x2	4x4	10x10	15x15
2	0.40%	2.55%	26.90%	73.01%
3	0.41%	2.37%	31.14%	66.11%
5	0.41%	2.48%	31.43%	68.05%
10	0.42%	2.48%	37.41%	73.41%

Figure 4.10: The effect of expanding parameters on out of eq frequency

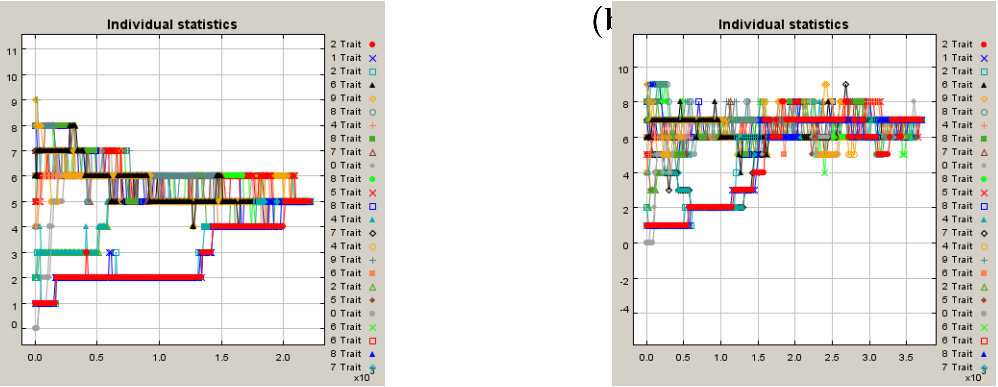
leading to larger subgroups in the population with homogenous views.(10) This result suggests the opposite.
One other explanation for the dynamics demonstrated here is that larger and more complicated systems take significantly longer to reach their first homogenous state in addition to being less stable. In many of the cases in these simulations with a 15x15 grid the system did not reach a homogeneous state within the first 100,001 rounds. A failure to reach a homogeneous state occurred between seventy and twenty percent of the time. The final figure of this section provides the average time to homogeneity for each model specification. For simulations where the state did not reach convergence, the maximum number of rounds was substituted provide some weight in calculating averages to runs where no convergence occurred.

Grid Size				
PolRange	2x2	4x4	10x10	15x15
2	8	247	12398	87344
3	12	201	17942	55164
5	16	254	20548	58684
10	25	431	24432	24432

Figure 4.11: Average time to convergence

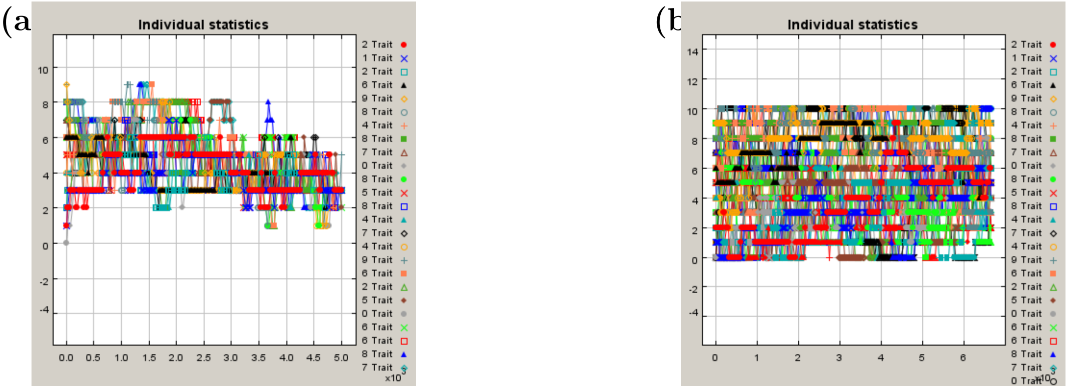
4.5 Revisiting Quantal Weights

This chapter has made no mention of the role of quantal allocation in weighting the probability of interacting a set of neighbors. This was in large part due to the fact that the types of data generated by quantal weighting does not vary in substance or form to that of closest neighbor allocation. However, just to provide a bit more color to our understanding of how these systems evolve, the following are four different version of a model running the quantal weighting of neighbors, with varying levels of choice error. Each game used 11 potential ideologies on a 5x5 grid.



(No choice error) (choice error with prob 0.05)

Figure 4.12: Quantal Allocation 4.12



(Choice error with prob 0.1) (Choice error with prob 0.25)

Figure 4.13: Quantal Allocation 4.13

Chapter 5

Conclusions

Cognitive dissonance is a powerful theory. This paper has endowed dissonance like dynamics is two different types of systems, that of the purely mathematical and the purely computational. Before moving on to the conclusions, it is necessary to first acknowledge the vulnerabilities of this type of research. The first potential problem with this approach is that it simply accords too much power to the role of dissonance reduction in determining the political ideology of individuals. That is, the assumption that a causal relationship may be teased out of cognitive dissonance at all. This is a real problem with this line of research: the potential that political choices are irreducibly complex decisions is a significant danger. To this, there are two responses. The first is an appeal to the authority of the research methodology of the vast literature on behavioral. Almost any behavioral model dealing with a sufficiently subtle phenomenon finds itself open to this critique, and on one level, it is a necessary component of looking into the causes of complex behavior. Further, it is always possible to retreat to the 'all-things-being-equal' defense; claiming that in isolation, it is plausible that these dynamics could drive the real world process. However, that response seems to skirt the issue. In fact, this paper is an argument that the dynamics of cognitive dissonance should be given more consideration as a good model of the political dynamics of individuals and large populations. In order to support this claim, however, it is clear that both micro-level and macro-level empirical work needs to be done in order to test the power of cognitive dissonance. The relative difficulty of parsing the myriad of potential causal factors, and disentangling the direction of causality in the relationship between social interactions and political choices creates an interesting area of potential future research.

The second possible vulnerability in this research comes from a theoretical perspective. This paper could make a more forceful argument if the preliminary conclusions were based in a more general analytic method. The conclusions in this paper derive legitimacy from two sources: special small case analytic models and computer simulations endowed with identical dynamics. However, in some cases, the conjectures are supported *only* by the simulation data, and even then, it comes from an observation of trends as opposed to an application of rigorous econometric methodology. While these two methods produce mutually affirming conclusions in many cases, it is still the case that there is a tremendous potential for further developing both of these approaches. The current limit in generality and scope of the analytic results is potentially a temporary issue, and one that may relent in the face of the application of more sophisticated mathematical tools. Failing that, there is the potential to refine and strengthen the causal conjectures derived from the simulation approach.

Finally, there are objections inherent both to the methodology of behavioral models in general and to the particular way in which that methodology is applied in this paper. There are an impressive array of assumptions contained in this paper, not only about the power of dissonance, but also regarding the manner in which individuals interact. Shoring up these assumptions in many cases suggests future research. In these cases, the relevant change to the model is explained in a subpoint. However, this is not the case for all, and the potential for future work in order to address these problems does not obviate these legitimate objections. They are as follows:

1. The assumption that individuals fundamentally operate in the same way. This paper attempts to address this objection by endowing the model with robustness to stochastic shocks. However, the heart of the model still assumes that individuals are fundamentally the same, and departures from this assumption are rare, uniformly distributed and temporary.

   - A model that specifically incorporates different types of individuals into the evolutionary dynamics of the game. This could be done either through natural selection dynamics, or simply as another way to check the strength of the conclusions in the face of varied model parameters.

2. The assumption that the political ideology of individuals can be condensed to a linear spectrum. There is a wealth of political science literature which suggests that a more refined perspective on the ways in which political beliefs vary will do a better job of explaining political ideology than a simple linear spectrum.

   - There is some reason to expect that cognitive dissonance dynamics could play out in a multi-dimensional political space. The simulation program was built with this potential expansion in mind, and some very limited trials seemed to indicate that the same dynamics played out, if over longer periods, but this was beyond the scope of the paper to insert.

3. Perhaps the most dangerous assumption that individuals can observe the political ideology of others, and allow or cannot prevent others to observe their own political ideology.

   - This could be mitigated by allowing players to engage in strategic communication of their political ideology, but that would only partially deal with to this significant issue.

4. Relating to the objection above, there is an assumption that there is a real connection between what amounts to am internal, subconscious choice of political ideology and external, conscious interactions with our environment. It is not at all guaranteed that even after condensing politics down to a single spectrum, that we can say individuals posses a x_i in any real way. There may be a constant, undirected drift of political ideology. There may not even be something sufficiently consistent, even when finally identified, to be operated on by the dynamics of cognitive dissonance.

5. Finally, the role of 'time' in this system was left intentionally vague. It is not defined how the conception of time applied in this game applies to the real-world time. Essentially, this method assumes individuals are infinitely lived and that they experience many interactions over the course of their life which have a strong impact on their political ideology. The dynamics of this game are not just dependent on the power of dissonance reduction in determining their political ideology, but also upon the frequency with which they interact with other individuals in ways that cause dissonance. It is not only the magnitude of the effect of dissonance but the frequency with which individuals engage in dissonance spurring interactions that lie at the core of this process, as it is driven over time.

