Formal Dialectical Games in Multiagent Argumentation

Simon Wells

A Thesis submitted for the degree of Doctor of Philosophy

School of Computing

University of Dundee

November 2006

Abstract

FORMAL dialectical games have been developed for use as tools in fallacy research. Recently such games have also found utility in multiagent systems (MAS) as a way to structure argumentative communication between intelligent agents. This thesis investigates dialectical games and proposes a unified representation for disparate games using a theoretical and computational framework named the "Architecture for Argumentation" (A4A). The need for such a framework is established as a way to move beyond the current *ad hoc* approaches to dialectical game development and is presented in the context of a contemporary move within MAS research towards frameworks for rapid development. Two deployments of the A4A are then explored. The first is in the "MAS and Dialectical Game Architecture" (MASADA) a component-oriented software framework for developing arguing agents. The second deployment is in the *i*-Xchange project, an investigation of the development of arguing agents that incorporate elements of planning, reasoning and argumentation. Finally the need to evaluate dialectical games is established. Two approaches to evaluation are explored including the use of MASADA components to provide an automated testbed for the comparative evaluation of dialectical games. The knowledge domain used by the agents in the testbed is graph-colouring, which is explored from two perspectives in relation to dialectical games. The first perspective is that of a test bed for evaluating dialectical games, and the second perspective is that of a problem domain in which dialectical games provide tools for solving distributed graph-colouring problems. Dialectical games are thus demonstrated to be more than merely a way to regulate communicative acts between agents but can also be used as a tool for solving distributed problems.

Acknowledgements

IN no particular order...

Chris Reed,

Glenn Rowe,

Everybody in the School of Computing, University of Dundee,

The Information Exchange; Nick Jennings, Tim Norman, Dionysis Kalofonos, Nishan Karunatillake,

The Engineering and Physical Sciences Research Council (EPSRC),

All of the peers & colleagues I met along the way,

Assorted friends, family, & foster siblings,

Mum,

Dad,

Thom,

Lee,

Jamie,

(Last but not least) Sabine.

WHAT a long strange trip it's been. Words are not sufficient. Thanks.

Declaration of the Candidate

I declare that I am the author of this thesis, and that, unless otherwise stated in the text all references cited have been consulted by me; that the work of which this thesis is a record has been done by myself; and has not been previously presented or accepted for a higher degree.

S. Wells

Declaration of the Supervisors

WE declare that Simon Wells has satisfied all the terms and conditions of the regulations made under Ordinances 12 and 39, and has compelte the required nine terms of research to qualify in submitting this thesis in application for the degree of Doctore of Philosophy.

C. A. Reed

G. W. A. Rowe

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With them the Seed of Wisdom did I sow,

And with my own hand labour'd it to grow:

And this was all the Harvest that I reap'd -

"I came like Water, and like Wind I go."

– Edward Fitzgerald, *Translation of the Rubaiiat of Omar Khayyam, Quatrain XXVIII*

FORMAL dialectical games have been developed for use as tools in fallacy research. They were suggested by Hamblin [44] as a way to model the interactions between participants in a dialogue in order to examine the circumstances in which certain logical fallacies, such as *petitio principii*, occur. More recently such games have found utility in a range of areas, for example in A.I. to model belief revision [40], in computer-based learning as a way to structure student-tutor interaction [86], in law as a way to explore evidential structures [13], and in argumentation research to investigate commitment [138] and instructional dialogues [41]. The field of multiagent systems (MAS) is an increasingly important research area that has grown out of the increasingly ubiquitous and distributed nature of modern computing [146]. MAS are constructed from individual intelligent agents that operate autonomously to pursue their goals. This pursuit can involve communication between agents as they share scarce resources, distribute tasks to capable agents, and form teams to perform complex tasks. Dialectical games are increasingly being adopted as a way to structure the communication between agents that is required to achieve these tasks [24, 81, 77].

This thesis investigates dialectical games in the context of MAS. It begins by examining a range of extant dialectical games and seeking to identify the commonalities between the disparate approaches.

Current research in dialectical games has tended towards individual or small group efforts. These efforts have resulted in many slight differences and incompatible modes of representation even though the games are built around the same core ideas. These differences are a hindrance when implementing dialectical games and does not help the wider adoption of dialectical games as tools in other fields of research. What is necessary to support the wider adoption of dialectical games is to standardise the different approaches into theoretical methodologies and software frameworks, i.e. a movement away from individual efforts towards an engineered approach. Engineering aims to be objective, independent from one individuals perception and not requiring unique skills. As such it should be reproducible, predictable and systematic. The distinction between individual, *ad hoc* work and engineering work is well expressed in the Software Engineering Institute (SEI) capability maturity model (CMM) [48]. The CMM has five levels which describe the maturity of an endeavour:

1. Initial: *Ad hoc*, individual efforts
2. Repeatable: A disciplined process
3. Defined: Institutionalised as a standard
4. Managed: Management and measurement of the standard
5. Optimising: Deliberate optimisation and improvement

It is clear that current efforts with regards to dialectical games are still at level one. In order to make a persuasive case for the widespread adoption of dialectical games within MAS requires that game development move from individual *ad hoc* efforts towards robust, repeatable software components developed as a part of a well-defined and disciplined process. One aim of this thesis is to begin this process of moving dialectical games further up the capability maturity model from the initial level one to the repeatable level two.

One method to support the movement from individual efforts towards an engineered methodology is to utilise a software framework that supports the wide range of components found in extant dialectical games and which reduces the effort and complexity of developing software that makes use of dialectical games. To this end a unified representation format of dialectical games is proposed as part of a theoretical and computational framework named the “Architecture for Argumentation” (A4A). Such a framework is a good way to move beyond the current *ad hoc* approaches to dialectical game development and is presented in the context of a contemporary move within MAS research towards frameworks for rapid development, such Jack [42], JUDE[67], and JADE [35], and a general trend in software engineering towards standard libraries and frameworks, such as the C++ Standard Library [122], and the C Standard Library [57].

Two deployments of the A4A in software applications are explored which tackle a range of issues. For example, just possessing a representation of the rules of a dialectical game does not help an agent to play the game well. Furthermore, even just playing the game, however badly, requires more capabilities of an agent than just knowing the rules. So the first deployment of the A4A is within a component of the "MAS and Dialectical Game Architecture" (MASADA) a component-oriented software framework for developing arguing intelligent agents and exploring related issues such as knowledge representation, reasoning, and strategic play. The second deployment is in the *i*-Xchange project, an investigation of the development of arguing agents that incorporate elements of planning, reasoning and argumentation. As agents become more sophisticated and possess a wider range of capabilities it will become necessary to ensure that these capabilities are properly integrated, work well together and that the agents themselves are developed as a cohesive unit of software.

For dialectical games to be widely accepted, both as communicative protocols and as tools for tackling problems, it is necessary to be able to state when a game is suitable for a given context of use. Such a goal requires methods and tools to be developed to support the evaluation of games. Two approaches to evaluation are explored; one uses the MASADA components to provide an automated testbed for the comparative evaluation of dialectical games, and the other is through the inspection of the game's rules and composition. The knowledge domain used by the agents in the testbed is graph-colouring, which is explored from two perspectives in relation to dialectical games. The first perspective is that of a test bed for the comparative evaluation of dialectical games, and the second perspective is that of a problem domain in which dialectical games provide tools for solving distributed graph-colouring problems. Dialectical games are thus demonstrated to be more than merely a way to regulate communicative acts between agents but can also be used as a tool for solving distributed problems.

There are a number of questions that this thesis tackles. Throughout the investigations, development activities and evaluations of arguing agents that form the core of this thesis, these questions have been a constant thread and can be summarised as follows:

1. What components do dialectical games use?
2. Can a unified representation of extant games and their constituent components be constructed?
3. Can a software framework be developed to support the rapid development and deployment of dialectical games, and to reduce the effort required to do so?
4. What additional capabilities do agents need in order to play dialectical games?
5. How do these additional capabilities affect the development of intelligent arguing agents when an agent:
   (a) is developed as a single cohesive software application?
   (b) has multiple developers who are incorporating competing capabilities?
6. Given the plethora of extant games and the huge space of possible games, how can games be evaluated in terms of their suitability to a given context?
7. Given a society of agents that possess dialectical game playing capabilities, can the agents being used to actually tackle certain problems as opposed to be used solely as a means to structure communicative acts?

1.1 Thesis Overview

This section presents a brief overview of the research explored in subsequent chapters.

Chapter 2 In this chapter an overview of relevant research literature is presented which explores argumentation theory, the nature of games, the relationship between games and dialogue, formal dialectical games, autonomous agents and multiagent systems, agent communication and interaction, argumentation in MAS, and testbeds and evaluative techniques.

Chapter 3 Chapter 3 explores techniques for analysing the rules of dialectical games before presenting the results of an analysis of some of the extant games from the fallacy and MAS research literature. This chapter seeks to collate the wide range of components and manipulations thereof that have been used in existing dialectical games.

Chapter 4 Based upon the results collated in chapter 3 a framework for representing dialectical games named the “Architecture for Argumentation” (A4A) is introduced. A number of extant games are reformulated into the A4A schema representation and a a computational implementation is presented.

Chapter 5 A component-oriented architecture called the “MAS and Dialectical Game Architecture” (MASADA) is introduced. Two components of MASADA are explored, the first, Colloqui, encapsulates an instance of the A4A along with a knowledge base and reasoning process. The second component, Caucus, implements domain specific behaviour modules to facilitate the development of different types of agent such as an arguing agent, an environment modelling agent, and a graphical user interface (GUI) agent.

