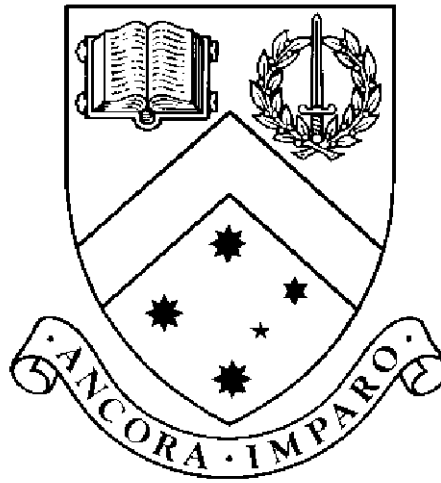


# HYPANT : A Hypergame Analysis Tool

by

Lachlan Brumley



Thesis

Submitted by Lachlan Brumley

in partial fulfillment of the Requirements for the Degree of

**Bachelor of Software Engineering with Honours (2770)**

in the School of Computer Science and Software Engineering at

Monash University

**Monash University**

November, 2003

© Copyright

by

Lachlan Brumley

2003

# Contents

<b>List of Figures</b> . . . . .	<b>v</b>
<b>List of Tables</b> . . . . .	<b>vi</b>
<b>Abstract</b> . . . . .	<b>vii</b>
<b>1 Previous Research</b> . . . . .	<b>1</b>
1.1 Game Theory . . . . .	1
1.1.1 Prisoner’s Dilemma . . . . .	2
1.2 Metagames . . . . .	3
1.3 Conflict Analysis . . . . .	3
1.4 Shortcomings of Game Theory . . . . .	4
1.5 Hypergames . . . . .	5
1.5.1 Strategic Surprise in Hypergames . . . . .	6
1.5.2 Deception in Hypergames . . . . .	7
1.6 Computerised Tools for Game Analysis . . . . .	8
<b>2 Research Methods</b> . . . . .	<b>9</b>
2.1 Specifications . . . . .	9
2.2 Implementation . . . . .	10
2.2.1 Hypergame Modelling Language (HML) . . . . .	10
2.2.2 Client Output . . . . .	13
2.2.3 Algorithm Implementation . . . . .	13
2.3 Testing . . . . .	14
2.3.1 Parser . . . . .	14
2.3.2 Analysis Algorithm . . . . .	15
<b>3 Results</b> . . . . .	<b>16</b>
3.1 Cuban Missile Crisis . . . . .	17
3.2 Operation Overlord . . . . .	18
3.3 Fall of France . . . . .	19

3.4	Garrison Diversion Unit . . . . .	20
3.5	Altered Cuban Missile Crisis 1 . . . . .	21
3.6	Altered Cuban Missile Crisis 2 . . . . .	22
<b>4</b>	<b>Conclusions . . . . .</b>	<b>23</b>
4.1	Observations . . . . .	23
4.2	Future Research . . . . .	23
	<b>Appendix A Hypergame Grammar . . . . .</b>	<b>25</b>
A.1	Initial grammar . . . . .	25
A.2	Refined Grammar . . . . .	25
	<b>Appendix B Hypergames Analysed . . . . .</b>	<b>27</b>
B.1	Cuban Missile Crisis Hypergame . . . . .	27
B.2	Operation Overlord . . . . .	28
B.3	Fall Of France . . . . .	31
B.4	Garrison Diversion Unit . . . . .	33
B.5	Altered Cuban Missile Crisis 1 . . . . .	36
B.6	Altered Cuban Missile Crisis 2 . . . . .	36
	<b>Appendix C Software Notes . . . . .</b>	<b>38</b>
C.1	Hypergame Model Data Structure . . . . .	38
C.2	Application Programming Interface . . . . .	38
C.2.1	Client . . . . .	38
C.2.2	Server . . . . .	40
C.2.3	Parser . . . . .	41
C.2.4	General . . . . .	41
C.3	Use of Colours . . . . .	42
	<b>Appendix D Screenshots . . . . .</b>	<b>43</b>
	<b>References . . . . .</b>	<b>55</b>

# List of Figures

1.1	A third level hypergame between players A and B . . . . .	6
C.1	Hypergame Model Data structure . . . . .	39
D.1	Cuban Missile Crisis Equilibriums . . . . .	43
D.2	Cuban Missile Crisis Stability Table . . . . .	44
D.3	Cuban Missile Crisis Game . . . . .	45
D.4	Fall Of France - France's Hypergame . . . . .	46
D.5	Fall Of France - Germany's Hypergame . . . . .	47
D.6	Fall of France Hypergame Analysis . . . . .	48
D.7	Garrison Analysis (1 of 2) . . . . .	49
D.8	Garrison Analysis (2 of 2) . . . . .	50
D.9	Alternate Cuban Game 1 Stability Table . . . . .	51
D.10	Alternate Cuban Game 1 Equilibriums . . . . .	52
D.11	Alternate Cuban Game 2 Stability Table . . . . .	53
D.12	Alternate Cuban Game 2 Equilibriums . . . . .	54

# List of Tables

3.1	Stability Tables for Cuban Missile Crisis (USA) . . . . .	17
3.2	Stability Tables for Cuban Missile Crisis (USSR) . . . . .	17
3.3	Equilibriums for Individual Games in Operation Overlord . . . . .	18
3.4	Equilibriums for Hypergames in Operation Overlord . . . . .	19
3.5	Equilibriums for Individual Games in Fall of France . . . . .	19
3.6	Equilibriums for Individual Games in Fall of France . . . . .	19
3.7	USSR Altered & Normal Cuban Missile Crisis Preference Vectors . . . . .	21
3.8	USA & USSR Altered Cuban Missile Crisis 2 Preference Vectors . . . . .	22
C.1	Available terminal colours . . . . .	42
C.2	Allocation of available colours to players . . . . .	42

# HYPANT : A Hypergame Analysis Tool

Lachlan Brumley, BSE(Hons)  
Monash University, 2003

Supervisor: Dr. Carlo Kopp

## Abstract

Hypergames are a method of modelling conflicts, using game theoretic methods, that can be applied when the players of a game do not have complete information about the game being played. A hypergame models an individual game as a series of games, where each one is based on a different player's perceptions of an aspect of the overall conflict.

A drawback to using hypergames to model situations is that there can be many games to analyse and the analysis of a hypergame is repetitive and tedious for a person to calculate by hand. Computers are a useful tool here, increasing both the speed and accuracy of the analysis.

There have been several programs written to help with hypergame analysis in the past, however these are either unavailable for free or for modern computer platforms. The main aim of this research is to develop a computer program to help perform hypergame analysis.

It is also apparent that a method of storing and retrieving previously analysed games would be a required functionality of the system and this implies the need for a language to describe a hypergame. A parser for such a language can be easily created using tools such as *lex* and *yacc*.

# Chapter 1

## Previous Research

### 1.1 Game Theory

Initial research into the area of game theory was undertaken by Morgenstern and von Neumann (1953) as an attempt to mathematically define a theory to make it possible to study and define “games of strategy”. These games were initially intended to provide insight into the economic behaviours of entities (individuals and companies) and reveal how these entities may maximise their “profits” or payoffs from certain situations, given some set of choices. It covers cases where multiple entities, or “players” are competing for resources of some sort. The main work of Morgenstern and von Neumann (1953) specifies a formal notation that can be used to describe most games and a theorem that states how results for the game can be analysed.

A game consists of a number of players, a series of rules that describe the game and the moves that players may make. Moves may be determined by either choices that a player can make or by random occurrences, such as the roll of a dice. A game can be written in its formal method or alternatively displayed in a matrix. For a 2 player game, this matrix has  $m \times n$  cells where  $m$  and  $n$  are the amount of moves that each player has. Each cell of the matrix contains a value that represents the payoff value that each player would receive if the two players moves index that cell in the matrix. Each player uses their decision to try to choose a value that leads to as high a value payoff as possible, but their choice is made in ignorance of the other player’s choice. The values of the payoffs are decided by the rules of the game.

Each player of a game will be seeking to maximise their payoff and minimise that of their opponent by choosing moves that reflect these goals. Since the players know of each others possible moves they can both make moves that take them towards a middle ground, where both player’s payoffs are maximised. The point is called the equilibrium or the saddle point.

The main theorem of Morgenstern and von Neumann’s (1953) Theory of Games is that for a number of games it is always possible to find an equilibrium from which neither player should deviate unilaterally from, that is there is no move they can make that will lead to a higher payoff for them. These equilibria exist for every two player game that is finite (has a finite number of moves and the gameplay ends after these moves), is a zero sum game (one player’s losses equal their opponent’s gain) and is a game of complete information (all players know their own moves and preferences and those of their opponents). This means for a certain game that meets these criteria, we can find the best possible outcome for both

players, which is a compromise between their most favoured move and their opponent's most favoured move.

One of the main restrictions of game theory is that players must act in a rational manner. That is from all the possible options a player can choose, they should choose the option that has the highest possible payoff for them over others with a lesser payoff.

In game theoretic games, it is always assumed that players will act in a rational manner, unless explicitly stated otherwise.

### 1.1.1 Prisoner's Dilemma

The Prisoner's Dilemma is a fundamental problem, that is frequently studied in game theory (Mèrö, 1998), (Howard, 1971). A description of this problem states that two criminals have been captured after together committing a serious crime. The police do not have evidence to prove that they were responsible for the serious crime, but can prove that they committed a minor crime. The police wish to close this case so they propose a deal to each of the criminals a deal - if they confess to the crime and implicate their accomplice, they will be set free and the minor charges dropped. This will lead to the other prisoner being locked up for a long time.

However this offer is only valid if the other prisoner does not confess to the crime. If both of the prisoners confess, their confessions are of no use in helping to clear up the case and they will both receive a moderate amount of time in jail. If neither prisoner confesses, then the prisoners will be charged with the minor crime and given a smaller sentence.

From these rules the game is seen to be played by the two prisoners against each other. Each prisoner has the option to either confess or to not confess. Howard (1971) has called these options as *Defect* and *Cooperate*. The reason for these names is that confessing can be seen as one prisoner defecting from his previous alliance with the other prisoner, whereas not confessing can be seen as one prisoner attempting to cooperate with the other. For the Prisoner's Dilemma game there are four different outcomes - Both prisoners defect and each spend a moderate amount of time in jail, both prisoners cooperate and each spend a small amount of time in jail each or one prisoner defects while the other tries to cooperate leading to the defector being freed and the cooperator spending a long time in jail.

For the prisoners to behave in a rational manner, they would both choose the outcome that leads to the most preferable payoff for them, that is defection and freedom. However two rational prisoners would both choose to defect, leading to a situation where they do not obtain the best payoff possible.

This is the crucial point of the Prisoner's Dilemma, if both players logically choose the option with the highest payoff for them individually - freedom - then they end up with a moderate amount of time in jail, which is for both of them, a fairly bad outcome. Paradoxically, if both prisoners cooperate and don't confess then they each receive the second highest payoff of only a small amount of time in jail for the minor charge. This strategy is not without risk however as if one prisoner cooperates while his opponent defects, then he will receive the worst payoff of a long time in prison, while his opponent is released.

Other real world situations can also be considered to be a Prisoner's Dilemma, such as an arms race between superpowers (Mèrö, 1998). Instead of confess and don't confess, the two superpowers have the options to either arm heavily or the arm moderately. If both superpowers arm heavily, then there is a balance of power, as both superpowers are heavily armed, yet these arms cost them quite a lot of money to obtain. If both superpowers choose

to arm moderately, then there is still a balance of power between the superpowers, but it comes at a much cheaper cost. If one superpower arms heavily and the other moderately, then the heavily armed superpower has a strategic advantage over the moderately armed superpower.

In this game, each superpower's best payoff can be achieved by choosing to arm heavily in the hope of achieving superiority over their moderately armed counterpart. Yet as with the Prisoner's Dilemma, two rational superpowers will both arm heavily leading to an expensive balance of power, whereas by cooperating they could have the balance of power at a lower cost.

From these examples it can be seen that the Prisoner's Dilemma is a fairly simple game that can occur in real life. An important concept that it demonstrates is that completely rational and logical actions may not always be the best action to take, depending on the circumstances.

## 1.2 Metagames

Morgenstern and von Neumann (1953) believed that games could be analysed by creating a number of other games that would only exist if any player could choose his strategy after the other players, with knowledge of their strategy. These new games, referred to as minorant and majorant games, could be analysed to find equilibriums and to gain a better understanding to the original game. This idea was later refined by Howard (1971), who called them "metagames" and applied them in a much more general approach.

For any game, it is possible to develop a metagame for each player presenting their options should they know what move their opponent will take. For a player  $n$  their metagame is referred to as  $n$ -metagame. Metagames can be described as a matrix, like the game they are derived from, but it is impractical to write this matrix for all but trivial games as it becomes quite large. If we have a metagame for a player, then that player has enough information to choose objective rational actions, that is they know enough relevant information to be able to rationally choose an option.

There is one case where a player can always make objective rational strategies and that is when for one player's strategy, no matter which strategy his opponent chooses, his strategy is still the best for him. This is called a "sure thing" and an example of it is the defect option from the Prisoner's Dilemma game. If each player's sure fire strategies overlap, the places of overlap are equilibrium points. Equilibrium points represent rational outcomes to the game for all players. The outcomes at these points maximise the payoff that players receive and it would not be in a player's best interest to choose a different option or set of options.