In spite of these objections, this analysis does provide some interesting preliminary results. Preliminary because the analytic framework for both the mathematical approach and the computer simulation approach has been very specialized. We are left with several conjunctures regarding the behavior of individuals and social systems bound by cognitive dissonance. Recall first the two insights from cognitive dissonance which formed the basis for the rest of the work, that dissonance is powerful enough to alter the political ideology of individuals in response to disagreement, and that as a result of the desire to avoid this discomfort, individuals choose to interact more frequently with people they agree with. As these are inputs into our work and not outputs, they cannot be called conclusions, but it is important to state them here as an integral part of the structure of this paper.

Conjecture 1 *A social system based ruled by cognitive dissonance dynamics will likely exhibit a strong long term trend towards convergent states.*

This result is affirmed in both the mathematical and simulation chapters. We have from the analytic section that homogeneous states are asymptotically stable and stable in the face of stochastic perturbations. The simulation data repeated showed how the system had a long term preference for convergence. Even in the face of error rates as high as $\delta, \varepsilon = 15\%$ the simulations spend over 50% of the runtime in a state with homogeneous political ideology.

As most of the conclusions in the paper, this result has only been proved for the cases specifically mentioned. We have reason to believe that these results extend into systems with a greater range of political ideology, but as of this point, it has only been mathematically proved for a game with three ideologies. The simulation data suggests this result is extremely generalizable, and there is no reason to suspect otherwise for the mathematical, but at this point a more general proof is unavailable.

Conjecture 2 *Social systems with a broad range of political ideologies will exhibit a tendency towards convergence specifically to the middle ground of the political spectrum.*

This result comes exclusively from the simulation data. The analytic method can only suggest that homogenous states at the middle of the political spectrum qualify as stochastically stable, and in some cases, there is reason to believe the transition probabilities may suggest that the process spends more time in the middle of the spectrum as a result of the ease of transiting from other states to the center, in some cases with resistance equal to zero.
It is interesting that this conjecture supports the standard game theoretic conclusion regarding convergence to the center of the spectrum. In the standard game theoretic model, politicians compete for votes evenly distributed on a line, and the only Nash equilibrium in a two-person game is for both politicians to go to the center. This dissonance model provides an alternate explanation for the same convergence along the spectrum effect.

Conjecture 3 *Increasing the number of players greatly extends the amount of time the system takes to get to convergence. The effects on the stability of homogeneous states is unclear.*

Looking at figure 4.4 we saw that increasing the size of the grid, and correspondingly the number of agents in the game, has tremendous effects on the time it takes the system to first reach convergence. This is an incredibly important result, because whereas the rest of the work implies that we should see a very strong trend towards convergence in the real world if this were to apply, this allows us to take a step back and provide perspective on the effect of this dynamic. It is plausible to think that in a society with nine member they may reach societal consensus on big issues relatively quickly. When we expand the scope by increasing the number of people involved, we see similar dyanmics to figures in previous sections, that is, local convergence and global segregation. This global segregation may in the long run be unstable, but it is the norm in the time between the homogeneous environments. This was difficult to capture in data due to the way in which states are recorded. A state which is perfectly segregated bar one individual who made a random error is said to be out of equilibrium. Attempts to refine the model to capture this phenomenon proved incredibly awkward and severely slowed down the time simulations took to run.

Conjecture 4 *Increasing the range of political ideology does not seem to lead to more or less stability of homogenous states.*

This result may be somewhat surprising at first glance, but it reasonable upon consideration of how the system will play out. As agents begin to converge locally to form semi-stable partnerships with at least one neighbor who they can find with an identical ideology, this shaking out process will pull everyone to the middle if there is sufficient space in the political spectrum. In systems with many agents, it becomes increasingly difficult to shift the to a different homogeneous state of the game, and accordingly the state will spend most of its time near the center. In relatively small games, say a 2x2 box, it is much easier to move between homogeneous states, but over the long run, the system still spends a disproportionate amount of time in the middle of the political spectrum. However, the overall frequency of convergence seems to remain unchanged.

Conjecture 5 *Higher rate of interaction across ideological boundaries may have important effects on the stability of segregated states.*

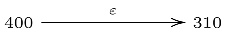
This conjecture is left intentionally vague due to the conflicting results between the two methods. The analytic method suggests that segregated states can be considered recurrence classes when δ equals or trends towards zero. There is also an intuitive argument that in environment with neighbor error, people are more likely to interact across ideological boundaries, and as a result of dissonance reduction, move towards closer to the other people in terms of political ideology. On the other hand, the simulation data does not support this idea. Aside from the relatively uninteresting case where $\varepsilon = 0$, the level of δ has no perceptible effect on the long run behavior of the system. This is perhaps because the transition to and from segregated states can occur from either a choice error or a neighbor error, and as a result, the resistance in and out of segregated states is a function of the maximum of δ and ε.

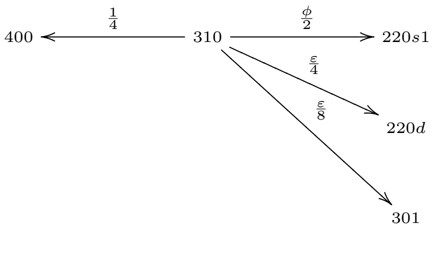
These conjectures arise from two methods of analyzing a system bound by insights derived from cognitive dissonance. It is the hope that this paper served as a first foray into the development of a new approach in applying cognitive dissonance to the issue of the evolutionary development of broad, network-based, social-political environments. There are clear areas on which to expand and solidify this approach, and this paper provides two alternative means to derive promising conjectures as to the state of dissonance bound systems, and our more general social environment.

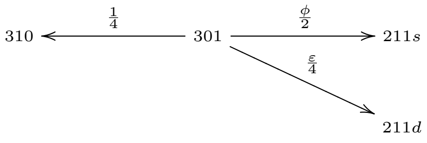
Appendix A

Expanded Game

Here are the individual state transition diagrams for each type in the three ideology game:







$$\begin{tikzcd}
310 & 220s \arrow[l, "\frac{(\delta+\frac{\varepsilon}{2})}{2}"'] \arrow[r, "\frac{\varepsilon}{4}"] \arrow[dr, "\frac{\phi}{2}"] & 211s \\
 & & 130
\end{tikzcd}$$

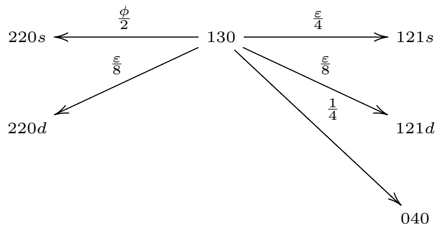
$$\begin{tikzcd}
310 & 220d \arrow[l, "\frac{1}{2}"'] \arrow[r, "\frac{\varepsilon}{4}"] \arrow[dr, "\frac{1}{2}"] & 211d \\
 & & 130
\end{tikzcd}$$

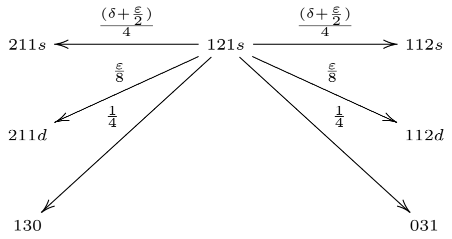
$$\begin{tikzcd}
301 & 211s \arrow[l, "\frac{1}{8}"'] \arrow[r, "\frac{1}{8}"] \arrow[dl, "\frac{1}{4}"'] \arrow[dr, "\frac{\phi}{4}"] \arrow[ddr, "\frac{\phi}{4}"] & 220s \\
220s & & 121s \\
 & & 121d
\end{tikzcd}$$

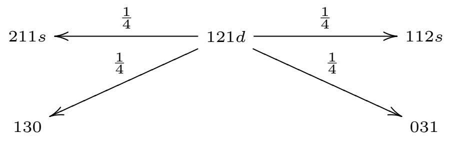
$$\begin{tikzcd}
220d & 211d \arrow[l, "\frac{1}{4}"'] \arrow[r, "\frac{\varepsilon}{8}"] \arrow[dl, "\frac{1}{4}"'] \arrow[dr, "\frac{1}{2}"] & 202d \\
301 & & 121s
\end{tikzcd}$$

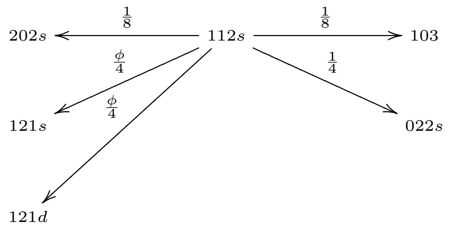
$$\begin{tikzcd}
211s & 202s \arrow[l, "\frac{\phi}{2}"'] \arrow[r, "\frac{\phi}{2}"] & 112s
\end{tikzcd}$$