Chapter 6 Approaches to evaluating dialectical games are explored and a range of both empirical and non-empirical metrics are identified. A non-empirical evaluation of a range of dialectical games is produced based upon the identified metrics. The application of MASADA components is explored as a testbed for the automated empirical evaluation of dialectical games. An empirical evaluation of communicative performance is carried out for a number of games. Finally an exploration of graph-colouring is presented in the context of dialectical game playing. Whereas graph-colouring provided a motivating knowledge domain in the empirical evaluations of dialectical games, dialectical games can, in turn, provide new techniques for finding solutions to distributed graph-colouring problems.

Chapter 7 In this chapter some final conclusions are drawn about this thesis as a whole, a summary of contributions is presented and some directions for future research are outlined.

Chapter 2

Background

THIS chapter presents an overview of relevant research literature. Topics are introduced which either underlie the dialectical game approach to dialogue protocols, such as argumentation theory, or which provide a motivation for various uses and deployments of dialectical games, such as autonomous agents and multiagent systems, or which provide either techniques or underlying theoretical basis for the approach taken in later chapters, such as concepts related to ludic games and the use of testbeds in computer science.

Section 2.1 gives an overview of Argumentation Theory. Because game playing has been a popular approach to building Artificial Intelligence testbeds as well as a way to model complex systems and interactions, section 2.2 looks at games from both the point of view of the formal context and that of the ludic context before looking at the ways that dialogue has been characterised both as a game and in terms of game related features. Section 2.4 introduces a popular game-oriented technique called Dialectical Games that are used to structure the interactions that occur during argumentative dialogues. A range of extant dialectical games are then introduced. Section 2.5 introduces Multiagent Systems in the context of large-scale, robust, distributed societies of autonomous intelligent agents. Section 2.6 motivates the need for communication between the member agents of a Multiagent System and surveys a number of approaches to structuring those communications. Section 2.7 looks at the applications of argumentation, particularly dialectical games, to multiagent communication. Section 2.8 looks at the ways that communications between agents have been tested, compared, and evaluated. Finally section 2.9 presents a summary of this chapter.

2.1 Argumentation Theory

Argumentation theory is a philosophical discipline concerned with the construction, identification, analysis and evaluation of arguments. Although there are many definitions of argument in the literature [135, pp.3-31] the term argument in this context refers to the use of reasons to support or criticise a claim [137, pp. 1]. The roots of argumentation theory return at least as far as the ancient Greek philosophers where the discipline played a part in the government of early democratic society [30, pp. 30]. Aristotle identifies three theories; *analytic*, *dialectic* and *rhetoric*. The analytic theory, now called logic, attempts to establish absolutely certain conclusions which follow necessarily from a given set of evidently true asserted or assumed premises and is commonly used to demonstrate proofs. The dialectical theory is the art of debate whose aims are to draw acceptable conclusions that follow from premises that are not evidently true. Finally the rhetorical theory is the art of persuading an audience through the use of cogent premises and conclusions such that whilst the rhetorical argument need not be logically or dialectically valid it still functions to persuade that audience towards which it is directed. Argumentation theory is mostly concerned with arguments which lie in the dialectical sphere. Such dialectical arguments are described as defeasible because the conclusions that follow generally or probably, but not necessarily, from their premises may be defeated by the addition of new information. This is in contrast to logical proofs where the recipient must accept the conclusion if they accept the premises and rules of inference that lead to the conclusion and new information is added monotonically so that previously established conclusions are preserved. Some overlapping topics pertinent to the current discussion that are of interest to researchers in argumentation theory are the fallacies, instances of errors in reasoning [51], enumerating the types of dialogue that can occur [138], modelling the interactions that occur between the participants of an argumentative dialogue[44], identifying the ways that individual arguments are used in real-world argumentation [97], and how such arguments can be expressed through argument representation schemas such as Toulmin's layout of argument [129], illustrated in figure 2.1 with an example in figure 2.2, and categorised using argumentation schemes [137].

Walton and Krabbe introduce a non-exhaustive typology of distinct dialogue types that can be identified in real-world communications [138]. The dialogue typology is introduced to support a notion of *licit* and *illicit* shifts between dialogue types which can correspond to instances of fallacies being committed and serve to explain how some fallacies occur in dialogue yet still appear reasonable [138, pp. 65]. The dialogue types include a number of simple types such as critical discussion, negotiation, inquiry,

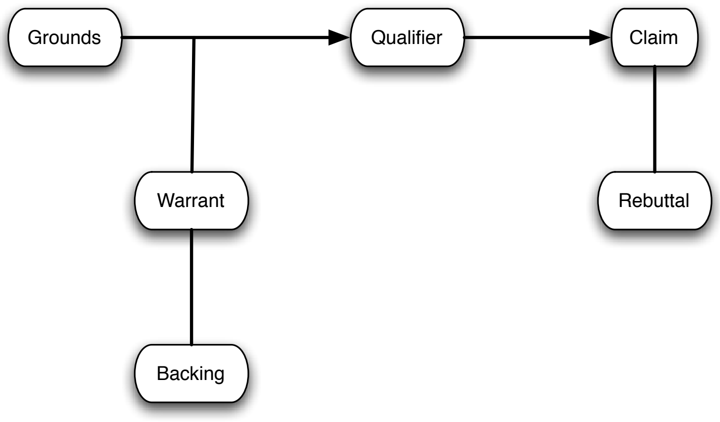

Figure 2.1: Toulmin's layout of argument

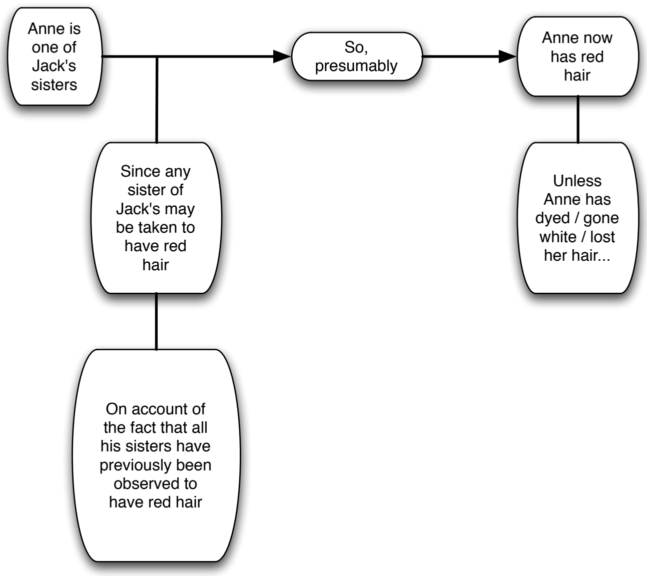
Anne is one of Jack's sisters
So, presumably
Anne now has red hair
Since any sister of Jack's may be taken to have red hair
Unless Anne has dyed / gone white / lost her hair...
On account of the fact that all his sisters have previously been observed to have red hair

Figure 2.2: Example argument diagrammed according to Toulmin's schema

information-seeking, deliberation, eristic, each of which may have more specialised sub-types, and complex dialogue types formed from compounds of the simple types. A good summary of the identified types and subtypes can be found in [138, pp. 66], and a condensed version is to be found in table 2.1. Dialogues are identified as conforming to a particular type on the basis of a characterisation of the initial situation, overall goal shared by the participants, and the participants individual aims which might be at odds.

Briefly, the aim of a critical discussion is to effect the resolution of a disagreement stemming from the participants conflicting points of view. During the dialogue each participant tries to get their point of view accepted by their opponent. A recent topic of research regarding persuasion-type dialogues has focussed on the issue of persuasion over action. Briefly, this topic is concerned with how an agent might be persuaded to take a particular course of action. One model of pursuasion over action is due to Atkinson [6] which identifies how autonomous agents may make, question, defend and jointly reason about proposals for action. This is achieved through the application of an argumentation scheme characterising *pursuasion over action* [6, pp. 63] which is based upon two of Walton's practical reasoning argumentation schemes, the *necessary condition schemes* and the *sufficient condition scheme* [132]. Both persuasion over action and critical discussion dialogues have been explored in many of the interaction protocols proposed in the argumentation theory and are discussed in section 2.4. The next three types, negotiation, inquiry, and deliberation, have all found utility in agent interaction protocols as discussed in the last section. Negotiation dialogues are characterised by a conflict of interests or need for co-operation in which the participants try to make a deal whilst simultaneously securing the best outcome for themselves. Inquiry dialogues start from a state of general ignorance and strive for a growth of knowledge. Deliberation dialogues stem from a need for action and the participants jointly aim to reach a decision whilst individually striving to influence the outcome. Information-seeking dialogues stem from personal ignorance and aim overall to spread knowledge or reveal a position with respect to the issue about which information is sought. Eristic dialogues are identified as a substitute for physical confrontation and stem from antagonism and conflict. In this context eristic dialogues are closer to the colloquial idea of an argument as a form of verbal confrontation than the other types of dialogue in the typology.

Rivals to Walton and Krabbe's typology include Baker's model of co-operation-in-interaction [10, pp. 131] and Dahlbäck's taxonomy [23], although these are not nearly so popular. There is also an attempt by Yuan to integrate the Walton and Krabbe and Baker models [148, pp. 10]. Baker's model is ge-

Dialogue Type	**Initial Situation**	**Overall Goal**	**Individual Goals**
Critical Discussion	Conflicting points of view	Verbal resolution of conflict	Persuade others
Negotiation	Conflict of interests & need for co-operation	Make a deal	Get best for oneself
Inquiry	General ignorance	Growth of knowledge and agreement	Find or destroy a proof
Deliberation	Need for action	Reach a decision	Influence outcome
Information Seeking	Personal ignorance	Spread knowledge & reveal positions	Gain, pass on, show or hide personal knowledge
Eristics	Conflict & antagonism	Reach (provisional) accommodation in a relationship	Strike the other party. Win in the eyes of onlookers

Table 2.1: The Walton & Krabbe dialogue typology

ometric and situates dialogue types upon three axis, symmetry, agreement, and alignment. The three axis enable Baker to identify eight basic types of dialogue which include, co-construction, apparent co-construction, co-argumentation, apparent co-argumentation, acquiescent co-elaboration, apparent acquiescent co-elaboration, one-sided argumentation, and apparent one-sided argumentation. The geometric arrangement of dialogues is shown in figure 2.3. The taxonomy that Dahlbäck's proposes categorises dialogue based on a non-exhaustive list of eight dimensions of dialogue which include modality, agent kind, interaction, context, number and type of tasks, task-distance, shared knowledge [23].