These equilibrium points are stable and represent solutions to the game that maximise payoffs for both players. If there are no equilibrium points for a game, then there are no stable solutions for the game. This means that one player will make a decision that is wrong and does not maximise his payoff.

## 1.3 Conflict Analysis

Conflict Analysis is the process of analysing games to find equilibriums that may be resolutions to the conflict. The conflict analysis algorithm was developed by Fraser and Hipel

(1984) and is applied to games, which are modelled using an extension of Howard's (1971) metagame model.

Using the methodology of Fraser and Hipel (1984), a conflict is considered to be a *game*, with two or more groups - referred to as *players* - in conflict over a resource or an issue. Each player has a number of actions that they may take, which are called *options*. Any set of options taken by a player is called a *strategy*. Each option is binary, in that it can be either on or off, or selected or rejected. Therefore there are  $2^n$  possible strategies that a player may take, where  $n$  is the number of options that the player has. The strategies together of all the players is called an *outcome*. For a game with  $m$  total outcomes there are  $2^m$  mathematically possible outcomes. However not all of these outcomes may be feasible. For example outcomes with two players both possessing the same strategic resource or one player both launching an attack and withdrawing from the same location would be infeasible and need to be removed from the model. Once they are removed, a *preference vector* for each player is developed, in which the remaining feasible outcomes are ranked from most desired to least desired. This completes the modelling process.

When modelling a system, it is necessary to consider only a simplified version of the real world problem. At first glance this may lead to the problem of creating too simple a model, which would be irrelevant or unusable. However according to Bennett and Huxham (1982), while a drastically oversimplified model appears to be a bad idea, it forces the creator to make specific decisions about what features to include or remove from the model and how the various aspects of the system link together. In other words it enhances the analyst's understanding of the conflict being modelled and therefore the quality of the model itself. A simple model can also be extended in the future to encompass more aspects of the real world problem.

Once a suitable model has been constructed, it is then analysed to find suitable resolutions to the problem. A suitable solution is one that is either rational or *stable* for all players in the game and this is obtained by determining the stability of each feasible outcome for each player. An outcome is considered stable if it is not advantageous for that player to change their strategy by moving to another outcome. If the change is of benefit to the player, the outcome is said to be *unstable*. If an outcome is stable for all players, then that outcome represents an *equilibrium* and constitutes a possible resolution to the conflict.

## 1.4 Shortcomings of Game Theory

While Game Theory provides us with a powerful method for constructing and analysing games it does have limitations. The main limitation is that the games all require perfect knowledge. That is all players must have a perfect understanding of what options they may take, which options their opponents can take and who their opponents are. Without this knowledge it is impossible for players of a game to make rational decisions about their moves. To be able to model situations where perfect understanding is not present a different model is required.

There have been various changes suggested to overcome this drawback. One of these is the "hypergame", which was first suggested by Bennett (1977). A hypergame instead of modelling one game, models each player's perception of the conflict as a game or a series of games. These sub-games need not be the same and in principle they need not have much in common (Bennett, Tait and MacDonagh, 1993).

Another method to allow the use of incomplete information is the work of Harsanyi (1968), in which the game is played by “Bayesian” players. Instead of the players having complete knowledge of each player’s strategies and preferences, they have a probability distribution over all the alternative possibilities.

However Bennett (1980a) is critical of the usefulness of this approach, arguing that real players are not Bayesian and trying to model them in this manner is incorrect. In a real conflict, the players don’t use probabilistic methods to determine what their opponent(s) preferences may be, so this method is difficult to use to map reality to a simpler model. According to Leclerc and Chaib-draa (2002) probabilistic approaches such as Bayesian players that introduce uncertainty into the model and are generally undesirable as the measures they use are subjective. A poor choice of values used to measure the uncertainty will lead to a highly inaccurate model.

## 1.5 Hypergames

The term “hypergame” was first used by Bennett (1977). A hypergame is model of a conflict that allows for misperceptions in each player’s view of the game being played. According to Fraser and Hipel (1984), the players in a hypergame may

1. Have a false understanding of the preferences of the other players.
2. Have an incorrect comprehension of the options available to the other players.
3. Not be aware of all the players in the game.
4. Have any combination of the above faulty interpretations.

The structure for a hypergame is defined by Wang, Hipel and Fraser (1989) as a set of individual games, where each individual game models one perspective of the overall conflict. Hypergames are said to have levels of perception (Bennett, 1980b), which allow many different perspectives can be built up, for example a player’s perspective of their opponents perspective of their own game.

A first level hypergame is one in which each player has their own view of the game being played. Each player is unaware that they are playing different games. A more complex game would be a second level hypergame, where one or more players is aware that a hypergame is being played. At this level each player has their own hypergame, which models the game that they believe that they are playing and the game that they believe that their opponents are playing. A third level hypergame would have each player having their own second level hypergame, which in turn consists of a series of hypergames showing their perceptions of the games that they themselves and their opponents are playing. According to Fraser and Hipel (1984) it is unnecessary to model conflicts any higher than this level, although it is possible. In general, a hypergame will consist of  $p^n$  games, where  $p$  is the amount of players and  $n$  is the level of the hypergame.

In Figure 1.1 a third level hypergame between two players is shown. Each player has their own perception of the situation (Hypergames A and B). Each of these hypergames is made up of two smaller first level hypergames. These hypergames show the games that each player is playing and the game their opponent is playing against them. For instance B’s perception of A’s game is modelled in the hypergame AB, whereas A’s perception of its hypergame is hypergame AA.

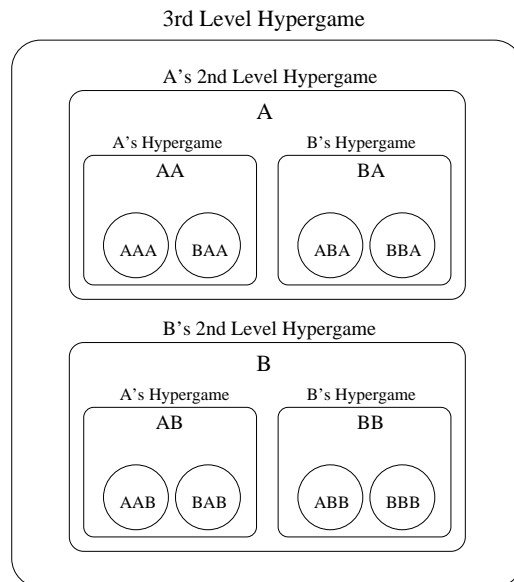


Figure 1.1: A third level hypergame between players A and B

Hypergames are analysed by using Fraser and Hipel's (1984) game analysis algorithm. First a new game is constructed to represent the relative aspect of the hypergame that is to be analysed. This is achieved by taking a preference vector for each player from a game that represents their perception of the situation and combining these together to construct the new game. The algorithm is applied to this new game. For example, consider the hypergame in Figure 1.1. To calculate the equilibriums for the hypergame BB, B's preference vector is taken from the game BBB and A's preference vector from ABB is taken and merged to make a new game to be analysed. Likewise, equilibriums for A's 2nd level hypergame can be obtained by analysing a game created with A's preference vector from AAA and B's preference vector from BAA. To analyse the overall hypergame equilibriums, the preference vectors from each player's perception of their own games are combined, namely A's vector from AAA and B's vector from BBB.

Analysing a hypergame involves analysing each of the games for stability and then comparing the results of the to find stable equilibriums for the hypergame. It is noted by Wang, Hipel and Fraser (1988) that solutions to hypergames may not necessarily be created by outcomes that are stable for all players and that it is possible that an outcome that is unstable individually for players may actually be an equilibrium for the hypergame.

Many hypergame analyses have been published, showing its use in modelling conflicts and their resolutions. Hypergames analysis methods are not just limited to military conflicts, but can be applied to conflicts such as international disputes (Fraser and Hipel, 1980), economic treaties and agreements (Giesen and Bennett, 1979), social issues (Bennett, Dando and Sharp, 1980) and environmental issues (Peter Savich and Fraser, 1983), (Fraser and Hipel, 1979).

### 1.5.1 Strategic Surprise in Hypergames

The Fall of France in 1940 has been previously analysed by Bennett and Dando (1979) and again by Fraser and Hipel (1984) and is a good example of using a hypergame to model strategic surprise by one player having options that other players do not perceive. In effect

the players are playing different games, that have differing options and outcomes. In the case of the Fall of France, the hidden option is the German's option to attack France through the Ardennes, a threat which France had disregarded, believing it was infeasible.

France believed that the Germans could either attack the Maginot line in the south or attack through Belgium in the north and were prepared for either of these actions. If the Germans went north, the French would intercept them and if they attacked the Maginot line to the south, then France would move the troops from the north down to hold the defenses. Bennett and Dando (1979) say that from a game theoretic viewpoint, the French defensive plan could not be faulted, since it covered all the possibilities from their perception of Germany's game.

In the German game, the Germans had their extra option, to attack through the Ardennes. This allowed them to launch an attack that France did not expect and had no strategy to deal with the attack. This strategy allowed Germany to defeat the Allies in France in less than two weeks. This situation was modelled as a second level hypergame, since the players are playing different games, but only the Germans are aware of this. In this case, the Allied defeat was due to their misperception of the situation, believing that they had an effective defense for any strategy that Germans might use, when they in fact did not.

Other published cases that use hypergames to model strategic surprise include the Suez Canal Crisis (Fraser and Hipel, 1984). This hypergame analysis studies the nationalisation of the Suez Canal by Egypt in 1956 and the later invasion of the canal by English, French and Israeli troops later that year.

### 1.5.2 Deception in Hypergames

Another military hypergame that has been analysed is that of Operation Overlord, the D-Day landings, by Fraser and Hipel (1984) and Takahashi, Fraser and Hipel (1984). In this analysis, D-Day is modelled as a third level hypergame, where the Allies have two options, invade at Normandy or invade at Calais. The Germans also have two options, defend at Normandy or defend at Calais. Each player's options are also mutually exclusive, the Allies can only invade one location and the Germans can only defend one location.

The Allies and the Germans had an accurate perception of each other's options but not of each other's preferences. The game was modelled as a third level hypergame in which the Allies had caused Germany to misperceive the games being played.

In the Allies' perception, the Germans have been persuaded that the Normandy invasion is a feint and that Calais is the real target. Based on this assumption, the Allies will then attack at Normandy.

In Germany's 2nd level hypergame, Germany believes that Calais is the invasion target and Normandy is a feint. Germany's perceived model of the Allied game has the Allies invading at Calais, believing that the Germans are defending Normandy. Germany therefore perceives that it has misled the Allies into thinking the feint at Normandy has worked.

The Allies used several strategies to convince the Germans that Calais was the invasion target. These strategies included letting the Germans know that they were buying maps of the Calais region and broadcasting to German pilots, directing them to Calais. In an information warfare context, these actions would be defined as corruption of Germany's information by the Allies, allowing them to manipulate the German model of the situation (Kopp, 2002).

For this conflict, the equilibrium state for the Allies' second level hypergame is that where they invade at Normandy and the Germans defend at Calais and the equilibrium state for the Germans second level hypergame is where the Allies attack at Calais and the Germans defend at Calais. From both of these equilibriums, it can be seen due to the German's misperception of the invasion target to be Calais, the equilibrium for the conflict is where the Germans defend Calais and the Allies attack Normandy, which was the historical outcome.

## 1.6 Computerised Tools for Game Analysis

The usefulness of computers as a tool for performing game analyses has long been known. Computers are an ideal tool for aiding hypergame analyses, as they can analyse a model a large number of players and/or options much more reliably and faster than a person using a pencil and paper. Some of these programs include CAP (Fraser and Hipel, 1984), CONAN (Howard, 1987), DecisionMaker (Fraser and Hipel, 1988) and INTERACT (Bennett et al., 1993).

CAP (Conflict Analysis Program) was an early analysis program, written in BASIC for the CompuColor-II and then the IBM PC. It was limited to modelling only single games that contained up to six players and at most ten options. These size restrictions are due to hardware and software limitations at the time, not the analysis algorithm itself.

CONAN (CONflict ANalysis) is another analysis program, which is capable of operating in an iterative manner, allowing the user to enter game details, analyse a game and then alter the game after receiving feedback about the game results. Users are able to compare game results with those from previous iterations. It was eventually envisaged that CONAN would be extended to draw strategy maps of the game being analysed and to simulate the behaviour of the game's players.

DecisionMaker is an updated version of Fraser and Hipel's (1984) CAP. DecisionMaker allows for up to 64 players and a total of 64 options. It also makes use of improved data structures, using binary trees (Fraser, 1993) to store player preferences. Other useful features are the ability to easily create and order preferences and also the easy removal of infeasible outcomes.

INTERACT is game analysis tool that was developed to be easy to use, avoiding the use of game theoretic terms in order to present information in an easily understood method to users. The user enters initial information about the conflict by creating a diagram representing the players and issues and the relationship between them. After the analysis, stability information is presented to the user in the form of a strategy map.

Of all the existing systems for creating and analysing games, there are several common features. Each tool also has some unique features to set it apart from the others. However none of these tools are currently freely available for modern computer platforms.

## Chapter 2

# Research Methods

### 2.1 Specifications

The first aspect of the design to be considered was the structure of the program, considering whether it should be a single monolithic program or in some sort of Client/Server structure. In the monolithic structure a single program encapsulates all of the required features whereas a Client/Server architecture would separate the user input and output from the hypergame analysis. The Client/Server approach was chosen due to several advantages that it had over the monolithic program. Some of these advantages include forcing the modularity of the system by separating different aspects of the system, allowing for easier extension to the program in the future and allowing for differing implementations of the client and/or server to be interchanged in the future.