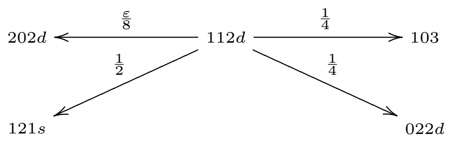
$$\begin{tikzcd}
211d & 202d \arrow[l, "\frac{1}{2}"'] \arrow[r, "\frac{1}{2}"] & 112d
\end{tikzcd}$$


220s
$\frac{\phi}{2}$
130
$\frac{\varepsilon}{4}$
121s
$\frac{\varepsilon}{8}$
$\frac{\varepsilon}{8}$
$\frac{1}{4}$
220d
121d
040


211s
$\frac{(\delta+\frac{\varepsilon}{2})}{4}$
121s
$\frac{(\delta+\frac{\varepsilon}{2})}{4}$
112s
$\frac{\varepsilon}{8}$
$\frac{\varepsilon}{8}$
$\frac{1}{4}$
$\frac{1}{4}$
211d
112d
130
031


211s
$\frac{1}{4}$
121d
$\frac{1}{4}$
112s
$\frac{1}{4}$
$\frac{1}{4}$
130
031


202s
$\frac{1}{8}$
112s
$\frac{1}{8}$
103
$\frac{\phi}{4}$
$\frac{1}{4}$
$\frac{\phi}{4}$
121s
022s
121d


202d
$\frac{\varepsilon}{8}$
112d
$\frac{1}{4}$
103
$\frac{1}{2}$
$\frac{1}{4}$
121s
022d

$$\begin{tikzcd}
112s & 103 \arrow[l, "\frac{\varepsilon}{2}"'] \arrow[r, "\frac{1}{4}"] \arrow[dl, "\frac{\varepsilon}{2}"'] & 013 \\
112d & &
\end{tikzcd}$$

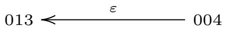
$$\begin{tikzcd}
130 & 040 \arrow[l, "\frac{\varepsilon}{2}"'] \arrow[r, "\frac{\varepsilon}{2}"] & 031
\end{tikzcd}$$

$$\begin{tikzcd}
121s & 031 \arrow[l, "\frac{\varepsilon}{4}"'] \arrow[r, "\frac{(\delta+\frac{\varepsilon}{2})}{2}"] \arrow[dl, "\frac{\varepsilon}{8}"'] \arrow[dr, "\frac{\varepsilon}{8}"] \arrow[ddl, "\frac{1}{4}"'] & 022s \\
121d & & 022d \\
040 & &
\end{tikzcd}$$

$$\begin{tikzcd}
112s & 022s \arrow[l, "\frac{\varepsilon}{4}"'] \arrow[r, "\frac{(\delta+\frac{\varepsilon}{2})}{2}"] \arrow[dl, "\frac{\phi}{2}"'] & 013 \\
031 & &
\end{tikzcd}$$

$$\begin{tikzcd}
112d & 022d \arrow[l, "\frac{\varepsilon}{4}"'] \arrow[r, "\frac{1}{2}"] \arrow[dl, "\frac{1}{2}"'] & 013 \\
031 & &
\end{tikzcd}$$

$$\begin{tikzcd}
103 & 013 \arrow[l, "\frac{\varepsilon}{8}"'] \arrow[r, "\frac{1}{2}"] \arrow[dl, "\frac{\phi}{8}"'] \arrow[ddl, "\frac{\varepsilon}{4}"'] & 004 \\
022s & & \\
022s & &
\end{tikzcd}$$