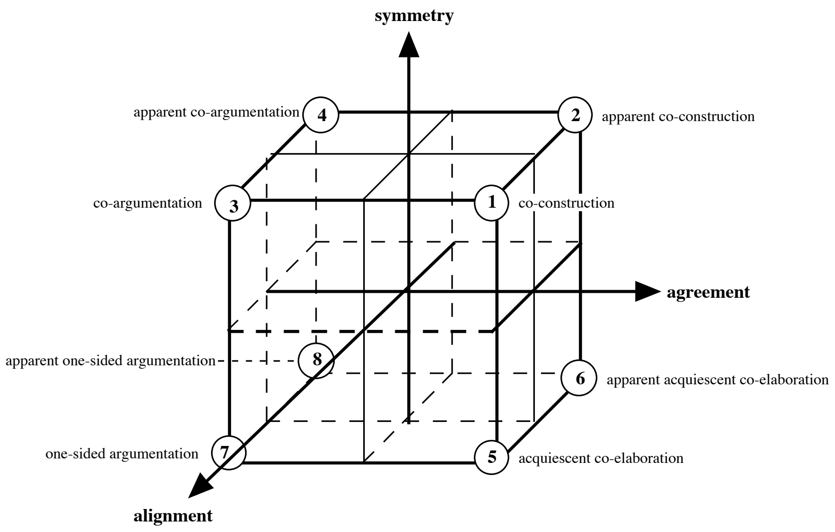
symmetry
apparent co-argumentation
4
2
apparent co-construction
co-argumentation
3
1
co-construction
agreement
apparent one-sided argumentation
8
6
apparent acquiescent co-elaboration
one-sided argumentation
7
5
acquiescent co-elaboration
alignment

Figure 2.3: Baker's Geometric Dialogue Typology

2.2 Games: Economic, Formal & Ludic

Games have a long history in computer science (CS), AI, and argumentation where they have been used to evaluate the current level of progress in a given domain, to situate examples and motivate testing, and as frameworks to guide the construction of models. Two concepts of games can be identified, the first is the concept of games used in economic game theory [89] to investigate social interaction in either co-operative [88] or non-co-operative modes [68]. The other concept of games stems from ludic theory and is predicated on the notion both of games as pastimes and as formal systems.

Since Claude Shannon first programmed a computer to play chess in 1950 [117], games have been identified with computer science and AI. The traditional application of games to AI has used the idea of mainly chess playing ability as a measure of progress within the field with the goal that intelligent programs should be able to compete with human game-playing champions [119]. Whilst lacking resources to implement software, Alan Turing managed to hand-simulate an algorithm for chess playing in 1953 [130] and the work of Alan Newell and Herb Simon lead to some of earliest fundamental results in AI [90]. Chess in particular has been described as the "*Drosophila* of AI" a phrase attributed to the Russian Mathematician Alexander Kronrod in 1965. A reference to the use of the fruit fly *Drosophila Melanogaster* in genetics research. The use of chess, or more correctly the approach taken to using chess in AI, does have critics, notably McCarthy who equates the use of chess in AI to the use of *Drosophila* in the biological sciences. McCarthy identifies that computer chess has developed similarly to how genetics would have developed if biologists had concentrated on breeding racing flies and concludes that biologists would have some science but would mainly have very fast fruit flies [83]. A recent criticism from the philosophical quarter sees AI as nothing more than a theory of games and game-playing as opposed to an investigation of intelligent behaviour through problem-solving [36].

Other games used in CS and AI run the gamut from popular traditional board and card games like Scrabble, Backgammon, Checkers, bridge, and poker, to less popular games like Othello, nine-mens morris, and connect-4, to more esoteric fare such as the Japanese game Go [113]. Ability to play these other games has met with mixed success, some games like connect-4 have been solved and a computer can play a "perfect" game against humans, whilst the ability of computers to play games like Go are still rudimentary and far below the ability of even average human players. Schaeffer identifies that games are popular domains for exploring the capabilities of intelligent systems because the rules are fixed, problem scope is constrained, and interactions are well defined [114].

In addition to acting as AI testbeds, the notion of games as formal systems is used to build models of interaction and to analyse real work phenomena. In [45] Haugeland considers a game to be a formal system in which tokens are manipulated according to a set of rules to see what configurations of tokens can be obtained. Three parameters are proposed that must be defined in order to specify such a game. The parameters are:

1. what the tokens are;
2. what the starting position is; and

3. what moves are allowed in any given position.

Implicit in Haugeland's treatment is the notion of the game board, the place where the tokens (from 1) are arranged (from 2) and moved (from 3). Whilst not all games are explicit in their representation of a game board their is nonetheless an implicit game board. If a game has tokens that are manipulated according to the rules of the game, then there is at least some abstract space in which those token manipulations occur. This abstract space constitutes the, explicit or implicit, game board. To ground this system consider the real-world game of Solitaire, the starting position of a solitaire game is shown in illustration 2.4.

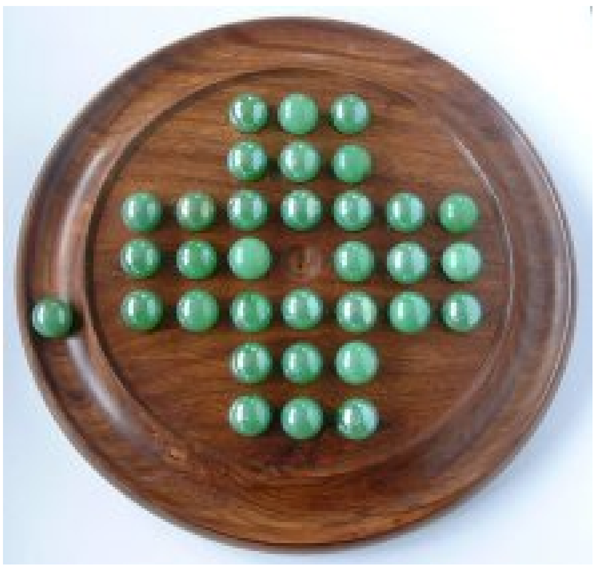
Figure 2.4: The board, tokens and starting position of a Solitaire game

In the illustration, the tokens are instantiated by marbles. The starting position involves every position on the board being occupied by a token with the exception of the centre position which is empty. The only allowable move is to 'jump' a marble over another into the empty space. The effect of this move is to cause the jumpee to be removed from the board. This description satisfies Haugland's three essential parameters. The goal is to make jump moves until a single token remains occupying the center position. A further three points are raised by Haugland in relation to Solitaire as a formal system. These points equate to characteristics that can be identified for a given system. Firstly, the system is *self-contained* and relies only on its own tokens, positions and moves. Secondly, the system is *perfectly definite* meaning that there are no ambiguities with regards to determining what the current position is or whether a move is legal. Lastly, for every position there are only a finite number of checks that must be made to determine whether a particular move played at that point is legal, such a system is therefore *finitely checkable*. Sets of rules developed to computationally model a phenomena should strive to exhibit Haugland's characteristics because they suggest a computationally tractable approach. The first characteristic is not critical, the largest impact of the self-containment characteristic appears to be that a game that relies on external components would be more fragile than a game that is completely self-contained and would break, or could not be played, if the external components were not available. However, if a game is not perfectly definite then either it cannot be determined whether a move is legal or illegal, or the current state of the game cannot be completely known. If the legality of a move is indeterminate and cannot be established then the rules are probably too loosely defined to allow a computational implementation. If an implementation was achieved in these circumstances then it would likely be a single interpretation from the set of interpretations of the ambiguous rules. In this case the original game that was not perfectly definite is better described as a meta-game that describes a family of related lower level instances of games. Finally, a game that is not finitely checkable may be unplayable. If the converse position is taken that a game whose rules are not finitely checkable is a game whose rules are uncheckable then the game is not tractable and cannot be implemented without altering the rules to make them require a finite number of checks. It is also desirable, but not necessary, that as well as being finite the number of checks is as small, non-time consuming, and non-computationally intensive as possible so as to minimise the gap between moves that is dependent upon legality checking.

Interestingly Haugeland barely mentions either the end-point or permissible outcomes of games in his definition and exploration of games as formal systems other than to state that they sometimes exist. This is due to the difference in approach and concerns between the investigation of games in the *formal context* of games as formal systems, and the *ludic context* of games as goal-oriented, interactive pastimes. Suits [123, pp. 34] defines games as follows;

> "To play a game is to attempt to achieve a specific state of affairs, using only means permitted by rules, where the rules prohibit use of more efficient in favour of less efficient means, and where the rules are accepted just because they make possible such an activity."