After deciding on a Client/Server structure, the method of communication between the components needs to be decided upon. Some possible communications methods available included shared memory or file or network sockets. A shared memory/file implementation would have the advantage of high speed communication between components, but is limited in that both processes need to run on the same machine. A sockets implementation of the program would have a slower communication but could run on separate machines, on different platforms in different locations. A sockets communications system was eventually chosen as the communications method as it allowed for different platforms to be used in the future for the client and server. Shared memory was deemed to be unsuitable as it would force the two processes to run on the same machine and its speed advantage is not very important in this application. TCP/IP was chosen as the sockets protocol since it is supported by many platforms, following on the theme of a multiplatform system.

The server will be a program that while operating simply waits for connections from client machines, accepts their game input, processes it and returns the results. It could run on any of the main platforms available today, such as Windows, Linux or another Unix variant. For this project, a standard x86 PC running Linux would be a suitable server platform as machines such as this are widely available at the University. Linux itself also has good support for programming, with suitable libraries for networking and sockets easily available. Documentation is also easily accessible for Linux. Another advantage of using Linux is that the server can be ported easily to other Unix platforms, should the need ever arise.

The client program could be run on many possible platforms, such as a desktop PC running Windows, Linux or a Unix variant or even a mobile computing device, such as a PDA, mobile phone or tablet PC. To keep things simple and avoid cross platform incompatibilities, it

was decided to use Linux for the client component as well. This will also increase the ability to reuse common code between the the server and client.

The interface for the client could be either a text based system or a GUI system. For a text based system, `curses` would be a suitable library to use and for a GUI, one of the available X Windows libraries, such as Motif or GTK+ could be used. A text based interface would be faster to develop than an X Windows. It would also be easier to port between platforms. It would also have the advantage of being easily accessible by a remote connection from another machine, without the extra problems and overhead that X Windows would have. It is also possible to replace the client program with another client that has a GUI.

## 2.2 Implementation

### 2.2.1 Hypergame Modelling Language (HML)

A major concern in the design of the system was how would data be transmitted between the proposed client and server components.

As the initial design of the system began, it became clear that an organised format for a hypergame was required to allow saving and loading of hypergames and the transfer of hypergames between components of the system. Some approaches to this problems that were considered included XML, a memory dump/read of the game structure or a custom language for describing a hypergame.

XML would provide transformation style sheets, be human readable and has many available parsers and related tools, but these could add extra overheads/difficulties, while the XML files could be quite large, although they could be compressed well. Another reason to use XML is the possibility of using XML transformation style sheets, to transform the XML data representing a game into HTML markup. XML has the advantages of being human readable and having many available libraries for manipulation but it is also complex and may lead to an overly large file format, although the data files could be compressed. Another point to consider is that an XML specification of a hypergame format could also be re-used in many other programs that would use hypergames. In the end XML was found to be unsuitable due to the larger file size and the need to become familiar with its use and related libraries.

A memory dump of the structure would be relatively compact, though almost impossible for a human to read or debug. The advantage of having a small file size would only be worthwhile if a text file description of a game was quite large. However in the case of a hypergame, the descriptions could be kept relatively compact, so there was little reason to use this format.

A custom language could be constructed that is both human readable and more compact than XML. It would also be useful as there are currently no languages for modelling a hypergame. A parser for this language could be constructed using tools such as *lex* and *yacc* or their open source alternatives *flex* and *bison*. A custom language was decided upon for these reasons:

- Easy to develop with existing tools
- Hadn't been attempted before
- Human readable form, allows for game creation with a text editor.

- Parser can easily convert from grammar to structures in memory.
- No need to become familiar with another language (XML).

The decision to develop a custom grammar for a hypergame was a major design step and required reducing the scope of the proposed software product from a Client/Server program to a stand alone program that implemented a hypergame grammar, that was able to read and write into this format. To further simplify the coding from this point, the requirement for a `curses` interface was dropped back to a simpler, plain text interface.

## Grammar Structure

The overall structure for a hypergame model can be represented as below.

A Model = 1 or more games.

A Game = 2 or more players and some record of who the game belongs to.

A Player = name, options, mutually exclusive options and a preference vector.

An Option = option identifier and a text description.

Mutually Exclusive Options = options that cannot both be taken at the same time.

A Preference Vector = preferred ordering of the feasible outcomes.

A model can be represented as either a list of games or a tree of games that make up a hypergame. When using the list method, each game's position in the hypergame needs to be explicitly stated inside the game, whereas by using a tree structure the overall position of a game is implicitly stored by the games location in the tree. It was decided to explicitly store the position of the game inside the overall hypergame as this should increase the readability of the file, allowing a reader immediately see where in the hypergame the game belongs, without needing to examine the rest of the file.

It was decided to use curly braces (`{` and `}`) to separate blocks of the grammar, with lists of blocks separated by commas. This format also gives the idea of blocks of text, similar to those used in programming languages such as C. Comments are also a feature of the language, allowing the various components of a game to be labeled in order to improve readability.

The grammar was written in Backus Naur Form (BNF) with non terminals enclosed in angle brackets (`<` and `>`). The terminal characters are curly braces (`{` and `}`), the comma (`,`), `playerID`, `playerName`, `optionID`, `optionDescription` and `val`. The `playerID` is a unique numeric identifier for the player, the `playerName` is a string representing the player's name, the `optionID` is a unique numeric identifier for identifying options, the `optionDescription` is a string describing what an option is and `val` is either 0 or 1. To prevent confusion between `optionID` and `val` when parsing, all `optionIDs` are prefixed with a `#` character. The preference vector is modelled as an ordered number of outcomes each of which is made up of a number of opt pairs. The opt pairs consist of an option ID and a value which indicates whether an option is to be taken or not taken for the outcome it is part of. The initial grammar is in Appendix A.1.

While implementing the grammar, various shortcomings and oversights became apparent. One of these was the method of storing the various perceptions of each of the sub games that make up a hypergame. If the perceiver is just a single identifier for a player then only first level or lower hypergames can be modelled correctly with this grammar. The perceiver was first modified to only show whose game was being perceived and who was perceiving

the game. This only stored two different player references and limited hypergame modelling to games with two or less levels. The perceiver was again modified, this time to list the full perceivers of a game, which shows exactly where a certain game fits into a larger hypergame. This method is the same as that of Fraser and Hipel (1984), which labels each game as of a hypergame by whose game it belongs to at each level. For instance the perception  $\{A,B,B\}$  states that the current game is B's perception of B's perception of the game being played by A.

Another extension was to improve support for strategic surprise. Previously whenever a player had no knowledge of a particular option, that option was absent from their outcomes. This led to difficulties with analysing games of strategic surprise, where outcomes that didn't contain all the same options were compared, as matches would not be found and this would usually an incorrect analysis result at best. Support was improved by allowing a value for an option to be '?' as well as 0 or 1. This is to indicate that the player has no knowledge of the option.

Another small change was to remove the IDs used to refer to players. This was done because the IDs were rarely needed and in the few instances where it was necessary to refer to a player, this could be done with the player's name instead.

The updated grammar has been implemented for HYPANT and the grammar for it is in Appendix A.2. Examples of the Hypergame Modelling Language can be found in Appendix B.1.

## Comments

The comment characters are two percent characters (`%%`), as the percent character is one that would not be needed in the text descriptions included in the HML file. These comments behave the same as those in C++ (`//`), in that they ignore all text from the comment characters to the end of the line. Comments would never need to be dealt with by the parser, as the lexer would strip them away before parsing. The comments also behave the same with all characters between the `%%` and the newline character ignored.

## Sanity Checking

It is also important to test that the values provided for a game are suitable for use, that is that they make sense in the context they are used in. For example while the parser will accept any text string as a suitable player name, when labeling a perception for a game the strings that are used here should only match players in the game. This is also the case in programming languages - while an assignment operation of a `long float` to a `char` may be a valid use of the syntax, it will not be accepted by a compiler as a valid operation.

It is therefore essential for any hypergame that is accepted by the parser to be sanity checked to ensure that it is suitable for analysis.

However sanity checking has not been implemented in the current version of HYPANT, due to time restraints. Extensive testing would also be required to ensure that the sanity checking catches only invalid uses of the language.

Some examples of cases that the sanity checking should catch.

- The various games that make up a hypergame should have the same players.
- The perceptions for all the games should have the same number of player names.

- The size of the perceptions should be consistent with the amount of players and games.
- Option ID numbers should be unique.
- Option IDs used in the mutual exclusives or preference vector exist.

### 2.2.2 Client Output

To make the output generated by HYPANT much clearer, colour has been used to highlight important results of analyses.

The terminal supports 16 different colours, which are listed in Table C.1. These colours can also be seen in use in the screenshots in Appendix D.

Properties can also be used with the text, such as blink, underline or reverse. Of the available colours, not all of them can be used. For example, the standard terminal usually has a black background with white text. This removes these two colours from those available. Next Bright White and Dark Grey were removed due to their similarity to Black and White. These four colours were set aside for use by the interface only, that is only general output would use these colours. Dark Red was also reserved for error messages. This leaves 11 colours that could be used for highlighting various elements of hypergames. Each of these colours is then allocated to a player, based on a hard coded allocation scheme. These colours were allocated as shown in Table C.2. The colours were allocated in such a way that the first two to six colours would be clearly distinct from each other. It would be a simple change to the program to allow the colour allocation to be set by the player as a configuration option. This could allow the colour allocation to be defined in an external file, in which the user can specify their desired colour scheme. As the colours are unique to each player, a player's colour is used to highlight which elements of a game are their own, such as their options and the perception of games in a hypergame.

Headings may also use these colours, except they are always be underlined, to differentiate them from elements that belong to a player. These colours are used to differentiate between the various player's options that make up a game, as in Appendix D.1 and Appendix D.2. This screenshot shows that the USA has their name and options coloured Light Red, while the USSR has their name and options coloured Bright Blue. The information for a player is made to stand out by using the player's colour whenever information relating to them is presented. This helps to make the tables clearer to understand and easier to see which information is linked to which player.

The interface reserved colours, such as Bright White and Dark Grey, were only used for making parts of the interface stand out to the user. For example Bright White was used for important headings, Dark Red was used for error messages and Dark Grey was used for indicating unselectable menu items.

### 2.2.3 Algorithm Implementation

When implementing the algorithm, it was necessary to place a limit on the depth of recursion for the calculation of Unilateral Improvements and deterrents to these improvements. For instance a player may be able to change their strategy from one outcome to another. This change is only a Unilateral Improvement if their opponent cannot then change their strategy from the new outcome to another outcome that is less preferred by the first player, thus deterring the improvement. When checking for these deterrents, it is possible to end up in a loop, with the players making improvements through a sequence of outcomes over and

over again that will never terminate. To avoid this problem, a limit to the depth of testing can be implemented. In HYPANT, this depth was hardcoded to be 10. This means that if one player's change in strategy cannot be deterred within 10 changes, then it is considered to be undeterred and the player may make that improvement. However this depth may not be deep enough for larger game models with many choices.

## 2.3 Testing

To test that the program operates correctly on input games, it would have to pass several tests, such as being:

1. Able to read in games into the correct structures in memory.
2. Able to calculate the correct Unilateral Improvements (UIs) for each player.
3. Able to calculate the correct stabilities for each player, based on the UIs.
4. Able to calculate the correct equilibriums for the overall game.

The first test covers the correct operation of the parser, while tests 2, 3 & 4 make certain that the various components of the analysis component operate correctly. HYPANT needs to be able to pass all of these tests for several different games before it is possible to say that it correctly analyses hypergames.

### 2.3.1 Parser

In order to test the parser, previously analysed conflicts were converted from their existing form to a Hypergame Modelling Language (HML) text file and then parsed. Basic testing of the parser checked that it would accept the input file. More intensive testing checked whether the information in the HML files was read in correctly, whether small parse errors would escape notice and whether games of varying configurations would be accepted by the parser.

Initial testing was performed on the Cuban Missile Crisis Game. The parser operation was tested by converting the game to HML and entering it into a text file, which was supplied to the game as input. After several parse errors had been corrected, the game could be read into memory and then the memory structure of the game could be successfully displayed. Since the Cuban Missile Crisis Game is only an individual game, it was necessary to next test whether hypergames could be correctly parsed. Firstly the previous game was copied several times into the same file to make a hypergame where each game was a copy of the previous game. HYPANT was able to parse this game, but when it was displayed it was impossible to tell whether each game structure corresponded to its related game in the file or whether they were all copies of the first game. This required an existing hypergame model to be entered.

For this case, the Fall of France hypergame (see Section 1.5.1) was used, since it was a relatively small and simple model made up of four games. This also helped to expose bugs in the implementation of the grammar. The display of the full game can be seen in the screenshots in Appendix D.4 and Appendix D.5. From these tests, it was determined that the parser was operating correctly on valid games and correctly placing the various elements of the games into the memory structures for the hypergame.