013
ε
004

Appendix B

Source Code for Simulations

CampbellModel.class

```
package dissonance;

import java.awt.Color; import java.util.*; import
uchicago.src.reflector.ListPropertyDescriptor; import
uchicago.src.sim.analysis.*; import uchicago.src.sim.engine.*;
import uchicago.src.sim.gui.*;
//import uchicago.src.sim.space.*;
import uchicago.src.sim.util.Random;

/**
    This implements in Repast J the modelling Leon Festinger's theory of cognitive dissonance
    in social networks.  Developed from AJ Bertie's version of the Axelrod culture model (1997).
    See http://repast.sourceforge.net/ for more information and the original source code.

    @version 1.0
    @author Alexander Campbell.
*/
public class CampbellModel extends SimModelImpl {

    private ArrayList agentList = new ArrayList();
    private DisplaySurface dsurf;   // Handles the drawing of the grid and creation of movies.
    private OpenSequenceGraph graph;    // A graph that plots numbers of regions and zones.
    //private OpenHistogram systemStateCount;
    private OpenSequenceGraph individualGraph;

    private int regionCount, zoneCount; // Current number of regions & zones.
    private RegionCounter regionCounter;    // Object that counts regions/zones.
    private DataRecorder recorder;  // Use a DataRecorder object to record any relevant data - writes to file.
    private Schedule schedule;
private Grid space; // The agents operate in a grid space defined by this object.
private long start; // For calculating elapsed time.


// Model parameters and their default values
// ----------------------------------------
/** The number of features possessed by each agent. */
protected int featureCount = 1;

/** Height of territory.    */
protected int gridHeight = 10;

/** Width of territory. */
protected int gridWidth = 10;


/** Size of neighbourhood. */
protected int neighbourhoodExtent = 1;

/** Type of neighbourhood. */
protected int neighbourhoodType = Grid.CLOSEST_NEIGHBOR;

/** If true, the territory "wraps around" so that no agent is on an edge. */
protected boolean torus = false;

/** The number of traits possessed by each feature. */
protected int traitCount = 10;  // change this to an array to allow different numbers of traits.

//New Parameters, errors in neighbor choice and error in decisions...

protected double errorNeighborProbability = 0.0;
protected double errorChoiceProbability = 0.0;
protected int[] sysDistribution = new int[traitCount];//+1 to keep track of system segregation
protected int[] sysNonConverge = new int[2];

// Simulation control parameters and their default values
// ----------------------------------------------------
/** Number of ticks between updates of display. */
protected int displayInterval = 1000;

/** If true, load GUI elements. */
protected boolean loadGui = true;

/** Number of ticks between output of data to Output window.    */
protected int outputInterval = 1000;

private long end = 10000;

protected long convergenceTime = 0;
protected boolean convergence;
```

```
public CampbellModel() {
    Hashtable h1 = new Hashtable();
    h1.put(new Integer(Grid.VON_NEUMANN), "Von Neumann");
    h1.put(new Integer(Grid.MOORE), "Moore");
    h1.put(new Integer(Grid.GLOBAL_UNIFORM), "Global uniform");
    h1.put(new Integer(Grid.CLOSEST_NEIGHBOR), "Closest Neighbor");
    h1.put(new Integer(Grid.QUANTAL_NEIGHBOR), "Quantal Neighbor");
    ListPropertyDescriptor pd = new ListPropertyDescriptor("NeighbourhoodType", h1);
    descriptors.put("NeighbourhoodType", pd);
    addSimEventListener(new PauseListener());
}

// get/set methods allowing for graphical and batch manipulation of the model & simulation parameters.
// ------------------------------------------------------------------------------------------------
public int getDisplayInterval() { return displayInterval; }
public void setDisplayInterval(int newDisplayInterval) { displayInterval = newDisplayInterval; }
public int getFeatureCount() { return featureCount; }
public void setFeatureCount(int newFeatureCount) {featureCount = Math.max(1,newFeatureCount);}
public int getGridWidth() { return gridWidth; }
public void setGridWidth(int newGridWidth) { gridWidth = newGridWidth; }
public int getGridHeight() { return gridHeight; }
public void setGridHeight(int newGridHeight) { gridHeight = newGridHeight; }
public boolean isLoadGui() {return loadGui;}
public void setLoadGui(boolean b) {this.loadGui = b;}

public int getNeighbourhoodExtent() {return neighbourhoodExtent;}
public void setNeighbourhoodExtent(int neighbourhoodExtent) {this.neighbourhoodExtent = neighbourhoodExtent;}
public int getNeighbourhoodType() {return neighbourhoodType;}
public void setNeighbourhoodType(int neighbourhoodType) {this.neighbourhoodType = neighbourhoodType;}
public int getOutputInterval() { return outputInterval; }
public void setOutputInterval(int newOutputInterval) { outputInterval = newOutputInterval; }
public int getTraitCount() { return traitCount; }
public void setTraitCount(int newTraitCount) { traitCount = newTraitCount; }
public boolean isTorus() { return torus; }
public void setTorus(boolean b) { torus = b; }
public long getEnd() {return end;}
public void setEnd(long e){end = e;}
public int[] getSysDistribution() {return sysDistribution;}
public void setSysDistribution(int[] sd) {sysDistribution = sd;}
public int[] getSysOutOfEq() {return sysNonConverge;}
public void setSysOutOfEq(int[] ooe) {sysNonConverge = ooe;}
public long getConvergenceTime() {return convergenceTime;}
public void setConvergenceTime(long time) {convergenceTime = time;}
public boolean getConvergence() {return convergence;}
public void setConvergence(boolean c) {convergence = c;}

//New methods, with error probabilities
public double getErrorNeighborProbability() {return errorNeighborProbability;}
public void setErrorNeighborProbability (double newErrorNeighborProbability) {
    errorNeighborProbability = newErrorNeighborProbability;
    if(errorNeighborProbability < 0) errorNeighborProbability = 0;
    else if(errorNeighborProbability > 1) errorNeighborProbability = 1;}

public double getErrorChoiceProbability() {return errorChoiceProbability;}
public void setErrorChoiceProbability (double newErrorChoiceProbability) {
    errorChoiceProbability = newErrorChoiceProbability;
    if(errorChoiceProbability < 0) errorChoiceProbability = 0;
    else if(errorChoiceProbability > 1) errorChoiceProbability = 1;}

class agentTrait implements BinDataSource{
    public double getBinValue(Object o) {
      CampbellAgent ca = (CampbellAgent)o;
      return (double)ca.getFirstTrait();
    }
  }

public String getStringSysDistribution() {
    StringBuffer sb = new StringBuffer(18 +(traitCount)*2); // Intitial Size??
    sb.append("System Distribution");
    for(int j = 0; j < sysDistribution.length; j++ ) {
        sb.append(","+sysDistribution[j]);
        }
    return sb.toString();
}
public String getStringSysOutOfEq() {
    StringBuffer sb = new StringBuffer(7 +(2)*2);   // Intitial Size??
    sb.append("OutOfEq");
    for(int j = 0; j < sysNonConverge.length; j++ ) {
        sb.append(","+sysNonConverge[j]);
        }
    return sb.toString();
}



public double getAverageTrait(){
    double average;
    double sum = 0;
    for (int i=0; i < agentList.size(); i++){
        CampbellAgent current = (CampbellAgent)agentList.get(i);
        sum += current.getFirstTrait();
    }
    average = sum/agentList.size();
    return average;
}

public double getSumGlobalTraits(){
    double sum = 0;
    for (int i=0; i < agentList.size(); i++){
        CampbellAgent current = (CampbellAgent)agentList.get(i);
```

```
            sum += current.getFirstTrait();
        }
        return sum;
    }

    /**
        Begins a simulation run. All initialization, building the model, display, etc. should take
        place here. This method is called whenever the Start button (or the Step button if the run
        has not yet begun) is clicked.
        If running in batch mode this is called to kick off a new simulation run.
    */
    public void begin() {
        buildModel();
        if (loadGui) {
            buildDisplay();
            graph.display();
            //systemStateCount.display();
            individualGraph.display();

        }
        buildSchedule();
        if (dsurf != null && loadGui)
            dsurf.display();
        start = System.currentTimeMillis(); // for timing.
    }

    // Builds the display
    private void buildDisplay() {
        Object2DDisplay agentDisplay = new Object2DDisplay(space);
        agentDisplay.setObjectList(agentList);

        dsurf.addDisplayableProbeable(agentDisplay, "Agents");
        dsurf.setBackground(java.awt.Color.white);
        addSimEventListener(dsurf);

        // Set up graph.
        graph.addSequence("Regions", new Sequence() {
            public double getSValue() { return regionCount; }},
            Color.red, OpenSequenceGraph.FILLED_CIRCLE);
        graph.addSequence("Zones", new Sequence() {
            public double getSValue() { return zoneCount; }},
            Color.blue, OpenSequenceGraph.CIRCLE);
        graph.setAxisTitles("Time", "Counts");
        graph.setXRange(0, 50000);
        graph.setYRange(0, agentList.size());
        graph.setSize(600, 400);
        //systemStateCount.createHistogramItem("System State Count",sysDistribution,new agentTrait());
        individualGraph = new OpenSequenceGraph("Individual statistics", this);
        for (Iterator iter = agentList.iterator(); iter.hasNext(); ) {
            CampbellAgent ca = (CampbellAgent) iter.next();
            individualGraph.createSequence(ca.getFirstTrait() + " Trait",
                                    ca, "getFirstTrait");
        }
        registerMediaProducer("individual graph", individualGraph);
    }

    /**
        Builds the model. Called from begin().
    */
    protected void buildModel() {
        int i;

        // Get the displayed random seed value from the RePast Parameters panel and use it to initialize the random number generator.
        BaseController controller = (BaseController) this.getController();
        long seed = controller.getRandomSeed();
        this.setRngSeed(seed);  // same effect as Random.setSeed(seed).

        Random.createUniform(); // Creates Colt object: static cern.jet.random.Uniform uniform.
        space = new Grid(gridWidth, gridHeight, torus, neighbourhoodType, neighbourhoodExtent, errorNeighborProbability);
        AgentColour siteColour = new AgentColour(featureCount, traitCount); // Create object for colouring the agents.

        int[] randomTraits = new int[featureCount];
        for (int x = 0; x < gridWidth; x++) {
            for (int y = 0; y < gridHeight; y++) {

                // Create random trait values for the agent.
                for (i = 0; i < featureCount; i++) {
                    randomTraits[i] = Random.uniform.nextIntFromTo(0, traitCount-1);    // Colt method.
                }

                // Create the agent and add it to the space.
                CampbellAgent agent = new CampbellAgent(x, y, space, featureCount, traitCount, randomTraits,
                        /*negate,*/ siteColour, errorChoiceProbability );
                agentList.add(agent);
                space.putObjectAt(x, y, agent);
            }
        }

        //sets all of the values for sysDistribution equal to 0
        for (int j = 0; j< sysDistribution.length; j++)
        {sysDistribution[j]=0;}
        for (int j = 0; j< sysNonConverge.length; j++)
        {sysNonConverge[j]=0;}

        convergenceTime = 0;

        regionCounter = new RegionCounter(featureCount, agentList, space);  // Create the object that counts the regions and zones.

        initDataRecorder();
```

```
}

// Builds the simulation schedule. Called from begin().
private void buildSchedule() {
    schedule.scheduleActionBeginning(0, new Interaction() );    // Core interaction of the Campbell model is executed at every tick.

    CountAction countAction = new CountAction();
    OutputAction outputAction = new OutputAction();
    StopAction stopAction = new StopAction();

    // Create ActionGroup to ensure that regions/zones are counted before outputted.
    ActionGroup actionGroup = new ActionGroup();
    actionGroup.addAction(countAction);
    if(loadGui) {
        actionGroup.addAction(outputAction);
        actionGroup.addAction(new BasicAction() {
            public void execute() {
                graph.step();
                individualGraph.step();
                }} );
    }
    class CampbellUpdateSysDistribution extends BasicAction {
          public void execute(){
              CampbellAgent agent;
              double val;
              boolean same;
              agent = (CampbellAgent) agentList.get(0);
              val = agent.getFirstTrait();
              same = true;

              for (int i=0; i < agentList.size();i++){
                  agent = (CampbellAgent) agentList.get(i);
                  if (val != agent.getFirstTrait()){
                      same = false;
                  }
              }

              if (same == false){
                  if (space.agentsOutOfEq()==0){
                      sysNonConverge[0]+=1;
                      System.out.println("Perfect Segregation");

                  }
                  else{
                      sysNonConverge[1]+=1;
                      System.out.println("Unstable...");
                  }
                  /*agentTraitDistribution.step();*/
                  }
              else{
                  int x;
                  System.out.println("Perfect Convergence...");
                  if (convergenceTime ==0){
                      convergenceTime = (long)getTickCount();
                  }

                  x = agent.getFirstTrait();
                  sysDistribution[x] +=1;
                  convergence = true;
              }
          }
        }

    schedule.scheduleActionAtInterval(1, new CampbellUpdateSysDistribution());

    actionGroup.addAction(stopAction);

    schedule.scheduleActionAt( 1, actionGroup, 1 );
    schedule.scheduleActionAtInterval( outputInterval, actionGroup );
    schedule.scheduleActionAtEnd(recorder, "record");
    schedule.scheduleActionAtEnd(recorder, "writeToFile");
}



public String[] getInitParam() {
    // Note order of strings determines non-alpha order of parameters in Repast model settings window.
    String[] params = { "gridWidth", "gridHeight", "torus", "neighbourhoodType", "neighbourhoodExtent",
        "featureCount", "traitCount", "errorNeighborProbability",
        "errorChoiceProbability", "displayInterval", "outputInterval", "loadGui", "End" };
    return params;
}

public String getName() { return "Campbell model"; }

public int getRegionCount() { return regionCount; }

public Schedule getSchedule() { return schedule; }

/**
    Returns all trait values of all agents in a single string.
*/
public String getTraits() {
    int n = agentList.size();
    CampbellAgent agent;
    StringBuffer sb = new StringBuffer(13+n*(1+featureCount*2));    // Intitial Size??
    sb.append("\nTrait values");
    for(int i = 0; i < n; i++ ) {
        agent = (CampbellAgent) agentList.get(i);
        sb.append("\n"+agent.traitsToString());
    }
    return sb.toString();
}

public int getZoneCount() { return zoneCount; }

/** Writes simulated data to file.  */
protected void initDataRecorder() {
    String header = "Campbell model\nRandom seed: "+getRngSeed();
    recorder = new DataRecorder("./models/Campbell.txt", this, header );
    //recorder.createNumericDataSource("regions", this, "getRegionCount", -1, -1);
    //recorder.createNumericDataSource("zones", this, "getZoneCount", -1, -1);
    recorder.createNumericDataSource("Average Trait", this, "getAverageTrait", -1, 2 );
    recorder.createNumericDataSource("Convergence Time", this, "getConvergenceTime");
    //if(loadGui)
        // Write trait values of all agents to file for statistical analysis.
        //{recorder.createObjectDataSource("", this, "getTraits" );}
    //recorder.createObjectDataSource("Trait List", this, "getTraits");
    recorder.createObjectDataSource("Convergence Distribution", this, "getStringSysDistribution");
    recorder.createObjectDataSource("OutOfEq Distribution", this, "getStringSysOutOfEq");
}

/**
    Called whenever the Setup Model button (2 arrows) is clicked.
    Also called when the model is first loaded.
*/
public void setup() {
    if (dsurf != null) {dsurf.dispose();}

    dsurf = null;
    schedule = null;
    if (graph != null) graph.dispose();
    graph = null;
    /*if (systemStateCount != null){
          systemStateCount.dispose();
        }
    systemStateCount = null;*/
    if (individualGraph != null)
            individualGraph.dispose();

    individualGraph = null;
    System.gc();

    String displayTitle = "Campbell Model Display";
    String plotTitle = "Campbell Time Series Plot";
    if (loadGui) {
        dsurf = new DisplaySurface(this, displayTitle);
        this.registerDisplaySurface(displayTitle, dsurf);
        graph = new OpenSequenceGraph(plotTitle, this);
        this.registerMediaProducer(plotTitle, graph); // ??
        //systemStateCount = new OpenHistogram("System State Count", traitCount+2, 0);
    }
    schedule = new Schedule(1);
    agentList = new ArrayList();
    space = null;
    recorder = null;
    convergence = false;

}

public static void main(String[] args) {
    SimInit init = new SimInit();
    CampbellModel model = new CampbellModel();
    init.loadModel(model, null, false);
}

// Counts the number of regions and zones.
class CountAction extends BasicAction {
    public void execute() { // Count the number of regions and zones.
        regionCount = regionCounter.countRegions();
        zoneCount = regionCounter.countZones();
    }
}

/*
    This implements the core interaction of the Campbell model.
    An agent is picked at random and its step() method is called.
    The step() method picks a neighbouring agent at random and, with probability equal to their
    cultural similarity, the two agents interact.
    This action is executed at every tick of the simulation.
*/
class Interaction extends BasicAction {
    public void execute() {
        // Pick an agent at random and attempt to interact with a neighbour.
        int i = Random.uniform.nextIntFromTo(0, agentList.size()-1);    // Colt method.
        CampbellAgent agent = (CampbellAgent) agentList.get(i);
        boolean event = agent.step();   // interact

        // Update grid screen display as necessary.
        if(dsurf != null && loadGui ) {
            if( displayInterval > 1 ) {
                if((CampbellModel.this.getTickCount() % displayInterval) == 0) dsurf.updateDisplay();
            }
            else if(event) dsurf.updateDisplay();
        }
    }
}
```

```
    // Outputs the number of ticks, regions and zones to the Output window.
    class OutputAction extends BasicAction {
        public void execute() {
            System.out.println((long)getTickCount()+" ticks: "+regionCount+" regions, "+zoneCount+" zones");
        }
    }

    // Updates the agent display when user clicks on pause or stop button.
    class PauseListener implements SimEventListener {
        public void simEventPerformed(SimEvent evt) {
            if( evt.getId() == SimEvent.PAUSE_EVENT || evt.getId() == SimEvent.STOP_EVENT)
                if (dsurf != null && loadGui)
                    dsurf.updateDisplay();
        }
    }

    /*
        Stops the simulation when the model converges to stable regions (zones).
        The final number of regions/zones is output.
        The elapsed time is calculated and output.
    */
    class StopAction extends BasicAction {
        public void execute() {
            if(getTickCount() > end  /*||zoneCount == 1 regionCount*/) {//changed this!
                System.out.println((long)getTickCount()+" ticks: "+regionCount+" regions, "+zoneCount+" zones");
                long stop = System.currentTimeMillis();
                System.out.println("Ended: elapsed time = "+(stop-start)/1000+" secs");
                stop();
                }
        }
    }
}
```