Suits investigates what are termed *prelusory goals*, introducing them in the context of the "state of affairs" from the game definition that the participants aim to bring about through their play. For Suits therefore, and in the wider ludic context, games are about bringing about specific states and particular outcomes as opposed to the formal context which is more concerned with the space of states that can be brought about. Suits also investigates the relationship between rules and ends [123, pp. 24–26], where ends are conceived in the sense of a particular outcome that the participants are trying to attain by playing, but Suits' investigation is in the context of rule-adherence and rule-breaking rather than as an examination of end-points and associated outcomes. With regards to rule-breaking, rules and ends are inseparable because a player who breaks the rules of a game renders the attainment of the end impossible within the game. This is because a person who steps outside the rules (cheats) in order to attain the end may walk away with the pot or a trophy, and in a technical sense has won, they have certainly not won the game in the sense that one cannot win a game without playing it, and to play a game requires the players to obey the rules.

Possibly, the reason for the lack of attention paid to terminating games may stem from the fact that a termination state is not a mandatory parameter for a game to be played unlike the other three parameters which are necessary conditions for play. As such a game may conceivably omit termination rules, as happens often in the kinds of games that are described as pastimes, where the object is the playing rather than achieving a particular outcome. Defining an end-point, in terms of goal states, and permissible outcomes, in terms of winning positions with respect to each participants, is an essential practicality when computational implementations are considered. End-points enable the participants to determine at exactly which point the game should terminate and identify those states should they arise. It is also necessary to consider the range of outcomes that can occur so that each participant can determine their own status with respect to the outcome of the game and each other.

It is interesting that whilst researchers in domains such as AI and CS have put a lot of effort into formalising various systems in terms of games, interpreting real-world games in terms of formal systems has, until very recently, received little attention from game designers. Studies of popular games and pastimes have been conducted, the most comprehensive survey being that of Avedon and Sutton-Smith [9], but it has been only recently that researchers into popular games have looked to formalise the rules that pertain in such situations. Research into popular games has been spurred by a rise in interest into formalising the rules used in computer games as well as more traditional pastimes [111, 112], and an attempt to build an academic discipline that studies the rules of such games as instantiations of ludic systems [43]. This interest has lead to attempts to build formal systems that can represent the rules of popular games and formal methods that support game development.

The motivation for formal approaches such as that of Grünvogel [43] is the recognition that all common accepted definitions of games, a survey of which can be found in Salen and Zimmerman [111, pp. 73–81], define games as systems but do not further define the term system. This is similar to the approach taken in dialectical game research that models argumentative dialogue as a (ludic) game but does not further explore the nature of such games. Grünvogel's approach sees games as a set of state changing objects whose states are rule-governed and influenced by players and other objects. The formalism is based upon the abstract control systems of Tabuada *et al* [128] in which a game is a triple $< \mathcal{S}, \mathcal{M}, \mathcal{F} >$ where $\mathcal{S}$ represents the state space of game objects, $\mathcal{M}$ the input actions, and $\mathcal{F}$ maps members of $\mathcal{M}$ onto $\mathcal{S}$. The value of this approach, at least in terms of describing the rules of games, is that a formalism is more precise than natural language, enabling a game designer to precisely define the concepts used in a design.

2.3 Dialogue & Games

There is a long history of relationships between games and dialogue that can be divided into three periods, the ancient period which includes the games played in Ancient Greece at least as far back as Aristotle, the medieval period which includes the uses of games of disputation as pedagogic tools, and the modern period which includes the games developed to model various aspects of argumentative dialogue. During the ancient period, the Socratic *Elenchus* (refutations) was a structured debate that occurred in two senses according to Robinson [109, ch. 2]. In the wider sense elenchus involves examining a person with regard to an initial statement, by uttering questions calling for further statements with the aim of determining the meaning of, and verifying the truth of, the initial statement. A narrower sense depicts elenchus as a form of cross-examination or refutation where the aim is to establish the falsity of the initial statement. Although primarily a tool for education and enquiry, Plato recognised that the elenchus were considered entertaining by bystanders and that games of elenchus would be played that imitated the dialogues of Socrates [109, ch. 2]. Aristotle's depiction of elenchus describes a dialectical game, governed by formal rules, where the questioner attempts to prove a thesis by extracting it from the concessions made by the adversary during a question-reply contest [120].

The medieval period, particularly the thirteenth and fourteenth centuries, saw the continued use of games of disputation as pedagogical tools known by the name *Obligationes* [29] and probably based directly upon elenchus. The general form of a game of Obligationes involves two parties, Opponent and Respondent. The game begins when the Opponent puts forward a proposition (*positum*) that the Respondent must accept as true for the sake of the game. The Opponent then makes more propositions, extended one at a time, that the Respondent must concede or deny on the basis of previous concessions. Thus the Respondent loses if a contradictory set of propositions is conceded and wins if time-limit of the game is reached whilst consistency is maintained. Hamblin provides a formulation for what is referred to as 'The Obligation Game' [44, pp. 260–262] and which is a variant of Obligationes. In the Obligation Game there is a questioner role and an answerer role, and the questioner personally makes no concessions but aims instead to trap the answering party into conceding a contradiction within a specified timelimit or number of moves. If the questioner managed to do this then the questioner won, otherwise the answerer won. Another formulation of rules for playing a game called *ars obligatoria* is due to Angelelli [3] and is based upon the *Logica Magna* and *Logica Parva* of Paulus Venetus. A good overview of medieval dialogue games is due to Spade [121].

The modern period begins with three semi-independent strands which include the language games of Hintikka [46], although the focus of Hintikka's games is more semantical than dialogical [30, pp. 246], the *Dialogische Logik* and dialectical games of Lorenzen which apply economic game-theoretic concepts to the semantics of dialogue [30, ch. 9], which lead to Barth and Krabbe's theory of argumentation called *formal dialectics* [12], and Hamblin's *formal dialectical games* [44]. The main difference between Hamblin's approach and the Lorenzen and Hintikka approaches is that Hamblin's games do not yield a concept of logical validity [30, pp. 247]. In this sense, the Lorenzen and the Hintikka games can be considered to be systems of dialogue logic whilst Hamblin's games are not. Instead Hamblin's games have closer ties to the ludic approach to games found in the medieval and ancient dialogue games.

2.4 Formal Dialectical Games

The formal dialectical game is an approach to structuring dialogical interaction whose origins lie in the Obligation Game. Dialectical games therefore are ludic games as opposed to economic game-theoretic games. The impetus to develop these modern versions of the Obligation Game is commonly ascribed to Hamblin [44, ch. 8] but Rescher independently developed a system of formal disputation which has proved influential in some circles [108]. Thus extant dialectical games now find their roots usually in either Hamblin's or Rescher's game but are sometimes based to some degree in both. A modern understanding of the term dialectic equates it with the practise and theory of conversations [60]. It is in this context that formal dialectical games are developed and deployed, as a set of rules specifying a ludic game that governs the interactions between participants of a dialogue. Formal dialectical games can be simply defined as two-player, turn-taking games in which the moves available to the players represent the locutional acts and utterances made by the participants in the dialogue.

Diagram 2.5 illustrates the development of dialectical games over time beginning with Aristotle and leading through the development of the Obligation Game in the medieval period, to the philosophical applications in fallacy theory, and eventually to the contemporary applications of dialectial games in multiagent systems. Although many dialectical games have been formulated during these periods, this thesis seeks at this point to investigate a representative sample of those games rather than to perform an exhuastive analysis of the space of extant games. The particular games selected were due either to their primacy in the literature, e.g. Hamblin's 'H' and Mackenzie's 'DC', or because they introduce new types of rule or game component that are not found in other games, e.g. Walton's 'CB' family of games which introduce notions of win-loss, Girle's 'DL' family of games which incorporate both new commitment store artifacts and a wide range of locutionary acts, Walton and Krabbe's 'PPD' and 'CPD' games which introduce the notion of dialectical shifts between dialogues of different types, and Bench-Capon's 'TDG' which was one of the first games to be oriented towards a computational implementation.

Hamblin's original aim, in suggesting dialectical games, was to provide a systematic formal account of how Aristotle's fallacies outside of language occur in dialogue, reaffirming the dialectical context within which many fallacies occur. The motivation for this was spurred by dissatisfaction with the way that the fallacies were dealt with in the logical textbooks, an approach that Hamblin described as the "*standard treatment*" [44, ch. 1]. Hamblin identifies the standard treatment as merely cataloguing a disparate set of fallacies and delivering that set with a number of tired and cliched examples as the whole of fallacy research. Hamblin aimed to motivate a new approach to fallacies as an antidote to this standard treatment and he began by examining Aristotle's fallacies outside of language within a strictly dialectical context. To achieve this examination of the fallacies Hamblin proposes an extension of formal logic to take into account the dialectical context in which fallacious arguments are presented.

Formal dialectical games were introduced within the more general framework of dialectical systems, regulated dialogues or families of dialogues that have a number of participants who speak in accordance with a set of rules or conventions. The rules specify what can be said given previous events, the overall context of the dialogue and the specific context of the current utterance. Hamblin identifies two distinct approaches to dialectical games which take opposing yet complementary approaches, descriptive dialectic and formal dialectic. Descriptive dialectic starts with examples of real dialogues and attempts to identify the underlying rules and conventions which can be used to describe and explain the features of those dialogues. Conversely, formal dialectic starts with a set of rules and conventions and aims to produce dialogue through the application of the rules. The aim of producing dialogue by applying formal dialectic rules is to explain how phenomena that can be identified in real-world dialogue might actually occur. Hamblin sees most value in a synergy of the two approaches, descriptive dialectic taking real-world dialogue and identifying formalisable features of that dialogue, and formal dialectic then being used to explain those features. In [44], Hamblin deals primarily with formal dialectical games because the fallacies that he was examining had been dealt with mostly from a non-dialectical, descriptive context that enumerated the most common features of arguments that exhibited the various fallacies.