### 2.3.2 Analysis Algorithm

The implementation of Fraser and Hipel's conflict analysis algorithm was tested first by running it on the Cuban Missile Crisis game. This was first done with extensive checks on the intermediate output of the various functions that perform the analysis. This helped to locate any errors in the intermediate calculations of the various functions, which would lead to incorrect results. After the analysis functions were operating correctly, the stabilities of all the outcomes and the unilateral improvements for each player were compared with those previously calculated. After it was demonstrated that the stability table entries could be calculated correctly, the calculated equilibriums were compared to the equilibriums previously calculated for that conflict. When it came to testing the analysis of hypergames, the previous steps were repeated for each game that made up the hypergame. After the individual games were checked for correctness, the equilibriums that were calculated for each of the hypergames in the model was checked against the existing equilibriums for the hypergames.

## Chapter 3

# Results

HYPANT was tested on several types of games, which included individual games, hypergames and hypergames with strategic surprise. First HYPANT was tested by analysing games that had previously been analysed, so that the results calculated could be compared with those that were already known. If the results obtained matched those of the previously analysed games, then it is safe to assume that the program is capable of at least being able to analyse these games correctly.

After these tests were completed, HYPANT was tested on other games. These games were created by taking existing models that had been analysed correctly and altering them to model a “What If” scenario. Two of these games were analysed, both of them based on the Cuban Missile Crisis. This game was chosen as it is a relatively simple game, that is also complex enough to be interesting when analysed. Its size makes it easy to study the effects of changes on the whole system. One modification to the game involves altering the USSR’s preference vector so that they are more aggressive, preferring to escalate the situation if action is taken against them. The other has both players more determined to “win” the conflict, which is represented in the model by making the USA more aggressive and the USSR more unwilling to back down.

The games that were analysed were

- *Cuban Missile Crisis*  
(Previously analysed by Fraser and Hipel (1984)).
- *Operation Overlord / D-Day*  
(Previously analysed by Fraser and Hipel (1984) & Takahashi et al. (1984)).
- *Fall of France*  
(Previously analysed by Bennett and Dando (1979) & Fraser and Hipel (1984)).
- *Garrison Diversion Unit*  
(Previously analysed by Fraser and Hipel (1984)).
- *Altered Cuban Missile Crisis 1*  
(Modelled with USSR more aggressive, no change to the USA).
- *Altered Cuban Missile Crisis 2*  
(Modelled with USA more aggressive, USSR more stubborn).

### 3.1 Cuban Missile Crisis

The Cuban Missile Crisis is analysed as a single game of perfect knowledge, with both the USSR and the USA having no misperceptions about the situation. In this conflict the USSR installed nuclear missiles in Cuba, less than 100 miles from the USA. The USA has the options of destroying the missiles with airstrikes or launching a naval blockade of Cuba. The USSR has the options of withdrawing the missiles or escalating the conflict. The historical resolution to this was of the USA launching a blockade of Cuba, while the USSR withdrew the missiles. Fraser and Hipel's model when analysed has two equilibriums: the historical outcome and the USA doing nothing, while the USSR withdraws the missiles.

Fraser and Hipel (1984) have modelled this conflict as a single game, a first level hypergame and a second level hypergame. In these tests, the game was only studied as an individual game. First the calculated stability table was contrasted with the previously calculated tables (Table 3.1 and Table 3.2). Without a correct stability table, any analysis will almost certainly be erroneous.

Previous		HYPANT	
Stability	UIs	Stability	UIs
r	-	r	-
s	1	s	1
u	1 2	u	1 2
u	1 2 3	u	1 2 3
r	-	r	-
u	5	u	5
u	5 6	u	5 6
u	5 6 7	u	5 6 7
r	-	r	-
u	9	u	9
u	9 10	u	9 10
u	9 10 11	u	9 10 11

Table 3.1: Stability Tables for Cuban Missile Crisis (USA)

Previous		HYPANT	
Stability	UIs	Stability	UIs
r	-	r	-
s	1	s	1
r	-	r	-
u	3	u	3
r	-	r	-
u	5	u	5
r	-	r	-
u	7	u	7
u	7 8	u	7 8
u	5 6	u	5 6
u	3 4	u	3 4
u	1 2	u	1 2

Table 3.2: Stability Tables for Cuban Missile Crisis (USSR)

Screenshots of HYPANT’s calculated stability tables can also be seen in Appendix D.2. With both the stability tables being calculated correctly, the overall analysis results can be compared. The original analysis provides two equilibriums, one where the USA takes no action and the USSR withdraws the missiles. The other occurs when the USA launches a blockade of Cuba and the USSR withdraws the missiles. HYPANT’s calculated equilibriums are where the USA takes no action and the USSR withdraws and where the USA blockades while the USSR withdraws, which matches the previously calculated equilibriums. The calculated results can be seen in the Appendix D.1.

From these results we can determine that HYPANT is capable of analysing smaller, individual games.

### 3.2 Operation Overlord

Operation Overlord is a 3rd level hypergame showing differing levels of misperceptions in the lead up to the D-Day landings during World War II. The background and structure of this hypergame was described earlier in Section 1.5.2.

Fraser and Hipel’s (1984) model was first converted from the various preference vectors and tables into the HML format and used as input for HYPANT. The converted file can be found in Appendix B.2. HYPANT was then used to analyse the individual games that make up the hypergame and the results were compared to those of Fraser and Hipel in Table 3.3. Since these are the smaller games of a hypergame, their analysis results aren’t needed for the hypergame analysis, but they may be related to any results obtained from the hypergame analysis.

Game	Fraser and Hipel’s Equilibriums		HYPANT Equilibriums	
	<i>Allies</i>	<i>Germany</i>	<i>Allies</i>	<i>Germany</i>
Allies, Allies, Allies	IN	DC	IN	DC
Germany, Allies, Allies	IC	DC	IC	DC
Allies, Germany, Allies	IC	DN	IC	DN
Germany, Germany, Allies	IC	DC	IC	DC
Allies, Allies, Germany	IC	DN	IC	DN
Germany, Allies, Germany	IN	DN	IN	DN
Allies, Germany, Germany	IC	DN	IC	DN
Germany, Germany, Germany	IC	DC	IC	DC

IC = Invade Calais DC = Defend Calais IN = Invade Normandy DN = Defend Normandy

Table 3.3: Equilibriums for Individual Games in Operation Overlord

After the individual games had been analysed and found to match the previously calculated results, the overall hypergame and sub hypergames were analysed. The results were then again compared with those previously obtained in Table 3.4.

The calculated results also match the existing results. This leads to the conclusion that HYPANT is at least capable of calculating the equilibriums for small hypergames up to the third level.

Hypergame	Fraser and Hipel's Equilibriums		HYPANT Equilibriums	
	<i>Allies</i>	<i>Germany</i>	<i>Allies</i>	<i>Germany</i>
Allies, Allies	IN	DC	IN	DC
Germany, Allies	IC	DC	IC	DC
Allies, Germany	IC	DN	IC	DN
Germany, Germany	IC	DC	IC	DC
Allies	IN	DC	IN	DC
Germany	IC	DC	IC	DC
Overall	IN	DC	IN	DC

IC = Invade Calais DC = Defend Calais IN = Invade Normandy DN = Defend Normandy

Table 3.4: Equilibriums for Hypergames in Operation Overlord

### 3.3 Fall of France

The Fall of France is a 2nd Level hypergame that demonstrates the applications of strategic surprise. The background for this game is presented in detail in Section 1.5.1. As before first the individual games of the hypergame are analysed and the results compared to those previously calculated (Table 3.5).

Game	Fraser and Hipel's Equilibriums		HYPANT Equilibriums	
	<i>France</i>	<i>Germany</i>	<i>France</i>	<i>Germany</i>
France, France	MN	AN	MN	AN
Germany, France	MN	AN	MN	AN
France, Germany	MN	AN	MN	AN
Germany, Germany	MN	AA	MN	AA
	DML	AA	DML	AA

MN = Move North, DML = Defend Maginot Line, AN = Attack North, AA = Attack Ardennes

Table 3.5: Equilibriums for Individual Games in Fall of France

Once again, HYPANT was capable of correctly analysing these games, as they are only small individual games. Since these games could be analysed, the next step was to analyse the hypergames that make up the hypergame and these results are contrasted with the previous calculations in Table 3.6. These results can also be seen in Appendix D.6.

Hypergame	Fraser and Hipel's Equilibriums		HYPANT Equilibriums	
	<i>France</i>	<i>Germany</i>	<i>France</i>	<i>Germany</i>
France	MN	AN	MN	AN
Germany	MN	AA	MN	AA
	DML	AA	DML	AA
Overall	MN	AA	MN	AA
	DML	AA	DML	AA

MN = Move North, DML = Defend Maginot Line, AN = Attack North, AA = Attack Ardennes

Table 3.6: Equilibriums for Individual Games in Fall of France

Once again HYPANT been able to correctly calculate the equilibriums for a hypergame, this time one involving strategic surprise.

### 3.4 Garrison Diversion Unit

The Garrison Diversion Unit conflict is a larger individual game, with four players who have nine options between themselves. In this conflict, a large water resources project is to be built in North Dakota in the USA. However this construction may lead to environmental damage of areas inside both the USA and Canada. The four players in this game are the US support for the Garrison, the US opposition to Garrison, the Canadian opposition to Garrison and the International Joint Commission (IJC), a group that was created to consider disputes between the USA and Canada.

The US support has the option of completing a full GDU (#1), a modified GDU to reduce the impact on Canada (#2) and a modified GDU to appease US environmentalists (#3). The US opposition has the option of taking legal action based on existing environmental laws (#4). The Canadian opposition has the option of taking legal action based on a Boundary Treaty signed with the US (#5). The IJC has the option of supporting the completion of a full GDU (#6), supporting completion of a GDU that addresses Canadian concerns (#7), supporting the suspension of the GDU except for the Lonetree Reservoir (#8) and the complete suspension of the GDU (#9). This conflict has  $2^9$  (512) possible outcomes, which is reduced to 23 feasible outcomes.

The results of this analysis can be seen in Appendix D.7 and Appendix D.8.

For this game, the results obtained do not match those previously calculated by Fraser and Hipel. Three of the four players in the game have incorrect stability tables and the overall equilibriums calculated are also incorrect due to this.

For the US Support, outcome 4 in their preference vector should have no Unilateral Improvement (UI) and be rational instead of stable, outcome 7 should also have no UI and be rational instead of stable. The US Opposition has the correct stability and preference vector. For the Canadian Opposition outcome 10 is correctly unstable but its UI should be to outcome 7 instead of outcome 1, outcomes 16 and 17 have the correct UIs but are unstable instead of stable. For the IJC the first nine outcomes are correctly identified as rational with no UIs, while the remaining 14 are not. These outcomes all have one, two or three UIs and are all said to be unstable when they should be rational with no UIs.

Overall equilibriums were calculated to be the outcome where the US Support has full support, the US Opposition takes legal action, the Canadian Opposition takes no action and the IJC provides full support and the outcome where the US Support attempts to appease the environmentalists, the US Opposition takes no action, the Canadian Opposition takes no action and the IJC provides full support. There are seven previous equilibriums for this game. HYPANT identified two of them, however the remaining five were not calculated to be equilibriums as it had wrongly calculated that each of them was not rational for at least one player.

The problem has been tracked down to the function that calculates each player's unilateral improvements. The function sometimes incorrectly calculates that the player is able to improve, when they cannot. This leads to an incorrect stability table for at least two of the four players, the US Support and the Canadian Opposition. The stability tables for the US Opposition is however correct. The IJC also has an incorrect stability table, but this may be due to either the function calculating incorrect unilateral improvements for the player, the results calculated being based on earlier incorrect results for other players or a combination of these two problems.

A possible cause of the problem is that the UI calculating function may not be able to operate correctly on the larger outcome strings that are present in this game. This leads to the players having extra UIs added, which then causes the stabilities for each players to be incorrect, as they are calculated based on different information.

In this case, it can be seen that while the overall analysis results did not match those previously calculated due to errors in the intermediate calculation of each player's stability table and UIs. However the overall equilibriums calculated seem to be sensible answers, given that these outcomes are the only ones that are stable or rational for all the players.

### 3.5 Altered Cuban Missile Crisis 1

In this altered game the preference vector of the USSR was modified to be more aggressive. The preference vector of the USSR was modified to make them want their missiles to remain in place and if this could not happen, then they would prefer outcomes where they escalate against any aggressive action against them over outcomes where the missiles are removed. The altered USSR preference vector can be seen in Table 3.7.

Preference	Normal	Altered
1	No action, No action	No action, No action
2	No action, Withdraw	Blockade, No action
3	Blockade, Withdraw	Airstrike, No action
4	Blockade, No action	Airstrike & Blockade, Escalate
5	Airstrike, Withdraw	Blockade, Escalate
6	Airstrike, No action	Airstrike, Escalate
7	Airstrike & Blockade, Withdraw	Airstrike & Blockade, Withdraw
8	Airstrike & Blockade, No action	Airstrike, Withdraw
9	Airstrike & Blockade, Escalate	Blockade, Withdraw
10	Airstrike, Escalate	Airstrike & Blockade, No action
11	Blockade, Escalate	No Action, Withdraw
12	No Action, Escalate	No Action, Escalate

Table 3.7: USSR Altered & Normal Cuban Missile Crisis Preference Vectors

The stability table calculated for this conflict can be seen in D.9 and the calculated equilibriums in D.10. HYPANT calculates that there are three equilibriums for the altered game. One involves the USA launching an airstrike and the USSR withdrawing, another the USA blockades and the USSR takes no action, while for the last the USA blockades and airstrikes while the USSR escalates. Of these three outcomes, the outcome where the USA blockades and the USSR takes no action seems to be the most moderate and represents both sides making conservative, safe actions, while at the same time trying to get the best payoff that they can. This seems as though it could be a reasonable and likely solution, given the preferences of both players.