CampbellAgent.class

```
package dissonance;

import java.awt.*; import java.util.*;
//import javax.swing.*;

//import uchicago.src.reflector.DescriptorContainer;
import uchicago.src.sim.gui.Drawable; import
uchicago.src.sim.gui.SimGraphics;
//import uchicago.src.sim.space.*;
//import uchicago.src.sim.util.SimUtilities;
import uchicago.src.sim.util.Random;

/**
    This class represents an agent in the cognitive dissonance model developed with reference to AJ Berties's
    version of the Axerlod.
    An agent is an individual in the grid.

    @version 1.0
    @author Alexander Campbell.
*/
public class CampbellAgent implements Drawable {

    private int featureCount; // Number of cultural features.
    private AgentColour siteColour; // Object that determines the colour of the displayed agent.
    private Grid space; // The grid in which the agents are situated.
    private int[] traitCount;   // The number of traits possessed by each feature. Allows for variable
                                // numbers of traits, but all same for time being.
    private int[] traits;   // The array of current trait values of the features.
    private double traitsum;

    private double errorChoiceProbability;

    /** Indicates whether this site has been processed in analysing regions and zones.  */
    protected boolean done; // N.B accessed directly from RegionCounter class.

    /** Agent's grid coordinates.   */
    protected int x, y; // N.B accessed directly from RegionCounter class.

    /**
        Create the agent.

        @param x, y the grid coordinates for agent.
        @param space the grid in which the agents are situated.
        @param featureCount number of cultural features possessed by the agent.
        @param traitCount number of traits possessed by all features.
        @param initialTraits initial set of trait values.
        @param siteColour the object that determines the colour of the displayed agent.
        @param traitsum adds up all of the traits accross features
    */
    public CampbellAgent(int x, int y, Grid space, int featureCount, int traitCount,
            int[] initialTraits, AgentColour siteColour, double errorChoiceProbability ) {
        this.x = x;
        this.y = y;
        this.featureCount = featureCount;
        this.space = space;
        this.siteColour = siteColour;
        this.errorChoiceProbability = errorChoiceProbability;
        this.traitCount = new int[featureCount];
        this.traits = new int[featureCount];
        this.traitsum = 0;
        for( int i = 0; i < featureCount; i++ ) {
            this.traits[i] = initialTraits[i];
            this.traitCount[i] = traitCount;
            traitsum += this.traits[i];
        }
    }

    /**
        Implements Drawable interface. Agents are drawn with different colors to identify their
        current combination of trait values (culture).
    */
    public void draw(SimGraphics g) {
        Color c = siteColour.getColour(traits);
        g.drawFastRect(c);
    }

    // get/set methods allowing the agent's state to be probed.
    // --------------------------------------------------------
    public int getX() {return x;}
    public int getY() {return y;}
    public String getTraits() {return traitsToString();}
    public void setTraits(String newTraits) {
        boolean traitsChanged = false;
        int i, t;
        String s;
        StringTokenizer st = new StringTokenizer( newTraits, " ,\t\n\r\f" );
        if(st.countTokens() != featureCount ) {
            System.out.println("Incorrect number of traits.");
            return;
        }
        int[] nt = new int[featureCount];
        for( i = 0; i < featureCount; i++ ) {
            s = st.nextToken();
            try {
                t = Integer.parseInt(s);
                if( t < 0 || t >= traitCount[i] ) {
                    System.out.println("Trait value "+t+" is invalid.");
                    return;
                }
                nt[i] = t;
            } catch(NumberFormatException ex) {
                System.out.println("Trait value "+s+" is invalid.");
                return;
            }
        }
        for( i = 0; i < featureCount; i++ ) {
            if(traits[i] != nt[i]) {
                traitsChanged = true;
                traits[i] = nt[i];
            }
        }
        if(traitsChanged)
            System.out.println("Site ("+x+","+y+") traits changed to "+newTraits+".");
    }
```

```
/**
    Returns the number of features in this site that are different from the specified site.
    (Used to count regions.)

    @param site another site.
    @return cultural distance.
    @throws IllegalArgumentException if sites have different numbers of features.
*/
public int distance(CampbellAgent site) {
    int count = 0;
    if(this.featureCount != site.featureCount)
        throw new IllegalArgumentException("Incompatible sites.");
    for( int i = 0; i < featureCount; i++ )
        if(this.traits[i] != site.traits[i]) ++count;
    return count;
}

public double totalDistance(CampbellAgent site) {
    double distance = 0;
    if(this.featureCount != site.featureCount)
        throw new IllegalArgumentException("Incompatible sites.");
    for( int i = 0; i < featureCount; i++ ){
        distance += Math.abs(this.traits[i] - site.traits[i]);
        }
    return distance;
}

/**
    Convergent interaction with a neighbouring site.
    Select at random a feature on which this agent and its neighbour
    differ (if there is one) and change this agent's trait on this feature
    to the neighbour's trait on this feature.

    @param neighbour a neighbouring agent.
    @return true if a change took place; false otherwise.
*/
private boolean interact( CampbellAgent neighbour ) {
    /*
        Method incorporating two different way in which agents may interact.  Either by the
        Axelrod method of finding a neighbour with whom the agent has traits in common, or by forcing
        an interaction as in the CampbellModel.
    */
    int bitCount=1; // count so as to give up when all bits tried.
    int bitTry;     // bit being tried looking for dissimilarity.
    int neighbourTrait;         // A trait value of neighbour.
    int model;      //The model type defined by the network parament

    model = space.getNeighbourhoodType();
    bitTry = Random.uniform.nextIntFromTo(0,featureCount-1);
    neighbourTrait = neighbour.traits[bitTry];
    /*This first runs a check to see what model type we are working with,
     * For models >=3, the model then logs the current trait under examination
     * then we allow the agent to reduce dissonance if it is optimal
     * noting that getMove checks for an error and returns a wrongMove, or wrong choice
     * if the error threshold is passed.
     * Finally, if there is no change in the trait, we report false, otherwise true
     *
     * */
    if ( model == 3 || model == 4){//end around to check for new model types
        double diff, oldTrait;
        diff = traits[bitTry] - neighbourTrait;
        oldTrait = traits[bitTry];
        traits[bitTry] += this.getMove(diff, bitTry);
        if (oldTrait == traits[bitTry]){
            return false;
        }
        else{
            return true;
        }
    }
    else{
        bitTry = Random.uniform.nextIntFromTo(0,featureCount-1);    // Colt method.
        do  {
            neighbourTrait  = neighbour.traits[bitTry];
            if( traits[bitTry] != neighbourTrait  ) {
                // Converges since unequal on current bit.
                traits[bitTry] = neighbourTrait ;
                return true;    // done with search.
            } else {
                if(++bitCount > featureCount)
                    return false;   // give up because just tried all bits.
                bitTry = (bitTry + 1) % featureCount;
            }
        } while(true);
    }
}

/**
    Returns the number of characters in an integer number.
    The sign of the number is included.

    @return the length of n.
*/
private static int lengthOf(int n)
    { return(Integer.toString(n).length()); }



/**
    Executes one step of the RePast simulation corresponding to one RePast tick.
    This method only attempts one interaction event, which may not require updating the display etc.

    @return true if an interaction took place; false otherwise.
*/
public boolean step() {
    int bit;
    int model;
    CampbellAgent neighbour;
    model = space.getNeighbourhoodType(); //used to check model type

    if ( model == 3 || model == 4){
        //System.out.println("Closest or Quantal working");
        neighbour = (CampbellAgent) space.getNeighbour(x,y); //interact with neighbour determined by model type
    return (interact (neighbour));
    }
    else{
        bit = Random.uniform.nextIntFromTo(0,featureCount-1);   // Randomly choose a feature.
        neighbour = (CampbellAgent) space.getNeighbour(x,y);    // Randomly choose a neighbour.
        return(interact( neighbour ));
        }
}

/** Return the current trait values as a string of numbers. */
public String traitsToString() {
    StringBuffer sb = new StringBuffer(featureCount*2);
    for( int i = 0; i < featureCount; i++ ) {
        for( int j = 0; j < lengthOf(traitCount[i]-1)-lengthOf(traits[i]); j++ )
            sb.append(" ");
        sb.append(traits[i]+" ");
    }
    return sb.toString();
}

public int getFirstTrait(){
    return traits[0];
}

public double getTraitSum(){
    double traitsum = 0;
    for (int i = 0 ; i < this.featureCount; i++){
        traitsum += this.traits[i];
        }
    return traitsum;
}

//This method returns, 1,0 or -1 depending on a positive or negative difference between agents and an error check
public double getMove (double diff, int featureID){
    double errorChoiceProbability; //used to find out where the neighbour’s value is
    boolean check;
    errorChoiceProbability = this.errorChoiceProbability;
```

```
            //CampbellModel.getErrorChoiceProbability();
        check = space.errorCheck(errorChoiceProbability);
        if (check){return this.getWrongMove(diff, featureID);}
        else{return this.getRightMove(diff);}
    }

    public double getRightMove(double diff){
        if (diff > 0){return -1;}
        else{
            if (diff == 0){return 0;}
            else{return 1;}}
    }

    public double getWrongMove(double diff, int featureID){
        int trait;
        double rightMove, wrongMove;
        rightMove = this.getRightMove(diff);
        trait = this.traits[featureID];


        /*This looks messy, but it basically finds a random WrongMove move which is different from rightMove
         * The nested ifs are a function of ensuring the random move does not
         * place the trait outside the bounds of the given model parameters, defined by
         * [0, traitCount()] where traitCount is the number of potential political ideologies
         * */
        if (trait == 0){
            do  {wrongMove = Random.uniform.nextIntFromTo(0,1);}
            while(rightMove == wrongMove);}
        else{
            if (trait == (this.traitCount[featureID]-1)){
                do  {wrongMove = Random.uniform.nextIntFromTo(-1,0);}
                while(rightMove == wrongMove);}
            else{
                do  {wrongMove = Random.uniform.nextIntFromTo(-1,1);}
                while(rightMove == wrongMove);}}
        return wrongMove;
    }

}
```