One of the most important contributions made by Hamblin to the study of dialectical games, in light of the historical games, is the explicit representation of the participants commitments in the commitment store. Hamblin explains that a speaker becomes committed to a statement when they either personally utter the statement or agree to the statement when it is uttered by another dialogue participant. Commitments are distinguished from beliefs because they are a function of the locutional events that have occurred during the dialogue although Hamblin allows that the resultant collection of commitments may be considered to represent a *persona* of beliefs. Belief and commitment are distinguished based on the idea that a speaker does not always believe everything they say but that their saying it still commits them whether they believe it or not. Retraction of commitments is also distinguished from denying the statement associated with the commitment because retraction removes the burden of that commitment from the retracting player whilst a denial entails incurring a contrary commitment. Commitment stores are introduced as a means for the players to keep a tally of any commitments that they have incurred during a dialogue. The players each maintain their own commitment store to which commitments are added or removed dependent upon the events that occur in the dialogue. Commitment stores are also publicly inspectable so that all players can inspect each others stores at any point. The commitment store is a flexible mechanism for specifying certain requirements over the gross structure of the game. For example, if it is required that a player must be consistent in their commitments then the rules of the game can be formulated to ensure that each new commitment incurred can be added without inconsistency to that player's commitment store. The commitment store is the main distinguishing factors of the Hamblin-type formal dialectic game and is the most common factor in the extant games that is used to determine whether a given locutional act is permissible at any given point in the dialogue.

2.4.1 Hamblin

Hamblin's simple dialectical system, named 'H' in [144], is a specific instantiation of a dialectical game that is constrained to two players who take turns to utter a single locution in accordance with the rules. To specify the gross structure of a dialectic system, Hamblin defines three sets of rules; locutional rules, syntactical rules, and commitment-store operations. Locutional rules specify the range of locutions that are permitted by the system. Syntactical rules are those rules which set out structural aspects of the system and do depend on commitment store operations. The rules in the set of commitment store operations define the operations that must be performed on the players commitment stores as a result of the utterance of particular locutions. All rules that pertain to H fit into one of these categories. The original formulation of rules for H are as follows;

Hamblins Game 'H'

Locutional Rules :

- **'Statement S'** or, in certain special cases, 'Statements S, T'.
- **'No commitment S, T, ..., X'** for any number of statements S, T, ..., X (one or more).
- **'Question S, T, ..., X?'** for any number of statements (one or more).
- **'Why S?'** for any statement S other than a substitution-instance of an axiom.
- **'Resolve S'**

Syntactical Rules :

- **S1** Each speaker contributes one locution at a time, except that a 'No commitment' locution may accompany a 'Why' one.
- **S2** 'Question S, T, ..., X?' must be followed by
  - **(a)** 'Statement¬(S∨T∨...∨X)'
  - **or (b)** 'No commitment S∨T∨...∨X'
  - **or (c)** 'Statement S' or
    'Statement T' or
    ———— or
    'Statement X'
  - **or (d)** 'No commitment S, T, ..., X.'
- **S3** 'Why S?' must be followed by
  - **(a)** 'Statement ¬S'

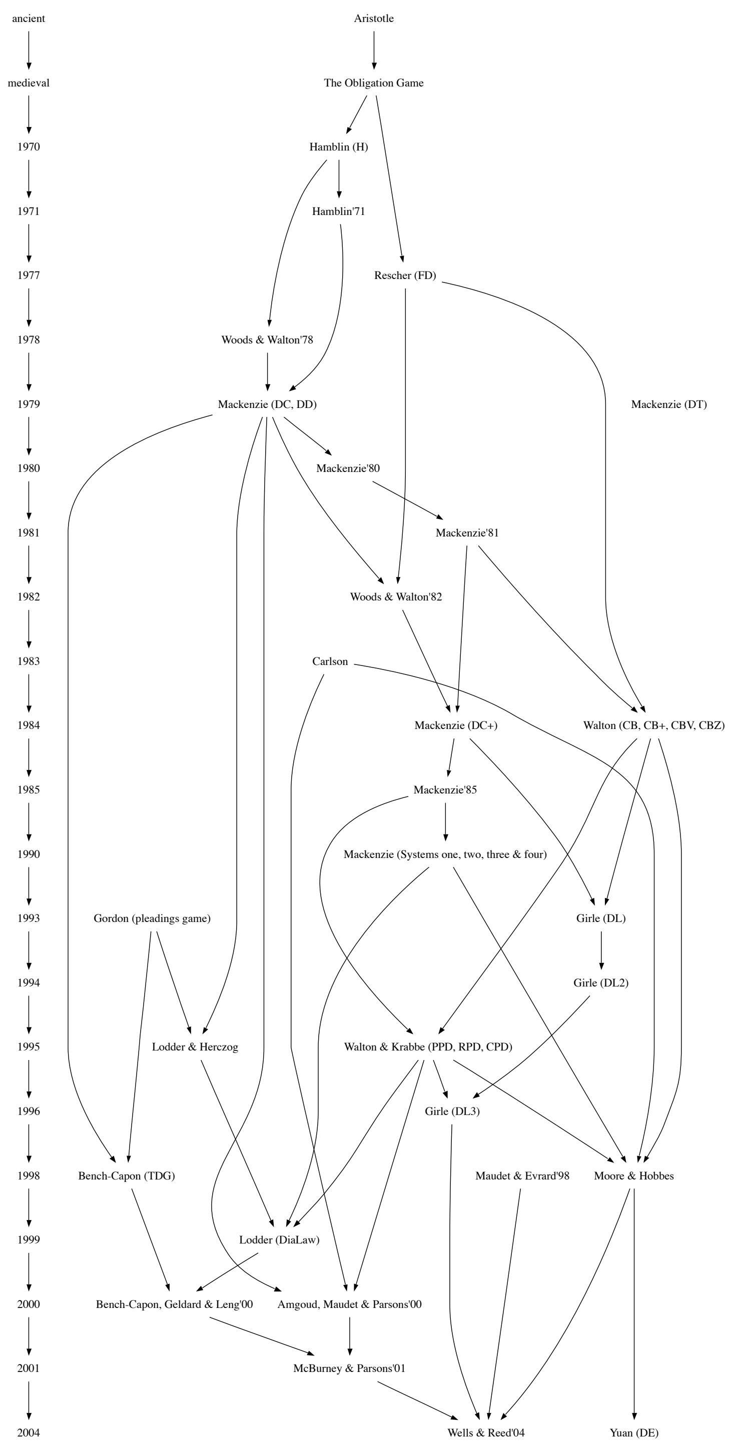

Figure 2.5: A Timeline of some Formal Dialectical Games.

or (b) 'No commitment S'

or (c) 'Statement T' where T is equivalent to S by primitive definition

or (d) 'Statements T, T⊃S' for any T

S4 'Statements S, T' may not be used except as in 3(d)

S5 'Resolve S' must be followed by

(a) 'No commitment S'

or (b) 'No commitment ¬S'

Commitment-Store Operations :

C1 'Statement S' places S in the speaker's commitment store except when it is already there, and in the hearer's commitment store unless his next locution states ¬S or indicates 'No commitment ' to S (with or without other statements); or, if the hearer's next locution is 'Why S?', insertion of S in the hearer's store is suspended but will take place as soon as the hearer explicitly or tacitly accepts the proffered reasons (see below).

C2 'Statements S, T' places both S and T in the speaker's and hearer's commitment stores under the same conditions as in C1.

C3 'No commitment S, T, ..., X' deletes from the speaker's commitment store any of S, T, ..., X that are in it and are not axioms.

C4 'Question S, T, ..., X?' places the statement S∨T∨...∨X in the speaker's store unless it is already there, and in the hearer's store unless he replies with 'Statement¬(S∨T∨...∨X)' or 'No commitment S∨T∨...∨X'

C5 'Why S?' places S in the hearer's commitment store unless it is there already or he replies 'Statement ¬S' or 'No commitment S'.

Woods and Walton in [144] use H to provide a theoretical underpinning for an examination of the fallacy of *petitio principii*. In [143] Woods and Walton examine *petitio* within the framework of doxastic-epistemic logic but in [144] they approach it from within a dialectical framework to determine whether *petitio* can be adequately characterised using dialectical systems. This is to answer a point raised by Barker in [11] which suggests that *petitio* presupposes a context of disputation. Woods and Walton's approach applies H, with additional rules A6 and A7, to a number of canonical examples of circular reasoning which are illustrated with fragments of dialogue. To determine whether H+A6+A7 is adequate to prohibit all circular reasoning Woods and Walton compare H with Kripke's intuitionistic semantics. A difference that they identify is in the *cumulativeness* of the system. Cumulatively in this context is in regard to values that the uttered propositions are claimed to possess. A system is defined as cumulative with respect to a given element if once that element is established it remains so. For example a system which is cumulative with respect to a players commitments requires that once a player incurs a commitment they can no longer retract that commitment. The system that Woods and Walton examine, Hamblin's system H plus the supplemental rules A6 and A7, is identified as non-cumulative and not prohibitive of circular dialogues. Another game, Mackenzie's DC for example is described as non-cumulative in the Woods and Walton sense but prohibits *petitio*. A Kripke tree is cumulative whereas H is essentially non-cumulative. A system is described as *Cumulative* if once something is established it remains established. For example, a version of H that did not allow retraction would be cumulative with respect to commitments because there would be no way to remove those commitments. Once a commitment was established it would remain so established for the remainder of the dialogue.

Hamblin provides additional and alternative rules for H, which include those rules explored by Woods and Walton. They are presented here thus;

The Additional Rules of H

Additional :

A1 Question S, T, , X? may occur only whenS∨T∨ ∨X is already a commitment of both speaker and hearer.

A2 Statement S may not occur when S is a commitment of the hearer.