When the USA airstrikes and the USSR withdraws, the USA is playing aggressively while the USSR is playing meekly. This seems to be a bit out of character for the more aggressive USSR, but this outcome is stable for both players and therefore can be considered an equilibrium. It is important to note that while this outcome is an equilibrium, both players have more preferred outcomes that they are deterred from changing strategies to.

In the outcome where the USA both blockades and airstrikes and the USSR escalates, both players can be seen to be playing aggressively. From the USA's point of view, if the USSR is going to escalate, then the USA feels that they must make it worth the effort by airstriking and blockading beforehand. From the point of view of an aggressive USSR, if the USA is going to airstrike and blockade them, then escalating seems to be an acceptable response. This outcome only becomes an equilibrium in this game because of its higher ranking in the USSR's new preference vector.

All three equilibriums proposed by HYPANT for this altered Cuban Missile Crisis seem plausible. A wide range of outcomes from fairly moderate to aggressive can be seen, depending on the behaviour of each player.

### 3.6 Altered Cuban Missile Crisis 2

The overall aim of the modifications to this game was to model a situation where the USA is more aggressive and prefers to take action against the the USSR, while the USSR is more stubborn and would prefer the missiles to stay where they are, even if action is taken against them. From the USA's point of view they are flexing their muscles, demonstrating that they won't be pushed around by their opposing superpower. The USSR is also takes a fairly hard line that the missiles will stay, no matter what the USA might do. The aim was to try and see if with these preferences, the USSR could be pushed to escalate the situation, which in the case of this game would lead to a nuclear exchange.

Preference	USA	USSR
1	Airstrike & Blockade, Withdraw	No action, No action
2	Airstrike, Withdraw	Blockade, No action
3	Blockade, Withdraw	Airstrike, No action
4	No action, Withdraw	Airstrike & Blockade, No action
5	Blockade, No action	Airstrike & Blockade, Withdraw
6	Airstrike, No action	Blockade, Withdraw
7	Airstrike & Blockade, No Action	Airstrike, Withdraw
8	No action, No action	No action, Withdraw
9	Airstrike & Blockade, Escalate	Airstrike & Blockade, Escalate
10	Airstrike, Escalate	Airstrike, Escalate
11	Blockade, Escalate	Blockade, Escalate
12	No Action, Escalate	No Action, Escalate

Table 3.8: USA & USSR Altered Cuban Missile Crisis 2 Preference Vectors

From the stability table for this game (Appendix D.11) it can be seen that for the USA the outcomes where the USSR withdraws the missiles are either rational or stable, while outcomes where they blockade while the USSR takes no action and where they airstrike and blockade while the USSR are also rational.

The overall calculated equilibrium (Appendix D.12) occurs when the USA blockades and the USSR takes no action. This is due to the fact it is the only outcome that both players believe is suitable (stable for the USA and rational for the USSR).

## Chapter 4

# Conclusions

### 4.1 Observations

As the analysis results calculated by HYPANT usually match those obtained by previous researchers, it can be inferred that HYPANT is correctly interpreting the game and also correctly analysing it. However it appears that there is at least one flaw in an intermediate step of the analysis and it will be necessary to analyse many more games to identify the exact cause of this flaw. At the very least, it is safe to assume that the program can analyse these smaller, simple games. However it may be unable to operate correctly on larger games with many players and options but this should not be the case as the algorithm used can scale up to analyse arbitrarily large hypergames.

HYPANT allows for the analysis of both individual games and hypergames. It has been tested on a wide variety of games that have differing numbers of players and options, strategic surprise and different depths.

Current drawbacks of HYPANT are the inability to create hypergame models with it. This forces the user to have already constructed a model using a text editor. It also requires the user to understand the syntax of the Hypergame Modelling Language and be able to express their model correctly in this form. In an ideal solution, they would be able to create their model using some graphical interface that requires them to have little knowledge about the underlying game theory, analysis methodology or language used to describe the game.

In hindsight, an object oriented language, such as C++ may have been a better choice for the language used. The datastructures used to store the hypergame models would have become a hypergame class, that could be reused in future projects. Also sections of code that refer extensively to the game object, such as the analysis algorithm and the output sections may have been easier to code with objects and method functions, as opposed to structs and functions. The existence of a hypergame class with appropriate methods would allow others to reuse this class in their own programs and also extend it as necessary.

### 4.2 Future Research

There are many possible changes that could be made to either extend HYPANT or to re-use the existing analysis components in new programs.

The grammar currently uses right recursion in many of the rules that generate a list of items. Right recursion is inefficient as it can require larger amounts of memory for a stack

storing the current state. Where ever possible, the grammar rules should be modified to be left recursive. An upside to this change is that many of the list construction functions for the parser will need to be replaced with ones operate by placing a new item at the end of the list, instead of the front. For example consider a function which takes a new element and places it at the head of the list. Whenever adding a new item to the list, a large array must be allocated, the new item placed at the front and each remaining item of the list is then placed after the new head and finally the old array deleted. However when using a left recursive grammar, a new array with room for one more element can be created and the new element can be quickly appended to the end, without moving the rest of the list.

HYPANT still lacks sanity checking to check whether the values for the various fields in a hypergame file make sense. This involves extensively checking the game model after it has been parsed. However these checks need to reject all erroneous input, while also accepting valid game models. Another useful feature to add to HYPANT would be improved handling of errors and feedback regarding errors. One such case could be providing line numbers and tokens where parse errors or syntax errors occur. Another case would be adding error recovery to the parser, so it can try to continue or at least clean up any memory previously allocated during the parse attempt. Currently HYPANT doesn't free memory allocated during the parse, if the parser fails due to incorrect grammar usage. This memory is leaked by the program and cannot be released. The main missing functionality from HYPANT is the ability to create hypergames and to edit games that have been created previously. This would probably also require the addition of a graphical, windowing interface to the program. Users need to be able to create games without having to enter a HML file into a text editor. The single part program could also be separated into the originally proposed Client/Server architecture. The separation of the client and server would also lead to the need for a method of transferring hypergame analysis results. For this reason there may be a need for another grammar to describe the analysis results for a hypergame.

One possible item for future research would be hypergame playing agents, that interact in an environment by playing hypergames against each other. This would involve an agent that takes some input from its surroundings, turns this into a hypergame, analyses it and takes actions based on the results of its analysis. This would allow the effects of misperceptions on a species to be modelled, for example what effect does excessive misperception of an agent have on that agent's life expectancy.

Another possible future research project could be to create a method for the 3D representation of hypergames and hypergame results, allowing games to be created and manipulated in 3D, with game results displayed in this manner as well. The three dimensional input method could possibly map different games or hypergames onto the different faces of a three dimensional object, which could be rotated around its various axes to expose the different perceptions of the game. The use of colour would also be an important tool in expressing the model structure and analysis results. If such an interface was constructed it would need to be capable of clearly presenting information to the user, without overwhelming or confusing them.

## Appendix A

# Hypergame Grammar

### A.1 Initial grammar

$\langle \text{MODEL} \rangle \rightarrow \{ \langle \text{GAMES} \rangle \}$   
 $\langle \text{GAMES} \rangle \rightarrow \langle \text{GAME} \rangle \mid \langle \text{GAME} \rangle, \langle \text{GAMES} \rangle$   
 $\langle \text{GAME} \rangle \rightarrow \{ \langle \text{PERCEIVER} \rangle, \langle \text{PLAYERS} \rangle \}$   
 $\langle \text{PERCEIVER} \rangle \rightarrow \{ \text{playerID} \}$   
 $\langle \text{PLAYERS} \rangle \rightarrow \{ \langle \text{PLAYER} \rangle \langle \text{NEXTPLAYER} \rangle \}$   
 $\langle \text{NEXTPLAYER} \rangle \rightarrow , \langle \text{PLAYER} \rangle \mid \epsilon$   
 $\langle \text{PLAYER} \rangle \rightarrow \{ \text{playerID}, \text{name}, \langle \text{OPTIONS} \rangle, \langle \text{MUTEX} \rangle, \langle \text{PREFVEC} \rangle \}$   
 $\langle \text{OPTIONS} \rangle \rightarrow \{ \langle \text{OPTION} \rangle \langle \text{NEXTOPTION} \rangle \}$   
 $\langle \text{NEXTOPTION} \rangle \rightarrow , \langle \text{OPTION} \rangle \langle \text{NEXTOPTION} \rangle \mid \epsilon$   
 $\langle \text{OPTION} \rangle \rightarrow \{ \text{optionID}, \text{description} \}$   
 $\langle \text{MUTEX} \rangle \rightarrow \{ \langle \text{PAIRS} \rangle \}$   
 $\langle \text{PAIRS} \rangle \rightarrow \{ \text{optionID}, \text{optionID} \} \langle \text{NEXTPAIR} \rangle \mid \{ \}$   
 $\langle \text{NEXTPAIR} \rangle \rightarrow , \langle \text{PAIRS} \rangle \mid \epsilon$   
 $\langle \text{PREFVEC} \rangle \rightarrow \{ \langle \text{OUTCOME} \rangle \langle \text{OUTCOMES} \rangle \}$   
 $\langle \text{OUTCOMES} \rangle \rightarrow , \langle \text{OUTCOME} \rangle \langle \text{OUTCOMES} \rangle \mid \epsilon$   
 $\langle \text{OUTCOME} \rangle \rightarrow \{ \langle \text{OPT} \rangle \langle \text{NEXTOPT} \rangle \}$   
 $\langle \text{OPT} \rangle \rightarrow \text{optionID} = \text{VAL}$   
 $\langle \text{NEXTOPT} \rangle \rightarrow , \langle \text{OPTS} \rangle \mid \epsilon$   
 $\langle \text{VAL} \rangle \rightarrow 1 \mid 0$

### A.2 Refined Grammar

$\langle \text{MODEL} \rangle \rightarrow \{ \langle \text{GAMES} \rangle \}$   
 $\langle \text{GAMES} \rangle \rightarrow \langle \text{GAME} \rangle \mid \langle \text{GAME} \rangle, \langle \text{GAMES} \rangle$   
 $\langle \text{GAME} \rangle \rightarrow \{ \langle \text{PERCEPTION} \rangle, \langle \text{PLAYERS} \rangle \}$   
 $\langle \text{PERCEPTION} \rangle \rightarrow \{ \text{playerName} \langle \text{PNAME} \rangle \} \mid \{ \}$   
 $\langle \text{PNAME} \rangle \rightarrow , \text{playerName} \langle \text{PNAME} \rangle \mid \epsilon$   
 $\langle \text{PLAYERS} \rangle \rightarrow \{ \langle \text{PLAYER} \rangle, \langle \text{PLAYER} \rangle \langle \text{NEXTPLAYER} \rangle \}$   
 $\langle \text{NEXTPLAYER} \rangle \rightarrow , \langle \text{PLAYER} \rangle \langle \text{NEXTPLAYER} \rangle \mid \epsilon$   
 $\langle \text{PLAYER} \rangle \rightarrow \{ \text{playerName}, \langle \text{OPTIONS} \rangle, \langle \text{MUTEX} \rangle, \langle \text{PREFVEC} \rangle \}$   
 $\langle \text{OPTIONS} \rangle \rightarrow \{ \langle \text{OPTION} \rangle \langle \text{NEXTOPTION} \rangle \}$   
 $\langle \text{NEXTOPTION} \rangle \rightarrow , \langle \text{OPTION} \rangle \langle \text{NEXTOPTION} \rangle \mid \epsilon$

$\langle \text{OPTION} \rangle \rightarrow \{\text{optionID}, \text{optionDescription}\}$   
 $\langle \text{MUTEX} \rangle \rightarrow \{\langle \text{PAIRS} \rangle\}$   
 $\langle \text{PAIRS} \rangle \rightarrow \{\text{optionID}, \text{optionID}\} \langle \text{NEXTPAIR} \rangle \mid \{\}$   
 $\langle \text{NEXTPAIR} \rangle \rightarrow , \langle \text{PAIRS} \rangle \mid \epsilon$   
 $\langle \text{PREFVEC} \rangle \rightarrow \{\langle \text{OUTCOME} \rangle \langle \text{OUTCOMES} \rangle\}$   
 $\langle \text{OUTCOMES} \rangle \rightarrow , \langle \text{OUTCOME} \rangle \langle \text{OUTCOMES} \rangle \mid \epsilon$   
 $\langle \text{OUTCOME} \rangle \rightarrow \{\langle \text{OPT} \rangle \langle \text{NEXTOPT} \rangle\}$   
 $\langle \text{OPT} \rangle \rightarrow \text{optionID}=\text{val}$   
 $\langle \text{NEXTOPT} \rangle \rightarrow , \langle \text{OPT} \rangle \langle \text{NEXTOPT} \rangle \mid \epsilon$

## Appendix B

# Hypergames Analysed

### B.1 Cuban Missile Crisis Hypergame

A simple game of the Cuban Missile Crisis.

```

{
  {
    {},
    {
      {
        USA,{{#1,Air Strike},{#2,Blockade}},{ },
        {
          {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},
          {#1=1,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=1,#2=1,#3=0,#4=0},{#1=0,#2=0,#3=0,#4=0},{#1=1,#2=1,#3=0,#4=1},
          {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=1}
        }
      },
      {
        USSR,{{#3,Withdraw},{#4,Escalate}},{{#3,#4}},
        {
          {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
          {#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=1,#2=1,#3=1,#4=0},{#1=1,#2=1,#3=0,#4=0},{#1=1,#2=1,#3=0,#4=1},
          {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=1}
        }
      }
    }
  }
}