Grid.class

```
package dissonance;

import java.awt.*; import java.util.*;
//import java.lang.math;


import uchicago.src.collection.*; import uchicago.src.sim.space.*;
import uchicago.src.sim.util.Random;
```

```
/**
    This class represents a grid of agents and their neighbourhoods.
    It is essentially a wrapper for RePast's <CODE>Object2DGrid</CODE> or <CODE>Object2DTorus</CODE>
    class that allows different types of neighbourhood.

    @version 1.0
    @author Campbell.
*/
public class Grid implements Discrete2DSpace {

    /** Defines the neighbourhood to be all other agents with equal probability of being chosen.    */
    public static final int GLOBAL_UNIFORM = 2; // N.B. VON_NEUMANN & MOORE are 0 & 1 respectively.
    public static final int CLOSEST_NEIGHBOR = 3;
    public static final int QUANTAL_NEIGHBOR = 4;

    protected Object2DGrid space;
    protected int neighbourhoodExtent;
    protected int neighbourhoodType;
    protected double errorNeighborProbability;

    private Vector neighbourhood;

    /**
        Create the grid.

        @param gridWidth, gridHeight the width and height of the grid.
        @param torus if true, the grid wraps around.
        @param neighbourhoodType the type of neighbourhood: VON_NEUMANN , MOORE or GLOBAL_UNIFORM.
        hopefully CLOSEST_NEIGHBOR and QUANTAL_NEIGHBOR as well
        @param neighbourhoodExtent the radius of the neighbourhood.
        @throws IllegalArgumentException if invalid neighbourhoodType.
    */
    public Grid(int gridWidth, int gridHeight, boolean torus, int neighbourhoodType, int neighbourhoodExtent,
            double errorNeighborProbability)
        {
        space = ( torus ? new Object2DTorus(gridWidth, gridHeight) : new Object2DGrid(gridWidth, gridHeight) );
        if( neighbourhoodType < 0 || neighbourhoodType > 5/*!!!!for CLOSEST.QUANTAL NEIGHBOR*/)
            throw new IllegalArgumentException("Invalid neighbourhood type.");
        this.neighbourhoodType = neighbourhoodType;
        this.neighbourhoodExtent = neighbourhoodExtent;
        this.errorNeighborProbability = errorNeighborProbability;
    }

    /**
        Chooses a neighbour depending on the type of grid.

        @param x, y the coordinates of the active agent.
        @return a randomly chosen neighbour of the active agent.
        CLOSEST_NEIGHBOR chooses the closest neighbour in terms of traits
        QUANTAL_NEIGHBOR probabilistically chooses a neighbour based on the quantal weighting system
```

```
*/
public Object getNeighbour(int x, int y) {
    switch(neighbourhoodType) {
        case VON_NEUMANN:
            neighbourhood = space.getVonNeumannNeighbors(x, y, neighbourhoodExtent, neighbourhoodExtent, false);
            return(neighbourhood.get(Random.uniform.nextIntFromTo(0, neighbourhood.size()-1)));
        case MOORE:
            neighbourhood = space.getMooreNeighbors(x, y, neighbourhoodExtent, neighbourhoodExtent, false);
            return(neighbourhood.get(Random.uniform.nextIntFromTo(0, neighbourhood.size()-1)));
        case GLOBAL_UNIFORM:
            int jx, jy; // Coordinates of neighbouring site.
            // Randomly choose any site in the territory that is not the active site.
            do  {
                jx = Random.uniform.nextIntFromTo(0, space.getSizeX()-1);
                jy = Random.uniform.nextIntFromTo(0, space.getSizeY()-1);
            } while( jx == x && jy == y );
            return(space.getObjectAt(jx,jy));
        case CLOSEST_NEIGHBOR:
            /*This method first checks to see if the agent makes an error in finding closest neighbour
             * If there is no neighbour error, ie, errorCheck returns false, then this simply returns the closest
             * neighbour available to the agent in question.  Otherwise the method has to find a neighbour
             * who has a political ideology different from that of the closest neighbour.  Remember that
             * neighbour error means agents specifically chooses a non-optimal neighbour to interact with.
             * If there were two neighbours with the same ideology close to the player in question, then a
             * neighbour error should return neither!
             * */
            neighbourhood = space.getVonNeumannNeighbors(x, y, neighbourhoodExtent, neighbourhoodExtent, false);

            /*This checks first to see all of the agent's neighbours have the same political ideology
            /If all neighbours do have the same ideology, it matters not which one is returned, so choose random.*/
            if (this.checkSameVal(neighbourhood) == true){
                int location = Random.uniform.nextIntFromTo(0,neighbourhood.size()-1);
                return neighbourhood.get(location);
            }
            else{
                int location, wrongLocation;
                boolean neighbourErrorCheck;
                location = this.getClosestLocation(x, y, neighbourhood);

                neighbourErrorCheck =  this.errorCheck(errorNeighborProbability);//Check to see if error occurs.

                if (neighbourErrorCheck != true){
                    return neighbourhood.get(location);
                }
                else{
                    int rightVal, wrongVal;
                    CampbellAgent rightNeighbor, wrongNeighbor;
                    rightNeighbor = (CampbellAgent)neighbourhood.get(location);
                    rightVal = rightNeighbor.getFirstTrait();
                    do{
                            wrongLocation = Random.uniform.nextIntFromTo(0,neighbourhood.size()-1);
                            wrongNeighbor = (CampbellAgent)neighbourhood.get(wrongLocation);
                            wrongVal = wrongNeighbor.getFirstTrait();
                        }while (rightVal == wrongVal);

                        return neighbourhood.get(wrongLocation);
                    }

                }



            case QUANTAL_NEIGHBOR:
                neighbourhood = space.getVonNeumannNeighbors(x, y, neighbourhoodExtent, neighbourhoodExtent, false);
                return this.getQuantalNeighbor(x, y, neighbourhood);
        }
        return null;    // to satisfy compiler.
    }

    /**
        Used for counting regions.
    */
    public Vector getVonNeumannNeighbors(int x, int y, boolean returnNulls) {
        return(space.getVonNeumannNeighbors(x, y, returnNulls));
    }

    public boolean checkSameVal(Vector neighbourhood){
        int val;
        boolean check = true;
        CampbellAgent neighbour = (CampbellAgent)neighbourhood.get(0);
        val = neighbour.getFirstTrait();

        for (int i = 0; i < neighbourhood.size(); i++){
            neighbour = (CampbellAgent)neighbourhood.get(0);
            if (val != neighbour.getFirstTrait()){
                check = false;
            }
        }
        return check;
    }

    public int getClosestLocation(int x, int y, Vector neighbourhood){
        int size, location;
        double distance;
        CampbellAgent agent, neighbour;

        size = neighbourhood.size();
        neighbour = (CampbellAgent)neighbourhood.get(0);

        //set up pointer to first neighbour in preparation for 'for' loop
        agent = (CampbellAgent)space.getObjectAt(x, y);
        distance = 10000; //Set high enough so first neighbour always gets selected
        location = 0;

        /*Loop to find closest neighbour
         *
         * If the neighbour under examination is closer, it sets the location pointer to the
         * relevant point in the vector of neighbours.
         * Had to be done this way as VonNeumannNeighbors returns different orders for agents on the end of grid
          */
        for (int i = 0; i < size; i++){
            double thisDistance;

            neighbour = (CampbellAgent)neighbourhood.get(i);
            thisDistance = agent.totalDistance(neighbour);
            if (distance <= thisDistance);
            else{
                location = i;
                distance = thisDistance;
                }
            }
        //System.out.println("neighbour is at "+location+" location");
        return location;

    }


    // Following methods implement Discrete2DSpace interface.
    public int getSizeX() {return(space.getSizeX());}
    public int getSizeY() {return(space.getSizeY());}
    public Dimension getSize() {return(space.getSize());}
    public Object getObjectAt(int x, int y) {return(space.getObjectAt(x,y));}
    public double getValueAt(int x, int y) {return(space.getValueAt(x,y));}
    public void putObjectAt(int x, int y, Object object) {space.putObjectAt(x,y,object);}
    public void putValueAt(int x, int y, double value) {space.putValueAt(x,y,value);}
    public BaseMatrix getMatrix() {return(space.getMatrix());}
    public int getNeighbourhoodType() {return neighbourhoodType;}
    public boolean errorCheck(double error) {
        double r;
        r = Math.random();
        if (r <= error){return true;}
        else {return false;}
    }

    //Finds a specific neighbour using the quantal weighting mechanism and the random weighted neighbour draw
    public Object getQuantalNeighbor (int x, int y, Vector neighbourhood){
        int size;
        size = neighbourhood.size();

        return neighbourhood.get(this.getWeightedNeighbor(size, this.getQuantalWeights(x, y, size, neighbourhood)));
```

```
}

public double[] getQuantalWeights(int x, int y, int size, Vector neighbourhood){
    double cog;
    CampbellAgent agent, neighbour;

    agent = (CampbellAgent)space.getObjectAt(x, y);
    cog = agent.getFirstTrait();

    double weights[] = new double[size];
    double distance[] = new double[size];
    double sum = 0;

    for (int i = 0; i < size; i++)
        {
        neighbour = (CampbellAgent)neighbourhood.get(i);
        distance[i] = Math.abs(cog - neighbour.getFirstTrait());

        weights[i] = Math.exp(-distance[i]);
        sum += weights[i];
        }
    for (int i = 0; i < size; i++){
        weights[i] = weights[i]/sum;
        }
    return weights;
}

/*Uses a random generator to pick an neighbours given some input of weighted pointers.
 *
 *
 */
public int agentsOutOfEq(){
    int count;
    CampbellAgent agent, neighbour;

    count = 0;

    //This gets  number for how many agents cannot find a neighbour who has the same trait as themselves
    for (int x = 0; x < getSizeX(); x++){
        for (int y = 0; y < getSizeY(); y++){
            agent = (CampbellAgent)space.getObjectAt(x, y);
            neighbourhood = space.getVonNeumannNeighbors(x, y, 1, 1, false);
            neighbour = (CampbellAgent)neighbourhood.get(getClosestLocation(x, y, neighbourhood));

            if (agent.getTraitSum() != neighbour.getTraitSum()){count+=1;}

            }

        }
        return count;
    }

    public int getWeightedNeighbor (int size, double weights[]){
        int location = 0;
        double lowerBound = 0;
        double upperBound = 0;
        double r = Math.random();

        for(int i = 0; i < size; i++){
            upperBound+= weights[i];
            if(r <= upperBound && r > lowerBound){
                location = i;
            }
            lowerBound = upperBound;
        }
        return location;
    }
}
```