A3 Question S, T, , X? may not occur when any of S, T, , X is a commitment of the speaker.

A4 Statement S may not occur when S is already a commitment of both speaker and hearer.

A5 Question S, T, , X? may not occur when any one or more of S, T, , X, is a commitment of the speaker and any one or more a commitment of the hearer.

A6 Why S? may not be used unless S is a commitment of the hearer and not of the speaker.

A7 The answer to Why S?, if it is not Statement ¬S or No commitment S, must be in terms of statements that are already commitments of both speaker and hearer.

A8a If there are commitments S, T, , X of one participant that are not those of the other, the second will, on any occasion on which he is not under compulsion to give some other locution, give Why S? or Why T? or or Why X?

A8b If one participant has uttered Statement S or Statements S, T or Statements T, S and remains committed to S, and the other has been and remains uncommitted to it, the second will (on any occasion on which he is not under compulsion to give some other locution, give Why S? or Why T? or or Why X?)

A9 When S is written into a commitment store it is written Sc if the other participants commitment store already contains S or Sc; otherwise it is written S.

2.4.2 Mackenzie

Mackenzie's system DC [70] responds to the Woods and Walton exploration of *petitio principii* [144] in which cumulativity is identified as the crucial factor for the occurrence of the *petitio* fallacy in a dialectical context.

Mackenzie's Game DC

Locutions :

(i) Statements. 'P', 'Q', etc. and truth-functional compounds of statements: 'Not P', 'If P then Q', 'Both P and Q'.

(ii) Withdrawals. The withdrawal of the statement 'P' is 'No commitment P'.

(iii) Questions. The question of the statement 'P' is 'Is it the case that P?'

(iv) Challenges. The challenge of the statement 'P' is 'Why is it to be supposed that P?' (or briefly 'Why P?').

(v) Resolution Demands. The resolution demand of the statement 'P' is 'Resolve whether P'.

Commitment Rules :

Statements, CR$_S$: After a statement 'P', unless the preceding event was a challenge, 'P' is included in both participants' commitments.

Defences, CR$_{YS}$: After a statement 'P', when the preceding event was 'Why Q?', both 'P' and 'If P then Q' are included in both participants' commitments.

Withdrawals, CR$_W$: After the withdrawal of 'P', the statement 'P' is not included in the speaker's commitment. The hearer's commitment is unchanged.

Challenges, CR$_W$: After the challenge of 'P', the statement 'P' is included in the hearer's commitment; the statement 'P' is not included in the speaker's commitment; and the challenge 'Why P?' is included in the speaker's commitment.

Questions and Resolution demands, CR$_Q$ and CR$_R$: These locutions do not themselves affect commitment.

Initial Commitment, CR$_0$: The initial commitment of each participant is null.

Rules Of Dialogue :

R$_{Form}$: Each participant contributes a locution at a time, in turn; and each locution must be either a statement, or the withdrawal, question, challenge or resolution demand of a statement.

R$_{Repstat}$: No statement may occur if it is a commitment of both speaker and hearer at that stage.

R$_{Imcon}$: A conditional whose consequent is an immediate consequence of its antecedent must not be withdrawn.

R$_{Quest}$: After 'Is it the case that P?', the next event must be either 'P', 'Not P' or 'No commitment P'.

R$_{LogCall}$: A conditional whose consequent is an immediate consequence of its antecedent must not be challenged.

R$_{Chall}$: After 'Why P?', the next event must be either;

1. 'No commitment P'; or
2. The resolution demand of an immediate consequence conditional whose consequent is 'P' and whose antecedent is a conjunction of statements to which the challenger is committed; or
3. A statement not under challenge with respect to its speaker (i.e., a statement to whose challenge its hearer is not committed).

R$_{Resolve}$: The resolution demand of 'P' can occur only if either;

1. 'P' is a conjunction of statements which are immediately inconsistent and to all of which its hearer is committed; or
2. 'P' is of the form 'If Q then R', and 'Q' is a conjunction of statements to all of which its hearer is committed; and 'R' is an immediate consequence of 'Q'; and the previous event was either 'No commitment R' or 'Why R?'.

R$_{Resolution}$: After 'Resolve whether P', the next event must be either;

1. The withdrawal of one of the conjuncts of 'P'; or
2. The withdrawal of one of the conjuncts of the antecedent of 'P'; or
3. The consequent of 'P'.

An additional system named DD is introduced in [70]. DD is constructed from DC by replacing the CRS rule with an new rule CR$_{SS}$. Whilst DC is described as prohibiting *petitio* and non-cumulative, the modified system named DD is described as "non-cumulative in an even stronger sense"[70].

Rules of DD

as for DC with the following amendment:

CR$_{SS}$: After a statement P unless the preceding event was a challenge, P is included in both participants commitments; and Why P? is not included in the speakers commitment

---

2.4.3 Walton

Walton introduces a number of games in [133] beginning with the game CB which can be described as a simple sub-game of H and DC that incorporates win/loss notions inspired by Rescher's game of disputation. Walton proposes CB as a minimal system for studying strategic play in dialectical games. This is examined through the notion of a win-strategy, a dialectical path though the dialogue terminating at a player's thesis. CB forms the basis for a group of systems through the application of a number of additional rules. These systems are focussed on the process of articulating and clarifying the players commitments through the question-answer structures found in argumentative dialogue. Walton arranges the systems in terms of strength where strength is defined in terms of the strictness with which commitments are handled. CB is thus identified as the weakest system in the CB family of games in terms of how commitments are handled, whilst CBZ is identified as the strongest.

Rules of CB

Locution Rules :

- **Statements** Statement-letters, S, T, U, ..., are permissible locutions, and truth-functional compounds of statement-letters.
- **Withdrawals** 'No commitment S' is the locution for withdrawal (retraction) of a statement.
- **Questions** The question 'S?' asks 'Is it the case that S is true?'
- **Challenges** The challenge 'Why S?' requests some statement that can serve as a basis in proof for S.

Commitment Rules :

- **(i)** After a player makes a statement, S, it is included in his commitment-store.
- **(ii)** After the withdrawal of S, the statement S is deleted from the speaker's commitment-store.
- **(iii)** 'Why S?' places S in the hearer's commitment-store unless it is already there or unless the hearer immediately retracts his commitment to S.
- **(iv)** Every statement that is shown by the speaker to be an immediate consequence of statements that are commitments of the hearer then becomes a commitment of the hearer's and is included in his commitment-store.
- **(v)** No commitment may be withdrawn by the hearer that is shown by the speaker to be an immediate consequence of statements that are previous commitments of the hearer.

Dialogue Rules :

- **(R1)** Each speaker takes his turn to move by advancing one locution at each turn. A no-commitment locution, however, may accompany a why-locution as one turn.
- **(R2)** A question 'S?' must be followed by (i) a statement 'S', (ii) a statement 'Not S', or (iii) 'No commitment S'.
- **(R3)** 'Why S?' must be followed by (i) 'No commitment S' or (ii) some statement 'T', where S is a consequence of T.

Strategic Rules :

- **(i)** Both players agree in advance that the game will terminate after some finite number of moves.
- **(ii)** For every statement S accepted by him as a commitment, a player is awarded one point.
- **(iii)** The first player to show that his own thesis is an immediate consequence of a set of commitments of the other player wins the game.
- **(iv)** If nobody wins as in (iii) by the agreed termination point, the player with the greatest number of points wins, or the game is a draw.

The system CB+ [133] incorporates the rules of CB with the addition of the (+) rule. This rule is formulated to enable the system to regulate inconsistency in the players commitments, as follows;

Rules of CB+

as for CB plus the following:

(+) A player loses all the points he has previously accumulated for incurring commitments if (i) he has previously indicated 'No commitment S' for some statement S, but is shown by his opponent to be an immediate consequence of some of his commitments, or if (ii) he is committed to some statement S but then moves 'No commitment T' where T is an immediate consequent of S.

The CBV system [133], where v stands for veiled, introduces the *veiled* or *dark-side commitment*. Each player possesses a dark-side commitment store which is private. The contents of the dark-side commitment store are not on public view unlike the conventional commitment store proposed by Hamblin. The rules of CBV set out the circumstances under which a commitment is moved from the dark-side to the light-side so that the contents of the dark-side store only become known to the participants during play.

Rules of CBV

as for CB with the following amendments:

Commitment Rule (*Additional*) :

(vi) If a player states 'No commitment S' and S is on the dark side of his commitment-store, then S is immediately transferred into the light side of his commitment-store.

Strategic Rules (*Replacement*) :

(i) Both players agree in advance that the game will terminate after some finite number of moves.

(ii) The first player to show that his own thesis is an immediate consequence of a set of commitments of the other player wins the game.

(iii) If nobody wins as in (ii) by the agreed termination point the game is declared a draw.

Finally CBZ is described by Walton as the strictest system allowed within Walton's dialectical game framework [133]. The notion of strictness in this sense means how the rules restrict the participants from withdrawing commitment at any given point during a game. A stricter game is one in which participants are more restricted from retracting commitment by rules which place more barriers in the way of retraction.

Rules of CBZ

Locution Rules :

(i) *Statements:* Statement-letters, S, T, U, ..., are permissible locutions, and truth-functional compounds of statement-letters.

(ii) *Withdrawals:* 'No commitment S' is the locution for withdrawal (retraction) of a statement.

(iii) *Questions:* The question 'S?' asks the hearer whether or not he wants to reply that S is true.

(iv) *Challenges:* The challenge 'Why S?' requests some statement that can serve as a basis of proof for S.

(v) *resolutions:* The resolution 'S, ¬S?' requests the hearer to select exactly one of the pair S, ¬S.

Dialogue Rules :

(i) Each speaker takes his turn to move by advancing exactly one locution at each move.