```

## B.2 Operation Overlord

Operation Overlord is a game modeling the D-Day landings in World War II. It is a third level hypergame played by the Allies and Germany.

```
{
  {
    {Allies,Allies,Allies},
    {
      {
        Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
        {
          {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
          {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
          {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=0}
        }
      },
      {
        Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
        {
          {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1},
          {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0}
        }
      }
    }
  },
  {
    {Germany,Allies,Allies},
    {
      {
        Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
        {
          {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
          {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
          {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
        }
      },
      {
        Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
        {
          {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1},
          {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0}
        }
      }
    }
  }
},
```

```

{
  {Allies,Germany,Allies},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
      {
        {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
      }
    },
    {
      Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
      {
        {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
        {#1=0,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},
        {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1}
      }
    }
  }
},

{
  {Germany,Germany,Allies},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
      {
        {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
      }
    },
    {
      Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
      {
        {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1},
        {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0}
      }
    }
  }
},

{
  {Allies,Allies,Germany},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},

```

```

{
  {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
  {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0}, {#1=1,#2=0,#3=1,#4=0},
  {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
}
},
{
  Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
  {
    {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
    {#1=0,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},
    {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1}
  }
}
},
{
  {Germany,Allies,Germany},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
      {
        {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
        {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=0}
      }
    },
    {
      Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
      {
        {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
        {#1=0,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},
        {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1}
      }
    }
  }
},
{
  {Allies,Germany,Germany},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
      {
        {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
      }
    },
  }
},

```

```

{
  Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
  {
    {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
    {#1=0,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},
    {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1}
  }
}
},
{
  {Germany,Germany,Germany},
  {
    {
      Allies,{{#1,Invade Normandy},{#2,Invade Calais}},{{#1,#2}},
      {
        {#1=0,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=1,#2=0,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=0},{#1=1,#2=0,#3=1,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=0}
      }
    },
    {
      Germany,{{#3,Defend Normandy},{#4,Defend Calais}},{{#3,#4}},
      {
        {#1=0,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=0,#4=1},
        {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
        {#1=0,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0}
      }
    }
  }
}
}

```

### B.3 Fall Of France

```

{
  %% France's hypergame
  {
    {France,France},
    {
      {France,{{#1,Defend Maginot Line},{#2,Move North}},{{#1,#2}},
      {
        {#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=1},
        {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=0,#4=1}
      }
    },
  }
}

```

```

{Germany,{{#3,Attack Maginot Line},{#4,Attack In The North}},{{#3,#4}},
  {
    {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},
    {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0}
  }
}
},
{
  {Germany,France},
  {
    {France,{{#1,Defend Maginot Line},{#2,Move North}},{{#1,#2}}},
    {
      {#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=1},
      {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=0,#4=1}
    }
  },
  {Germany,{{#3,Attack Maginot Line},{#4,Attack In The North}},{{#3,#4}},
    {
      {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},
      {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0}
    }
  }
},
%% Germany's hypergame
{
  {France,Germany},
  {
    {France,{{#1,Defend Maginot Line},{#2,Move North}},{{#1,#2}}},
    {
      {#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=1},
      {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=0,#4=1}
    }
  },
  {Germany,{{#3,Attack Maginot Line},{#4,Attack In The North}},{{#3,#4}},
    {
      {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},
      {#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0}
    }
  }
},
{
  {Germany,Germany},
  {
    {France,{{#1,Defend Maginot Line},{#2,Move North}},{{#1,#2}}},

```



```

    {#1=1,#2=0,#3=0,#4=1,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=1,#2=0,#3=0,#4=1,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=0,#9=1}
  }
},

{
  US Opposition, {{#4, Legal Action}}, {},
  {
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=0,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=0,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=1,#7=0,#8=0,#9=0},
    {#1=1,#2=0,#3=0,#4=1,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=1,#2=0,#3=0,#4=1,#5=1,#6=0,#7=1,#8=0,#9=0},
    {#1=1,#2=0,#3=0,#4=1,#5=1,#6=1,#7=0,#8=0,#9=0},
    {#1=1,#2=0,#3=0,#4=1,#5=0,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=1,#7=0,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=0,#6=1,#7=0,#8=0,#9=0}
  }
},

{
  Canadian Opposition, {{#5, Treaty}}, {},
  {
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=0,#8=0,#9=1},
    {#1=0,#2=1,#3=0,#4=1,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=1,#6=0,#7=0,#8=1,#9=0},
    {#1=0,#2=1,#3=0,#4=1,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=1,#3=0,#4=0,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=0,#6=0,#7=1,#8=0,#9=0},
    {#1=0,#2=0,#3=1,#4=0,#5=0,#6=0,#7=1,#8=0,#9=0}
  }
}

```



## B.5 Altered Cuban Missile Crisis 1

A modified Cuban Missile Crisis game where the USSR is more aggressive and prefers to retaliate against hostile actions. The USA remains the same.

```

%%Modified Cuban Missile Crisis Game
{
  {
    {},
    {
      {
        USA,{{#1,Air Strike},{#2,Blockade}},{ },
        {
          {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},
          {#1=1,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=1,#2=1,#3=0,#4=0},{#1=0,#2=0,#3=0,#4=0},{#1=1,#2=1,#3=0,#4=1},
          {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=1}
        }
      },
      {
        USSR,{{#3,Withdraw},{#4,Escalate}},{{#3,#4}},
        {
          %% USSR is more aggressive
          {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
          {#1=1,#2=1,#3=0,#4=1},{#1=1,#2=0,#3=1,#4=0},{#1=1,#2=0,#3=0,#4=1},
          {#1=1,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
          {#1=1,#2=1,#3=0,#4=0},{#1=0,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=0,#4=1}
        }
      }
    }
  }
}

```

## B.6 Altered Cuban Missile Crisis 2

A different modification to the Cuban Missile Crisis game, where the USA is more aggressive and the USSR is more stubborn in wanting the missiles to be removed.

```

%%Cuban Missile Crisis Game
{
  {
    {},
    {
      {

```

```

USA,{{#1,Air Strike},{#2,Blockade}},{ },
{
  %%More aggressive US
  {#1=1,#2=1,#3=1,#4=0},{#1=1,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
  {#1=0,#2=0,#3=1,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
  {#1=1,#2=1,#3=0,#4=0},{#1=0,#2=0,#3=0,#4=0},{#1=1,#2=1,#3=0,#4=1},
  {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=1}
}
},
{
  USSR,{{#3,Withdraw},{#4,Escalate}},{{#3,#4}},
  {
    %%USSR stubborn, missiles will stay
    {#1=0,#2=0,#3=0,#4=0},{#1=0,#2=1,#3=0,#4=0},{#1=1,#2=0,#3=0,#4=0},
    {#1=1,#2=1,#3=0,#4=0},{#1=1,#2=1,#3=1,#4=0},{#1=0,#2=1,#3=1,#4=0},
    {#1=1,#2=0,#3=1,#4=0},{#1=0,#2=0,#3=1,#4=0},{#1=1,#2=1,#3=0,#4=1},
    {#1=1,#2=0,#3=0,#4=1},{#1=0,#2=1,#3=0,#4=1},{#1=0,#2=0,#3=0,#4=1}
  }
}
}
}
}
}

```

# Appendix C

## Software Notes

### C.1 Hypergame Model Data Structure

The following details the data structure used for modelling hypergames in Hypant, which can be seen in C.1.

### C.2 Application Programming Interface

These functions are those that would be useful for those who wish to use the existing hypergame analysis engine in another program or to extend the existing program.

#### C.2.1 Client

```
void textcolour(int attr, int fg, int bg);
```

The `textcolour` function takes three integers, representing values for the desired text colour. It then changes the colour of the terminal output to this new colour. Of its arguments `attr` is the attribute that the text can take such as dim, bright or underline, `fg` is the colour of the text and `bg` is the background colour. Values for these arguments are defined in `client.h`.

```
int createColourTable(Model* modelPtr);
```

This function takes a pointer to a hypergame model and creates a lookup table that matches a player's name to their assigned colour. Due to limitations with the number of available colours, only the first 11 players will be assigned colours. The first player in the game is assigned one colour, the second player the next colour and so on, until either all the players have colours, or if there are more than 11 players no more colours can be allocated. These players will have no colour assigned to them.

This function returns 0 if no colour table can be created or 1 if it completes successfully.

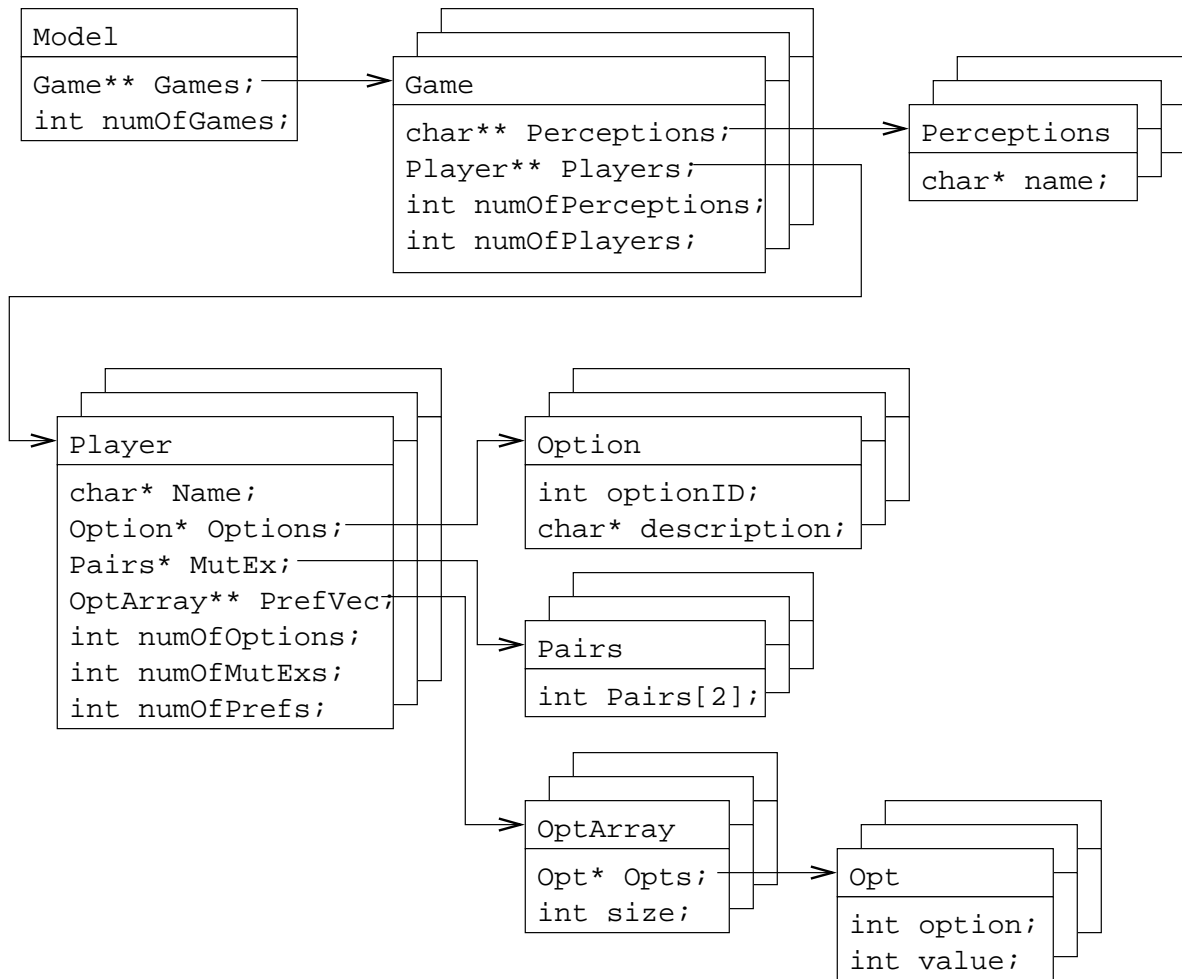


Figure C.1: Hypergame Model Data structure

```
void changeColour(char* aPlayerName);
```

Takes a player's name, looks up their name in the colour table and calls `textcolour` to change the colour ready for some output. If the name cannot be found, the colour will not be changed.

```
void displayModel(Model* modelPtr);
```

Takes a pointer to a model of the game and displays all the games inside that model.

```
void displayResults(Results* results, Model* modelPtr);
```

Takes a pointer to game results and a pointer to a game model and displays the results calculated for that game. The results are calculated by `analyseModel`, which means that they are the results for the individual games that make up the hypergame. The results will only be displayed if the number of results matches the number of games.

```
void deleteResults(Results* results);
```

Takes a pointer to a results structure and frees all the elements in it. After this function has been called, the pointer passed will be a dangling reference and should not be used again, until it has been assigned new results.

```
void displayMenu(char* inputfile, Model* modelPtr);
```

Displays a menu of available options, based on the game that is currently open, if any. Takes the name of the currently open file and a pointer to the model contained in it. If no game is open, these may be NULL. If the game is a hypergame, then the option to analyse the hypergame will be selectable, otherwise it will be greyed out.

```
void displayHyperResults(HyperResults* results);
```

Takes the results for a hypergame and displays the equilibriums for the various games. This function isn't called directly, but is called by `makeHyperModels` after it has constructed the results to be displayed.