AgentColour.class

```
package dissonance;

import java.awt.Color;

/**
    This class represents the colour of a site/agent in the Campbell model.
    The colour is determined by the trait values. The aim is that <CODE>getColour()</CODE> returns
    unique colours for different combinations of traits.

    @version 1.0
    @author Alexander Campbell.
*/
public class AgentColour {

    private int featureCount;   // Number of cultural features.
    private int[] traitCounts;  // The number of traits possessed by each feature. Allows for variable
                                // numbers of traits, but all same for time being.
    private float rmax=0, gmax=0, bmax=0;
    private int ri, gi, bi;

    /**
        Create the colouring object.

        @param featureCount number of cultural features possessed by agents.
        @param traitCount number of traits possessed by all features.
        @throws IllegalArgumentException if invalid number of features.
    */
    public AgentColour( int featureCount, int traitCount ) {
        this.featureCount = featureCount;
        this.traitCounts = new int[featureCount];
        for( int i = 0; i < featureCount; i++ ) {
            this.traitCounts[i] = traitCount;
        }

        if(this.featureCount > 3) {
            int k = Math.round(featureCount/3.0f);
            ri = k; gi = ri+k; bi = featureCount;
            rmax=0; gmax=0; bmax=0;
            int i;
            for( i = 0; i < ri; i++ ) rmax += traitCounts[i]-1;
            for( i = ri; i < gi; i++ ) gmax += traitCounts[i]-1;
            for( i = gi; i < bi; i++ ) bmax += traitCounts[i]-1;
        } else if(this.featureCount == 3) {
            rmax = traitCounts[0]-1;
            gmax = traitCounts[1]-1;
            bmax = traitCounts[2]-1;
        } else if(this.featureCount == 2) {
            rmax = traitCounts[0]-1;
            gmax = traitCounts[1]-1;
        } else if(this.featureCount == 1) {
            rmax = traitCounts[0]-1;
        } else
            throw new IllegalArgumentException("Invalid number of features.");
    }

    /**
        Return a colour that represents the set of trait values.

        @param traits the trait values.
        @return the colour.
    */
    public Color getColour(int[] traits) {
        if(featureCount > 3) {
            float r=0,g=0,b=0;  // range from 0 to 1.
            int i;
            for( i = 0; i < ri; i++ ) r += traits[i];
            for( i = ri; i < gi; i++ ) g += traits[i];
            for( i = gi; i < bi; i++ ) b += traits[i];
            return(new Color(r/rmax, g/gmax, b/bmax));
        } else if(featureCount == 3)
            return(new Color(traits[0]/rmax, traits[1]/gmax, traits[2]/bmax));
        else if(featureCount == 2) {
            float c1 = traits[0]/rmax;
            float c2 = traits[1]/gmax;
            return(new Color(c1, c2, 0.5f*(c1+c2)));
        } else {
            float c = traits[0]/rmax;
            return(new Color(c, c, c));
        }
    }
}
```