(ii) A question 'S?' must be followed by (i) a statement 's', (ii) a statement '¬S', or (iii) 'No commitment S'.

(iii) 'Why S?' must be followed by (i) 'No commitment S' or (ii) some statement 'T'.

(iv) For a speaker to legally pose a resolution-request 'S, ¬S', the hearer must be committed to at least one of the pair S, ¬S.

(v) A (legal) resolution-request must be followed by a statement 'S' or a statement '¬S'.

(vi) If a statement S and also its negation ¬S become included in the light side of a player's commitment-store, the opposing player must pose a resolution request 'S, ¬S' at his next free move.

(vii) If a speaker states 'No commitment S' but S is in his light-side commitment store, the hearer must pose a resolution request 'S, ¬S' at his next move.

Commitment Rules :

(i) After a player makes a statement, S, it is included in his commitment-store.

(ii) After the withdrawal of S, the statement S is deleted from the speaker's commitment-store.

(iii) 'Why S?' places S in the hearer's commitment store unless it is already there or unless the hearer immediately retracts his commitment to S.

(iv) Every statement that is shown by the speaker to be an immediate consequence of statements that are commitments of the hearer then becomes a commitment of the hearer's and is included in his commitment-store.

(v) No commitment that is shown to be an immediate consequence of statements that are commitments of the hearer may be withdrawn by the hearer, unless the speaker agrees.

(vi) If a player states 'No commitment S' and S is included in the dark side of his commitment-store, then S is immediately transferred into the light side of that player's commitment-store.

(vii) Whenever a statement S goes into the light side of a player's commitment store, if its negation ¬ is on the dark side of that player's store, it must immediately be transferred to the light-side. Similarly, S must go from the dark (if it is there) to the light side as soon as ¬S appears on the light side.

(viii) No commitment may be added to or deleted from a player's store except by one of the above six commitment rules.

Strategic Rules :

(i) Any player who makes a move other than those permitted by the six dialogue rules immediately loses the game.

(ii) The first player to show that his own thesis is an immediate consequence of a set of light side commitments of the other player wins the game.

(iii) Both players agree in advance that the game will terminate after some finite number of moves.

(iv) If nobody wins as in (ii) by the point agreed on in (iii), the game is a draw. Or if it becomes evident to all the players that the dark sides of their commitment-stores are empty, the game may be ended by universal consent. In the latter case, the players may agree to maintain their light side commitment-stores and begin with a new set of dark side commitment-stores.

2.4.4 Girle

In [39, 40, 41], Girle introduces four similar dialectical games. In setting out the first game, DL [39], Girle draws parallels between issues in belief set expansion and contraction, and epistemic entrenchment, and the way commitment is incurred and retracted, and how rules can be formulated to govern the retraction of a series of connected commitments. DL2 [40] sets out a method for building, justifying and revising an agents beliefs. Finally Girle looks at the need for commands and instructions in dialogue, and how locutions to govern the use of such imperatives can be used to extend an existing dialectical game. The DL3 [41] game is introduced as a general model of dialogue which is then extended, through the addition of instructional locutions and the required rules to govern their use, to yield the IDL3 [41] game.

In [40], Girle proposes the DL2 game as a basis for a belief system that facilitates the *"everyday belief"* of agents whose beliefs are possibly disconnected, disorganised and inconsistent. This is to avoid problems associated with the logically and deductively omniscient ideal agent whose beliefs are commonly perceived as well-ordered, well-organised and logically consistent. Girle proposes that agents build their belief sets through construction of a commitment store and sets out the rules of DL3 to facilitate this.

Girle identifies dialogue logics as a sub-species of joint activity system where the dialogue logic sets out the necessary conditions for dialogue to be a "joint activity of rational debate" [41]. DL3 is introduced as a baseline game which can be used to explore issues in such joint activity systems.

Girle identifies in [41] that contemporary dialogue logics omit provision for commands and instructions and that such a provision is required if dialogue based systems are to be used in computer assisted learning situations where all interactions and hence control of the system are through verbal interchanges. Girle identifies that such utterances as "tell me more", or "would you repeat that" are instructional in nature and play an important role in the dialogues in which they occur. Likewise sequences of exchanges can occur in dialogue where an instruction is given and the instructee enquires of the instructor how to go about satisfying the instruction. Girle seeks to extend the kinds of dialogue that dialectical games can be applied to and fine-tunes the DL3 system to apply to such situations where instructional dialogue occurs. The DL3 system is thus extended with a number of new rules to yield the IDL3 (Instructional Dialogue Logic 3) system as a way to make provision for the use of commands and instructions in dialogue. IDL3 incorporates new locution types such as, instructions, purpose declarations, method questions and inputs to account for the identified needs of instructional dialogues.

Rules of DL3

Locutions :

Categorical Statements The *categorical statements* are statements such as *P*, *not P*, *P and Q*, *P or Q*, *If P then Q* and *statements of ignorance (I do not know whether or not P)*. The last is abbreviated to τP.

Reactive Statements The *reactive statements* are *grounds (Because P)*, abbreviates to $\because P$.

Logical Statements The *logical statements* are *immediate consequence conditionals* such as: If *P* and *P implies Q* then *Q*.

Term Declarations A *term declaration* is the utterance of some *term*, say *t*.

Withdrawals The *withdrawal* of *P* is of the form *I withdraw P*, *I do not accept P*, *not P*, or *I no longer know whether P*. The first and second are abbreviated as *–P*.

tf-questions The *tf-questions* are of the form *Is it the case that P?*, abbreviated to P?.

wh-questions The *wh-questions* are of the form *What (when, where, who, which) is an (the) F?*. The strict logical form is *(Qx)Fx*, where Q is the interrogative quantifier, and for each such formula there will be an associated statement $(\exists x)Fx$.

Challenges A *challenge* is of the form *Why is it supposed to be that P?*, abbreviated to *Why P?*.

Resolution Demands The *resolution demands* are of the form *Resolve P*.

Commitment Store Rules :

C1 *Statements*: After an event $\langle$ *n*, *S*, *P* $\rangle$, where *P* is a statement, unless the preceding event was a challenge, *P* goes into the commitment stores of both participants.

C2 *Defenses*: After the event $\langle$ *n*, *S*, $\because P$ $\rangle$, when: *Why Q?* and *Q* are in the speaker's commitment store, the *justification sequence*: $\langle$ *P*, *If P then Q*, *Q*, *Why Q?* $\rangle$, and *P* and *If P then Q* go into the commitment stores of both participants. The challenge: *Why Q?* is removed from the commitment stores of both participants.

C3 *Withdrawals*: After any of the following three events:

$\langle$ ***n*, *S*, *-P*** $\rangle$ or

$\langle$ ***n*, *S*, *not P*** $\rangle$ or

$\langle$ ***n*, *S*, τ*P*** $\rangle$, where *P* is in *S*'s commitment store then

(a) the statement *P* is removed from the speaker's commitment store, and

(b) if the withdrawal was of form *not P* or τP, the the withdrawal goes into the commitment stores of both participants, and

(c) if *P* is a statement associated with a *wh*-question, say (*Qx*)*Fx*, then *P* is removed from the hearer's commitment store and (*Qx*)*Fx* is removed from the commitment stores of both participants, and

(d) if the withdrawal was preceded by the event $\langle$ *n-1*, *H*, *Why P?* $\rangle$, then *Why P?* is removed from the commitment stores of both participants, and

(e) if $\langle$ *P*, *If P then Q*, *Q*, *Why Q?* $\rangle$ is in the speaker's commitment store, then it is removed; and if *Q* is in either participants' store, then it is removed.

C4 *Challenges*: After the event $\langle$ *n*, *S*, *Why P?* $\rangle$, the challenge, *Why P?*, goes into the commitment stores of both participants. If *P* is not in the hearer's commitment store then: *P* goes into the hearer's commitment store. If *P* is in the speaker's commitment store, it is removed. If the *P* is present in the speaker's commitment store as part of a justification sequence, the justification sequence is removed.

C5 *Information*: After the event $\langle$ *n*, *S*, *(Qx)Fx* $\rangle$, the associated statement, ($\exists$x)Fx, goes into the commitment stores of all participants, and the *wh*-question, (*Qx*)*Fx* goes into the hearer's commitment store.

C6 *Reply*: After the declaring of a term in an event, say $\langle$ *n*, *S*, *t* $\rangle$, when the previous event was a *wh*-question, *(Qx)Fx*, then Ft goes into the commitment stores of both participants. Ft is known as the *wh*-answer to *(Qx)Fx*.

C7 *True/False*: After the event $\langle$ *n*, *S*, *P?* $\rangle$, if the statement *P* is in the speaker's commitment store it is removed.