## C.2.2 Server

```
int findGame(Model* model, char** Perception, int numOfPerceptions);
```

The `findGame` function takes a model pointer, a perception and the size of a perception and proceeds to search the games that make up the model for one that has a perception that matches the one passed. If a matching game is found, its index in the model is returned, otherwise -1 is returned.

This function is called by `makeModel`, which constructs a number of models that can in turn be analysed to give the results for various elements of a hypergame. It is used to find suitable games to include in the new models being made.

```
void hyperAnalyse(Model* someGames);
```

Takes a number of games, placed together in a model and makes a new game, that represents the hypergame made up of the initial games sent. This new game is then analysed and the results displayed. This function is called from `makeHyperModels`, which joins several games together to create the model passed to this function.

```
void makeHyperModels(Model* modelPtr);
```

Takes a pointer to a model of a game and from this constructs a model for each of the hypergames that contains the low level, original games that are used to create the hypergame. Each of these models is later passed to `hyperAnalyse` for analysis and output of results.

```
Results* analyseModel(Model* modelPtr);
```

Takes a pointer to a model and calculates the results for the individual games in that model. These are packaged together into a results structure, which is returned. This can later be displayed by `displayResults`.

```
void deleteHyperResults(HyperResults* myResults);
```

Takes a pointer to the results from a hypergame analysis structure and frees all the elements in this structure. This function is used internally by `makeHyperModels`.

### C.2.3 Parser

```
Model* parseGame(char* filename, int type);
```

Takes a pointer to a string and a operation type. If type is `FROM_FILE` then filename is the name of a file that the game can be read from and parsed. This is the default method of operation. If type is `FROM_MEM` then filename is a pointer to a memory buffer where the contents of the file has been read in previously. This mode currently does not operate correctly. If a game fails to parse, then memory allocated during the failed parse is not freed and will be leaked.

### C.2.4 General

```
int testModel(Model* modelPtr);
```

Takes a pointer to a game model and performs sanity checks on that the game, after it has been parsed. Returns 1 if the model passes and 0 if it fails. Some tests to

be performed by this function include making sure that each player in a hypergame has a correct number of games for that situation and making sure that options are numbered consistently across games. This function is currently empty, as no testing is performed on the models.

```
void deleteModel(Model* modelPtr);
```

Takes a pointer to a game model and frees all the elements in this structure. After calling this function the model pointer should be set to NULL to avoid dangling pointer problems.

```
void saveLocalModel(Model* modelPtr, char* file);
```

Takes a pointer to a model and a pointer to a filename and saves the program into the current directory on the local machine. This will write from the game model structure, so any comments from the original input file will not reappear in the output, as they were discarded earlier during the parsing. The layout of the output will also be different from that of the earlier input, however the new game file will still parse.

### C.3 Use of Colours

● Black	● Green	● Red	● Purple
● Dark Grey	● Lime	● Light Red	● Magenta
● White	● Dark Blue	● Orange	● Teal
Bright White	● Light Blue	● Yellow	● Cyan

Table C.1: Available terminal colours

<i>Player</i>	<i>Colour</i>
1	● Light Red
2	● Light Blue
3	● Teal
4	● Green
5	● Yellow
6	● Purple
7	● Dark Blue
8	● Orange
9	● Lime
10	● Cyan
11	● Magenta

Table C.2: Allocation of available colours to players

## Appendix D

### Screenshots

```
*****
* The overall stability shows which outcomes are *
* possible resolutions to the conflict. *
*****

Equilibrium(s)

Equilibrium #1:
USA Options
- None taken
USSR Options
+ Withdraw

Equilibrium #2:
USA Options
+ Blockade
USSR Options
+ Withdraw
```

Figure D.1: Cuban Missile Crisis Equilibriums

Game is :

Player: USA					Stability	
	Preference Vector					UIs
	Outcome					
	#1	#2	#3	#4		
1	0	0	1	0	r	-
2	0	1	1	0	s	1
3	1	0	1	0	u	1, 2
4	1	1	1	0	u	1, 2, 3
5	0	1	0	0	r	-
6	1	0	0	0	u	5
7	1	1	0	0	u	5, 6
8	0	0	0	0	u	5, 6, 7
9	1	1	0	1	r	-
10	1	0	0	1	u	9
11	0	1	0	1	u	9, 10
12	0	0	0	1	u	9, 10, 11

Player: USSR					Stability	
	Preference Vector					UIs
	Outcome					
	#1	#2	#3	#4		
1	0	0	0	0	r	-
2	0	0	1	0	s	1
3	0	1	1	0	r	-
4	0	1	0	0	u	3
5	1	0	1	0	r	-
6	1	0	0	0	u	5
7	1	1	1	0	r	-
8	1	1	0	0	u	7
9	1	1	0	1	u	7, 8
10	1	0	0	1	u	5, 6
11	0	1	0	1	u	3, 4
12	0	0	0	1	u	1, 2

Figure D.2: Cuban Missile Crisis Stability Table

```

Model contains 1 game(s).

Player 1 - USA
Options for USA :
#1 - Air Strike
#2 - Blockade
USA has no mutually exclusive options

Preference Vector
Options
#1 #2 #3 #4
0 0 1 0      Most Preferred
0 1 1 0
1 0 1 0
1 1 1 0
0 1 0 0
1 0 0 0
1 1 0 0
0 0 0 0
1 1 0 1
1 0 0 1
0 1 0 1
0 0 0 1      Least Preferred

Player 2 - USSR
Options for USSR :
#3 - Withdraw
#4 - Escalate
3 and 4 are mutually exclusive for USSR

Preference Vector
Options
#1 #2 #3 #4
0 0 0 0      Most Preferred
0 0 1 0
0 1 1 0
0 1 0 0
1 0 1 0
1 0 0 0
1 1 1 0
1 1 0 0
1 1 0 1
1 0 0 1
0 1 0 1
0 0 0 1      Least Preferred

```

Figure D.3: Cuban Missile Crisis Game

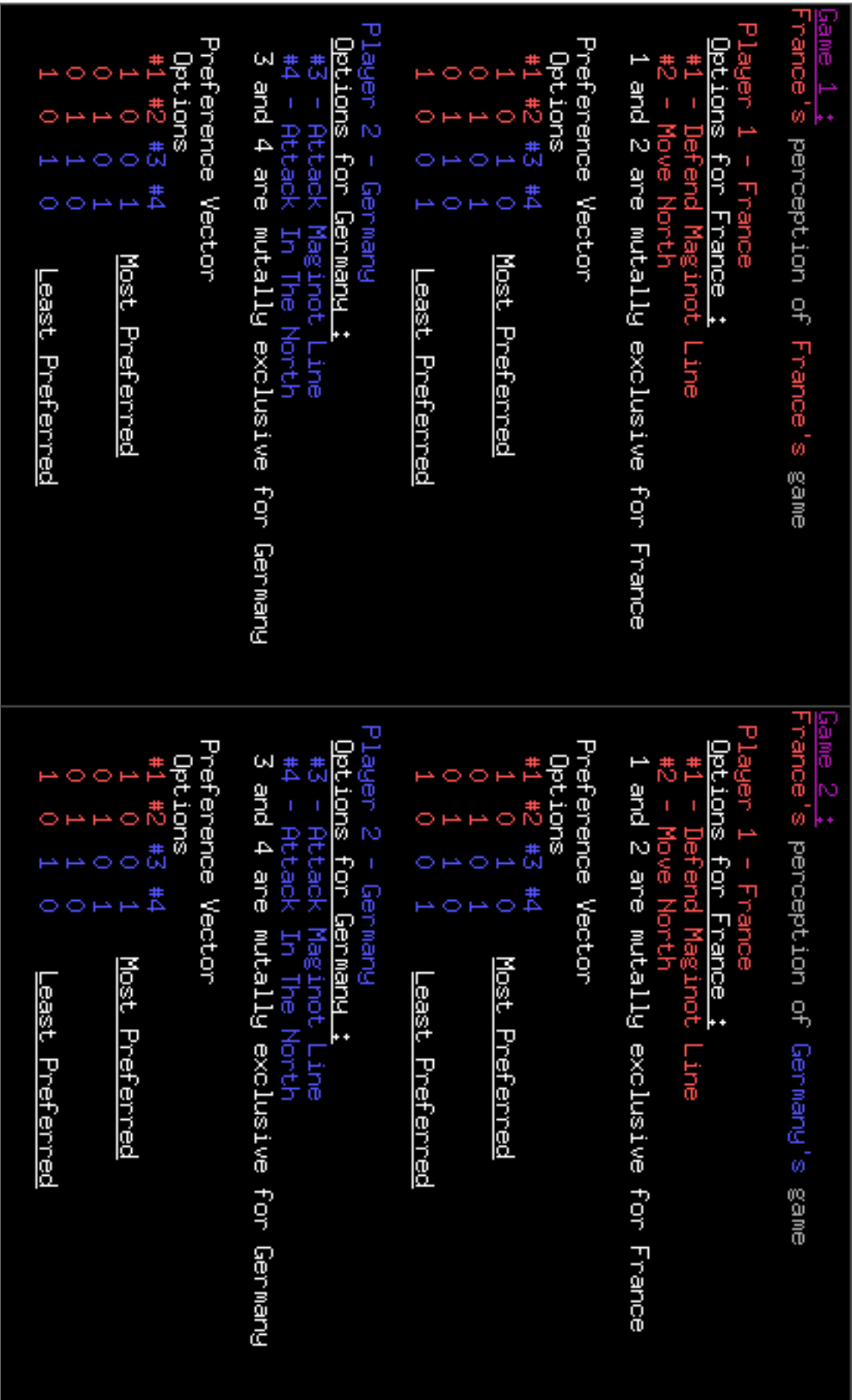


Figure D.4: Fall Of France - France's Hypergame

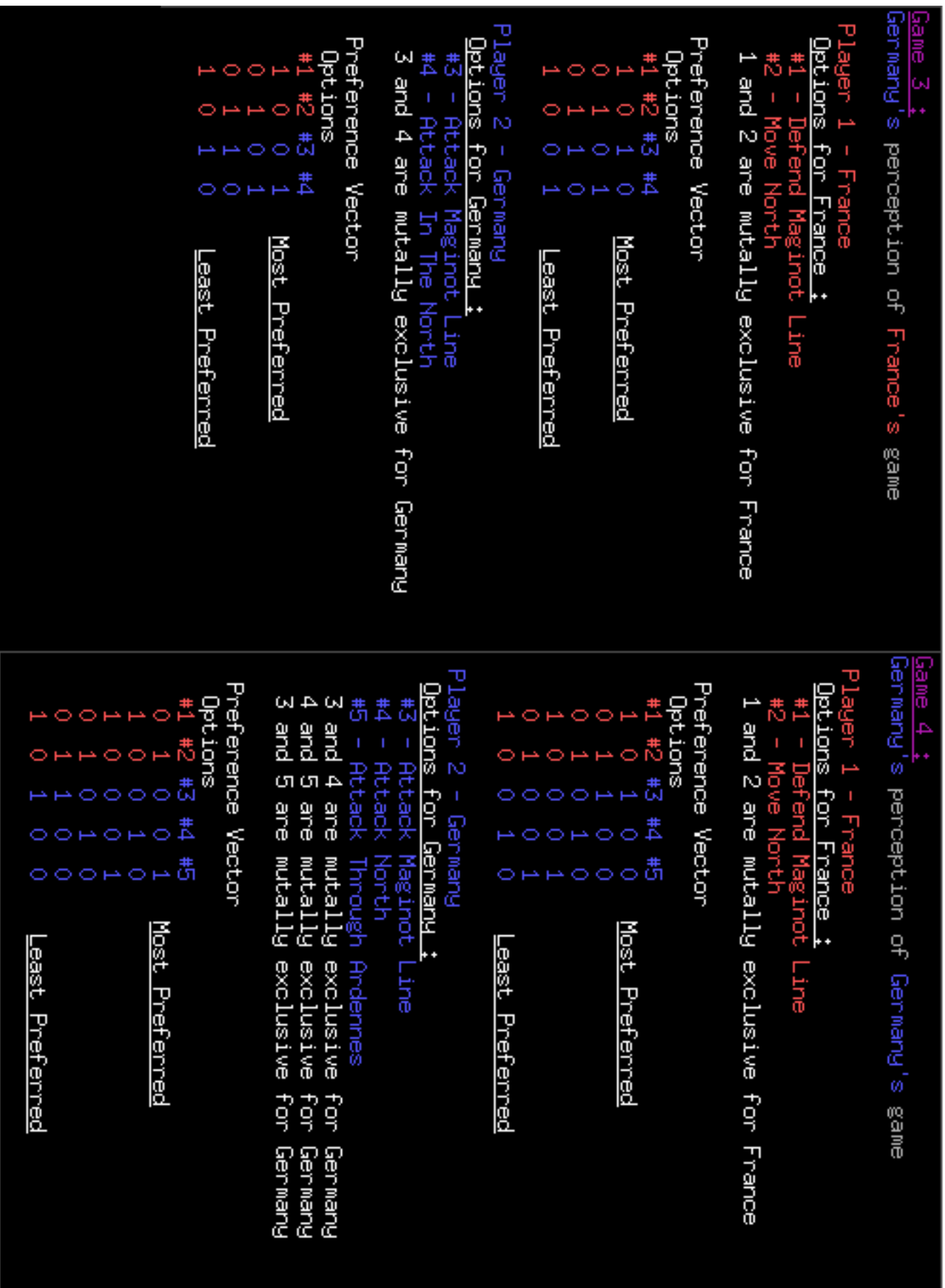


Figure D.5: Fall Of France - Germany's Hypergame

```

*****
* The Hypergame analysis shows the equilibriums for the *
* various levels of hypergames.                          *
*****

>>>-- 2nd level Hypergame results --<<<
Overall Hypergame
Equilibrium(s)

Equilibrium #1:
France Options
+ Move North
Germany Options
+ Attack Through Ardennes

Equilibrium #2:
France Options
+ Defend Maginot Line
Germany Options
+ Attack Through Ardennes

>>>-- 1st level Hypergame results --<<<
Hypergame is for France's game
Equilibrium(s)

Equilibrium #1:
France Options
+ Move North
Germany Options
+ Attack In The North

Hypergame is for Germany's game
Equilibrium(s)

Equilibrium #1:
France Options
+ Move North
Germany Options
+ Attack Through Ardennes

Equilibrium #2:
France Options
+ Defend Maginot Line
Germany Options
+ Attack Through Ardennes