RegionCounter.class

```
package dissonance;

import java.util.*;

/**
    This class encapsulates algorithms for counting the number of regions and zones in
    an Campbell model.

    @version 1.0
    @author Alexander Campbell.
*/
public class RegionCounter {

    private Vector readyQueue = new Vector(20,10);  // ??
    private Vector myNeighborhood;

    int featureCount;
    ArrayList agentList;
    Grid space;

    /**
        Create the region counter.

        @param featureCount the number of features in the cultural model.
        @param agentList the agents in the cultural model.
        @param space the grid of agents in the cultural model.
    */
    public RegionCounter(int featureCount, ArrayList agentList, Grid space) {
        this.featureCount = featureCount;
        this.agentList = agentList;
        this.space = space;
    }

    /**
        Based on procedures Analyze_zones & Analyze_regions in Axelrod's code for his full
        cultural model (version 2.6).

        @param maxDistance if the number of features that are different exceed this,
            then sites are in different zones.
        @return number of regions that have more than maxDistance features that are different.
    */
    private int analyzeZones(int maxDistance) {
        // count number of regions, and types.
        // based on breadth first search. See Stubbs and Webre, Data Structures, 359ff.
        int i;
        int regions = 0;
    readyQueue.clear();
    for (i = 0; i < agentList.size(); i++) {
        CampbellAgent agent = (CampbellAgent) agentList.get(i);
        agent.done = false;
    }

    for (i = 0; i < agentList.size(); i++) {
        CampbellAgent agent = (CampbellAgent) agentList.get(i);
        if(!agent.done) {// if not done then record it and visit it.
            ++regions;  // count regions
            visit(agent, maxDistance);  // 0 is min distance allowed for regions.
        }
    }

    return regions;
}

/**
    Count the number of regions.

    @return the number of regions.
*/
public int countRegions() {return(analyzeZones(0));}

/**
    Count the number of cultural zones. A cultural zone is a set of contiguous sites,
    each of which has a neighbour with a "compatible" culture. Cultures are compatible
    if they have at least one feature in common.

    @return the number of zones.
*/
public int countZones() {return(analyzeZones(featureCount-1));}

/**
    Based on procedure Visit in Campbell's code for his full cultural model (version 2.6).
*/
private int visit( CampbellAgent agent, int criticalDist) {
    int regionSize = 0;
    CampbellAgent active, neighbour;

    readyQueue.add(agent);   // Add node to ready queue.
    while(!readyQueue.isEmpty()) {  // while ready queue not empty...
        // Get node from end of ready queue.
        active = (CampbellAgent) readyQueue.lastElement();
        readyQueue.removeElementAt(readyQueue.size()-1);

        // Add to ready queue the neighbours who are legal, not done, and 0 dist.
        myNeighborhood = space.getVonNeumannNeighbors(active.x, active.y, false);   // If space not torus, can return nulls?

        for(int i = 0; i < myNeighborhood.size(); i++ ) {
                neighbour = (CampbellAgent) myNeighborhood.get(i);
                if( neighbour.done ) continue;
                if(active.distance(neighbour) <= criticalDist ) //crit distance = 0 for region, bitmax-1 for zone}
                    {
                    ++regionSize;
                    neighbour.done = true;
                    readyQueue.add(neighbour);
                    }
            }
        }
        if(regionSize == 0) regionSize = 1; // Needed for 1 x 1 regions which have no valid neighbours.
        return regionSize;
    }

}
```

References

[1] R.P. Abelson.

[2] R.P. Abelson. Mathematical models of the distribution of attitudes under controversy. *Contributions to Mathematical Psychology*, pages 142–160, 1964.

[3] G.A. Akerlof. Social Distance and Social Decisions. *Econometrica*, 65(5):1005–1027, 1997.

[4] G.A. Akerlof and W.T. Dickens. The Economic Consequences of Cognitive Dissonance. *The American Economic Review*, 72(3):307–319, 1982.

[5] E. Aronson. Dissonance theory: Progress and problems. *Theories of cognitive consistency: A sourcebook*, pages 5–27, 1968.

[6] E. Aronson. The Return of the Repressed: Dissonance Theory Makes a Comeback. *Psychological Inquiry*, 3(4):303–311, 1992.

[7] E. Aronson. *The social animal*. WH Freeman, 1999.

[8] E. Aronson and J. Mills. The effect of severity of initiation on liking for a group. *Journal of Abnormal and Social Psychology*, 59:177–181, 1959.

[9] R. Axelrod. *Evolution of Cooperation*. HarperCollins Canada, Limited, 1985.

[10] R. Axelrod. The Dissemination of Culture: A Model with Local Convergence and Global Polarization. *Journal of Conflict Resolution*, 41(2):203–226, 1997.

[11] Andrew Bertie. *Axelrod's dissemination of culture model in Repast J.* World Wide Web, http://repast.sourceforge.net/contributions/AxelrodModelSources.zip, 2007.

[12] N. Collier, T. Howe, and M North. *REPAST Agent Simulation toolkit.* 2007.

[13] J. Cooper and R.H. Fazio. A new look at dissonance theory. *Advances in experimental social psychology*, 17:229–266, 1984.

[14] W.H. Cummings and M. Venkatesan. Cognitive Dissonance and Consumer Behavior: A Review of the Evidence. *Journal of Marketing Research*, 13(3):303–308, 1976.

[15] KE DAVIS and EE JONES. Changes in interpersonal perception as a means of reducing cognitive dissonance. *J Abnorm Soc Psychol*, 61:402–10, 1960.

[16] C.A. Dickerson, R. Thibodeau, E. Aronson, and D. Miller. Using cognitive dissonance to encourage water conservation. *Journal of Applied Social Psychology*, 22(11):841–854, 1992.

[17] L. Festinger. *A theory of cognitive dissonance.* Stanford University Press Stanford, Calif, 1957.

[18] Leon Festinger and J.M. Carlsmith. Cognitive consequences of forced compliance. *Journal of Abnormal and Social Psychology*, 58:203–211, 1959.

[19] J.L. Freedman. Long-term behavioral effects of cognitive dissonance. *Journal of Experimental Social Psychology*, 1:145–155, 1965.

[20] DC GLASS. Changes in liking as a means of reducing cognitive discrepancies between self-esteem and aggression. *J Pers*, 32:531–49, 1964.

[21] MF Hoyt, MD Henley, and BE Collins. Studiesin forced compliance: The confluence of choice and consequences on attitude change. *Journal of Personality and Social Psychology*, 22:1–7, 1972.

[22] R.R. Huckfeldt, J.D. Sprague, and P.E. Johnson. *Political Disagreement: the survival of diverse opinions within communication networks.* Cambridge University Press, 2004.

[23] D. Kahneman and A. Tversky. Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2):263–292, 1979.

[24] H.H. Kassarjian and J.B. Cohen. Cognitive Dissonance and Consumer Behavior: Reactions to the Surgeon Generals Report on Smoking and Health. *California Management Review*, 8:55–64, 1965.

[25] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Springer, 1976.

[26] E. Kim, A. Morse, and Zingales L. What has mattered to economics since 1970. *Journal of Economic Perspectives*, 2006.

[27] J. Konow. Fair Shares: Accountability and Cognitive Dissonance in Allocation Decisions. *The American Economic Review*, 90(4):1072–1091, 2000.

[28] C.G. Lord, L. Ross, and M.R. Lepper. Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of Personality and Social Psychology*, 37(11):2098–2109, 1979.

[29] G.J. Stigler. Economics: The Imperial Science? *The Scandinavian Journal of Economics*, 86(3):301–313, 1984.

[30] J. Stone, E. Aronson, A.L. Crain, M.P. Winslow, and C.B. Fried. Inducing Hypocrisy as a Means of Encouraging Young Adults to Use Condoms. *Personality and Social Psychology Bulletin*, 20(1):116, 1994.

[31] R. Thibodeau and E. Aronson. Taking a Closer Look: Reasserting the Role of the Self-Concept in Dissonance Theory. *Personality and Social Psychology Bulletin*, 18(5):591, 1992.

[32] A. Tversky and D. Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science*, 185(4157):1124, 1974.

[33] H.P. Young. The Evolution of Conventions. *Econometrica*, 61(1):57–84, 1993.

[34] H.P. Young. *Individual Strategy and Social Structure: an evolutionary theory of institutions*. Princeton Univ Pr, 1998.