```

Figure D.6: Fall of France Hypergame Analysis

```

Game is ↓
Player: US Support
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4 #5 #6 #7 #8 #9
1  1 0 0 1 0 1 0 0 0
2  1 0 0 1 1 1 0 0 0
3  0 1 0 0 0 1 0 0 0
4  0 0 1 0 0 1 0 0 0
5  0 1 0 1 0 1 0 0 0
6  0 1 0 0 1 1 0 0 0
7  0 0 1 0 1 1 0 0 0
8  0 1 0 1 1 1 0 0 0
9  0 1 0 0 0 0 1 0 0
10 0 1 0 0 1 0 1 0 0
11 0 1 0 1 0 0 1 0 0
12 0 1 0 1 1 0 1 0 0
13 0 0 1 0 0 0 1 0 0
14 0 0 1 0 1 0 1 0 0
15 1 0 0 1 1 0 1 0 0
16 0 0 1 0 1 0 0 1 0
17 0 1 0 0 1 0 0 1 0
18 1 0 0 1 1 0 0 1 0
19 0 1 0 1 1 0 0 1 0
20 0 0 1 0 1 0 0 0 1
21 0 1 0 0 1 0 0 0 1
22 1 0 0 1 1 0 0 0 1
23 0 1 0 1 1 0 0 0 1

Player: US Opposition
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4 #5 #6 #7 #8 #9
1  0 0 1 0 1 0 0 0 1
2  0 0 1 0 1 0 0 1 0
3  0 0 1 0 1 0 1 0 0
4  0 0 1 0 0 0 1 0 0
5  0 0 1 0 1 1 0 0 0
6  0 0 1 0 0 1 0 0 0
7  0 1 0 1 1 0 0 0 1
8  0 1 0 1 1 0 0 1 0
9  0 1 0 1 1 0 1 0 0
10 0 1 0 1 0 0 1 0 0
11 0 1 0 1 1 1 0 0 0
12 0 1 0 1 0 1 0 0 0
13 1 0 0 1 1 0 0 0 1
14 1 0 0 1 1 0 0 1 0
15 1 0 0 1 1 0 1 0 0
16 1 0 0 1 1 1 0 0 0
17 1 0 0 1 0 1 0 0 0
18 0 1 0 0 1 0 0 0 1
19 0 1 0 0 1 0 0 1 0
20 0 1 0 0 1 0 1 0 0
21 0 1 0 0 0 0 1 0 0
22 0 1 0 0 1 1 0 0 0
23 0 1 0 0 0 1 0 0 0

```

Figure D.7: Garrison Analysis (1 of 2)

Player: Canadian Opposition										
	Preference Vector									Stability
	Outcome									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	UIs
1	0	1	0	1	1	0	0	0	1	-
2	0	1	0	0	1	0	0	0	1	-
3	0	0	1	0	1	0	0	0	1	-
4	0	1	0	1	1	0	0	1	0	-
5	0	1	0	0	1	0	0	1	0	-
6	0	0	1	0	1	0	0	1	0	-
7	0	1	0	1	0	0	1	0	0	-
8	0	1	0	0	0	0	1	0	0	-
9	0	0	1	0	0	0	1	0	0	-
10	0	1	0	1	1	0	1	0	0	7
11	0	1	0	0	1	0	1	0	0	8
12	0	0	1	0	1	0	1	0	0	9
13	0	0	1	0	0	1	0	0	0	-
14	0	1	0	1	0	1	0	0	0	-
15	0	1	0	0	0	1	0	0	0	-
16	0	1	0	1	1	1	0	0	0	14
17	0	1	0	0	1	1	0	0	0	15
18	0	0	1	0	1	1	0	0	0	13
19	1	0	0	1	1	0	0	0	1	-
20	1	0	0	1	1	0	0	1	0	-
21	1	0	0	1	1	0	1	0	0	-
22	1	0	0	1	0	1	0	0	0	-
23	1	0	0	1	1	1	0	0	0	22

Player: IJC										
	Preference Vector									Stability
	Outcome									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	UIs
1	0	1	0	0	0	1	0	0	0	-
2	0	0	1	0	0	1	0	0	0	-
3	1	0	0	1	0	1	0	0	0	-
4	0	1	0	1	0	1	0	0	0	-
5	0	1	0	0	1	1	0	0	0	-
6	0	0	1	0	1	1	0	0	0	-
7	1	0	0	1	1	1	0	0	0	-
8	0	1	0	1	1	1	0	0	0	-
9	0	1	0	0	0	0	1	0	0	1
10	0	0	1	0	0	0	1	0	0	2
11	0	1	0	1	0	0	1	0	0	4
12	0	1	0	0	1	0	1	0	0	5
13	0	0	1	0	1	0	1	0	0	6
14	1	0	0	1	1	0	1	0	0	7
15	0	1	0	1	1	0	1	0	0	8
16	0	1	0	0	1	0	0	1	0	5, 12
17	0	0	1	0	1	0	0	1	0	6, 13
18	1	0	0	1	1	0	0	1	0	7, 14
19	0	1	0	1	1	0	0	1	0	8, 15
20	0	1	0	0	1	0	0	0	1	5, 12, 16
21	0	0	1	0	1	0	0	0	1	6, 13, 17
22	1	0	0	1	1	0	0	0	1	7, 14, 18
23	0	1	0	1	1	0	0	0	1	8, 15, 19

Figure D.8: Garrison Analysis (2 of 2)

```

Game is :
Player: USA
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4
1      0  0  1  0      r      -
2      0  1  1  0      s      1
3      1  0  1  0      s      1, 2
4      1  1  1  0      s      1, 2, 3
5      0  1  0  0      r      -
6      1  0  0  0      u      5
7      1  1  0  0      u      5, 6
8      0  0  0  0      u      5, 6, 7
9      1  1  0  1      r      -
10     1  0  0  1      u      9
11     0  1  0  1      u      9, 10
12     0  0  0  1      u      9, 10, 11

Player: USSR
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4
1      0  0  0  0      r      -
2      0  1  0  0      r      -
3      1  0  0  0      r      -
4      1  1  0  1      r      -
5      1  0  1  0      s      3
6      1  0  0  1      s      3, 5
7      1  1  1  0      u      4
8      1  0  1  0      s      3
9      0  1  1  0      u      2
10     1  1  0  0      u      4, 7
11     0  0  1  0      u      1
12     0  0  0  1      u      1, 11

```

Figure D.9: Alternate Cuban Game 1 Stability Table

```

*****
* The overall stability shows which outcomes are          *
* possible resolutions to the conflict.                   *
*****

Equilibrium(s)

Equilibrium #1:
  USA Options
  + Air Strike
  USSR Options
  + Withdraw

Equilibrium #2:
  USA Options
  + Blockade
  USSR Options
  - None taken

Equilibrium #3:
  USA Options
  + Air Strike
  + Blockade
  USSR Options
  + Escalate

```

Figure D.10: Alternate Cuban Game 1 Equilibriums

```

Game is :
Player: USA
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4
1      1  1  1  0      r      -
2      1  0  1  0      s      1
3      0  1  1  0      s      1, 2
4      0  0  1  0      s      1, 2, 3
5      0  1  0  0      r      -
6      1  0  0  0      u      5
7      1  1  0  0      u      5, 6
8      0  0  0  0      u      5, 6, 7
9      1  1  0  1      r      -
10     1  0  0  1      u      9
11     0  1  0  1      u      9, 10
12     0  0  0  1      u      9, 10, 11

Player: USSR
Preference Vector      Stability
Outcome              UIs
#1 #2 #3 #4
1      0  0  0  0      r      -
2      0  1  0  0      r      -
3      1  0  0  0      r      -
4      1  1  0  0      r      -
5      1  1  1  0      u      4
6      0  1  1  0      u      2
7      1  0  1  0      u      3
8      0  0  1  0      u      1
9      1  1  0  1      u      4, 5
10     1  0  0  1      u      3, 7
11     0  1  0  1      u      2, 6
12     0  0  0  1      u      1, 8

```

Figure D.11: Alternate Cuban Game 2 Stability Table

```
*****  
* The overall stability shows which outcomes are *  
* possible resolutions to the conflict. *  
*****  
  
Equilibrium(s)  
  
Equilibrium #1:  
USA Options  
+ Blockade  
USSR Options  
- None taken
```

Figure D.12: Alternate Cuban Game 2 Equilibriums

# References

- Bennett, P. G. (1977). Towards a theory of Hypergame, *Omega* **5**: 749–751.
- Bennett, P. G. (1980a). Bidders and Dispenser: Manipulative Hypergames in a Multinational Context, *European Journal of Operational Research* **4**: 293–306.
- Bennett, P. G. (1980b). Hypergames : Developing a Model of Conflict, *Futures* **12**: 489–507.
- Bennett, P. G. and Dando, M. R. (1979). Complex Strategic Analysis: A Hypergame Study of the Fall of France, *Journal of the Operational Research Society* **30**(1): 23–32.
- Bennett, P. G., Dando, M. R. and Sharp, R. G. (1980). Using Hypergames to Model Difficult Social Issues: An Approach to the case of Soccer Hooliganism, *Journal of the Operational Research Society* **31**(7): 621–635.
- Bennett, P. G. and Huxham, C. S. (1982). Hypergames and What They Do: A ‘Soft O.R.’ Approach, *Journal of the Operational Research Society* **33**(1): 41–50.
- Bennett, P., Tait, A. and MacDonagh, K. (1993). Interact: Developing software for Interactive Decisions, *Group Decisions and Negotiations* **3**: 351–372.
- Fraser, N. and Hipel, K. W. (1979). Solving Complex Conflicts, *IEEE Transactions on Systems, Man and Cybernetics* **SMC-9**(12): 805–816.
- Fraser, N. M. (1993). Applications of Preference Trees, *1993 International Conference on Systems, Man and Cybernetics. Systems Engineering in the Service of Humans*, IEEE, pp. 132–136.
- Fraser, N. M. and Hipel, K. W. (1980). Metagame Analysis of the Poplar River Conflict, *Journal of the Operational Research Society* **31**: 377–385.
- Fraser, N. M. and Hipel, K. W. (1984). *Conflict Analysis, Models and Resolutions*, Elsevier Science Publishing Co. Inc., New York.
- Fraser, N. M. and Hipel, K. W. (1988). Using the DecisionMaker Computer Program for Analysing Environmental Conflicts, *Journal of Environmental Management* **27**: 213–228.
- Giesen, M. and Bennett, P. (1979). Aristotle’s Fallacy: A Hypergame in the Oil Shipping Business, *Omega* **7**(4): 309–320.
- Harsanyi, J. (1968). Games with incomplete information played by ‘Bayesian’ players, *Management Science* **14**: 159,320,486.
- Howard, N. (1971). *Paradoxes of Rationality : Theory of Metagames and Political Behaviour*, M.I.T. Press.

- Howard, N. (1987). The Present and Future of Metagame Analysis, *European Journal of Operational Research* **32**: 1–25.
- Kopp, C. (2002). Shannon, Hypergames and Information Warfare, in W. Hutchinson (ed.), *Proceedings of the 3rd Australian Information Warfare & Security Conference 2002 (IWAR 2002)*.
- Leclerc, M. and Chaib-draa, B. (2002). Hypergame Analysis in E-Commerce: A Preliminary Report. Accessed 18/5/2003  
<http://www.cirano.qc.ca/pdf/publication/2002-s66.pdf>.
- Mèrö, L. (1998). *Moral Calculations*, Copernicus, New York.
- Morgenstern, O. and von Neumann, J. (1953). *Theory of Games and Economic Behaviour*, Princeton University Press, Princeton.
- Peter Savich, K. W. H. and Fraser, N. M. (1983). The Alaskan Gas Pipeline Conflict, *Energy* **8**(3): 213–224.
- Takahashi, M. A., Fraser, N. M. and Hipel, K. W. (1984). A procedure for analyzing hypergames, *European Journal of Operational Research* **18**: 111–122.
- Wang, M., Hipel, K. W. and Fraser, N. M. (1988). Modeling Misperceptions in Games, *Behavioural Science* **33**: 207–233.
- Wang, M., Hipel, K. W. and Fraser, N. M. (1989). Solution Concepts in Hypergames, *Applied Mathematics And Computation* **34**: 147–171.