Interaction Rules :

i *Repstat* No statement may occur if it is in the commitment stores of both participants.

ii *Imcon* A conditional whose consequent is an immediate consequence of its antecedent must not be withdrawn.

iii *LogChall* An immediate consequence conditional must not be withdrawn.

iv *TF-Quest* After $\langle$ *n*, *S*, *Is it the case that P?* $\rangle$, the next event must be $\langle$ *n+1*, *H*, *Q* $\rangle$, where Q is either:

- **(a)** a statement that *P*, or
- **(b)** a statement that *not P*, or
- **(c)** a withdrawal of *P*, or
- **(d)** a statement of ignorance (*I do not know whether or not P*).

v *Chall*: After $\langle$ *n*, *S*, *Why P?* $\rangle$ the next event must be $\langle$ *n+1*, *H*, *Q* $\rangle$, where *Q* is either

- **(a)** a withdrawal, or denial of *P*, or
- **(b)** the resolution demand of an immediate consequence conditional whose consequent is *P* and whose antecedent is a conjunction of the statements to which the challenger is committed, or
- **(c)** a statement of grounds *acceptable* to the challenger.

vi *Resolve*: The resolution demand in $\langle$ *n*, *Resolve whether P*, *P?* $\rangle$ can only occur if either

- **(a)** *P* is a statement or conjunction of statements which is immediately inconsistent and to which its hearer is committed, or
- **(b)** *P* is of the form *If Q then R* and Q is a conjunction of statements to all of which its hearer is committed, and *R* is an immediate consequence of *Q*, and the previous event was either $\langle$ *n-1*, *H*, *I withdraw P* $\rangle$ or $\langle$ *n*, *Resolve whether P*, *Why R?* $\rangle$.

vii *Resolution*: After the event $\langle$ *n*, *S*, *Resolve whether P* $\rangle$ the next event must be $\langle$ *n+1*, *H*, *Q* $\rangle$, where *Q* is either

- **(a)** the withdrawal of one of the conjuncts of *P*, or
- **(b)** the withdrawal of one of the conjuncts of the antecedent of *P*, or
- **(c)** a statement of the consequent of *P*.

viii *Enlightenment*: After the event, $\langle$ *n*, *S*, *(Qx)Fx* $\rangle$,

- **(a)** the next event must be $\langle$ *n+1*, *H*, *Q* $\rangle$, where *Q* is either
  - **(i)** the declaration of some *term*, say *t*, or
  - **(ii)** the withdrawal or denying of the associated statement *(∃x)Fx*, or
  - **(iii)** a statement of ignorance (*I do not know whether or not (∃x)Fx*).

  and
- **(b)** in the case of the hearer declaring some term, at the earliest subsequent event the asker of the wh-question must state the wh-answer to the wh-question, or its denial, where the earliest subsequent event is the event separated from the term declaration by nothing other than the asker asking further wh-questions and their wh-answers, where the wh-questions contain only predicates and terms from the wh-answers immediately preceding them.

2.4.5 Walton & Krabbe

In [138] a game named PPD$_0$ is introduced to model permissive persuasion type dialogues. Walton and Krabbe propose PPD$_0$ as just one example of a dialectic game to govern permissive persuasion dialogue and set out eighteen general attributes which can be used to characterise permissive persuasion dialogues. Their approach is to begin with a simple game to model such dialogues which can be extended through the inclusion, removal and replacement of rules to enable particular problems to be studied. In [138] Walton and Krabbe introduce a number of new features to formal dialectic research whilst simultaneously producing mature treatments of some existing concepts. The notions of dialogue stability and the idea of dialogue embeddings and transitions are introduced as new features of dialectical games. Dark-side commitments, introduced in [133], and their associated stores are given a central role in the model of permissive persuasion captured in PPD_0. The notion of commitment in general is analysed in depth and the ways of both incurring and retracting commitments, whether propositional or of other types are enumerated.

Walton and Krabbe state that how commitment should be incurred or retracted is dependent upon a number of general factors which include the kind of move made, the type of dialogue, the goal of the dialogue, the speaker's role, and the rules appropriate to the dialogue, however not all of these factors have been fully exploited in dialectical games. In this analysis Walton and Krabbe are beginning to move away from the purely technical approach to commitment as a device for maintaining relevancy as espoused by Hamblin towards an approach where commitment is a central consideration. Although not fully developed, the factors that Walton and Krabbe identify as pertinent to commitment begin the process of identifying a framework within which each dialectical game formulation can be situated. They identify a partial taxonomy of dialogue types which include persuasion, negotiation, inquiry, deliberation, information-seeking, eristic and mixed. Each type of dialogue may have further subtypes and can be classified based upon the initial situation, main goal and the participants individual goals. For each type of dialogue thus identified it is assumed that a dialectical game could be formulated to account for the interactions which occur during an instance of that dialogue. Furthermore, because social factors have a bearing on argumentative dialogue, for each dialogue type there may well be a whole family of dialectical games to account for each situation under which a dialogue of any particular type may occur.

Rules of PPD_0

Locution Rules :

1. Permitted locutions are of the following types:

   Statements :

   - ***Assertions*** *a*(P)
   - ***Concessions*** *c*(P)

   Elementary Arguments ΔsoP

   Questions :

   - ***Requests*** *con*(P)?
   - ***Extractors*** *serious*(P)?

Confronters *resolve*(P,Q)

Challenges P??

Retractions :

of *commitment* *nc*(P)

of *strong commitment* *na*(P)

2. Moves have the following structure:
$\langle$*nc*(P)/*na*(P), *c*(P), *resolve*(P,Q), *con*(P)?/*serious*(P)?, ΔsoP, P?? $\rangle$

At each slot in the six-tuple more than one element of the indicated type is allowed. Not every slot needs to be filled.

Besides moves of this structure, there are preparatory moves that precede the actual dialogue. In these preparatory moves, the locutions *a*(P) and *c*(P) are used. See commitment rules 2 below.

Commitment Rules :

1. Each participant (B,W) has three commitment stores. For participant X these are indicated as DX, AX_n, CX_n. (The index *n* refers to the stage of the discussion just completed.) DX, the set of X's *dark-side commitments*, remains fixed throughout the dialogue. Its elements are propositions. AX_n and CX_n contain formulas of L. These formulas are called *light-side commitments*. The elements of AX_n are called *assertions* of X and those of CX_n *concessions* of X. The formulas in CX_n that are not in AX_n are called *mere concessions* of X.
2. Preceding the actual dialogue there are some preparatory moves in which the contents of AW_0, CW_0, AB_0 and CB_0 are determined. To p0ut a formula into both AX_0 and CX_0, X uses the expression *a*(P). To put a formula merely into CX_0, X uses the expression *c*(P). The locutions *a*(P) are permitted in preparatory moves only.
3. The preparatory moves (which shall not be formulated in detail) are such that at least $AB_0 \neq \emptyset$, that $CW_0 \cap AB_0 = \emptyset$and that $CB_0 \cap AW_0 = \emptyset$. (Moreover, clearly $AB_0 \subseteq CB_0$ and $AW_0 \subseteq CW_-$)
4. **a.** A *retraction nc*(P) removes *P* from both the set of the speaker's assertions and the set of the speaker's concessions (if it is there in the first place). That is to say if X utters *nc*(P) at the *n*th stage then $P \notin AX_n$ and $P \notin CX_n$.

   b. A *retraction na*(P) removes *P* from the set of the speaker's assertions, but not from the set of the speaker's consessions.
5. *c*(P) enters *P* into the speaker's set of concessions, but not into the speaker's set of assertions.
6. ΔsoP enters all the elements of (explicit premises and warrant) of the argument ΔsoP into the set of assertions and into the set of concessions of the speaker.

Structural Rules :

1. The parties move alternately. W makes the first move in the actual dialogue.
2. Each party in its first move challenges or concedes each initial assertion presented by the other party.
3. For each move

   a. if the preceding move contained *resolve*(P,Q), the speaker must use either *nc*(P) or *nc*(Q);

   b. if the preceding move contained *con*(P)?, the speaker must either use *c*(P) or *nc*(P);

   c. if the preceding move contained *serious*(P)? the speaker utters either *c*(P) or *nc*(P), but if the proposition expressed by P is in the dark-side commitment store of the speaker, the speaker must utter *c*(P);

   d. if the preceding move contained ΔsoP the speaker must, for each element Q in the elementary argument ΔsoP that is not a concession of the speaker, and has not been challenged by the speaker thus far, either utter *c*(Q) or Q??;

   e. if the preceding move contained P??, the speaker must utter an argument ΔsoP, or utter *nc*(P) or *na*(P).
4. **a.** If the proposition expressed by P is in the dark-side commitment store of X and X has been confronted with an extractor *serious*(P)?, then the locution nc(P) is not available to X for the rest of the dialogue;

   b. *nc*(P) can be used only if either

1. the preceding move contained *con*(P)? or *serious*(P)?, or

2. P is a concession of the speaker;

c. *na*(P) can be used only if P is an assertion of the speaker.

5. A statement *c*(P) is allowed only if P is not a concession of the speaker and either P is an assertion of the other party or the preceding move contained *con*(P)? or *serious*(P)?
6. A request *resolve*(P,Q) may be used only if

   1. *P* and *Q* are explicit contradictories, and

   2. *P* and *Q* are both concessions of the listener.
7. A request *con*(P)? may only be used if

   1. one of the speaker's assertions is not among the concessions of the listener and

   2. *P* is not a concession of the listener.
8. The question *serious*(P)? may be used only if the preceding move contained *nc*(P) or *P*??
9. ΔsoP may be used only if *P* is not among the concessions of the listener and some earlier move contained *P*??
10. *P*?? can be used only if *P* is an unchallenged assertion of the listener and not a concession of the speaker (after completion of the move).
11. In each move the use of locutions of the forms *nc*(P), *na*(P), *c*(P), *P*?? is to be such that after completion of the move the following are true:

    1. For each elementary argument $\{P_1, ..., P_n\}$*so*C of the listener if $P_1, ..., P_n$ and $(P_1 \wedge ... \wedge P_n) \longrightarrow$ C are concessions of the speaker, then the conclusion is also a concession of the speaker.

    2. Each unchallenged assertion of the listener is a concession of the speaker.

    3. For each basic argument Δ of the speaker and for each Q that is an element of Δ, if Q is retracted by the speaker, then all the elements of Δ that are implicit or explicit immediate premises for Q must be retracted (as assertions at least); moreover, if Q itself functions in Δ as an immediate premise for P, all Q's copremises for P must be retracted. This procedure is recursive. (However, retraction of an element need not be put through if this element functions as a premise in some elementary argument of the speaker that is no part of a basic argument for Q or for a copremise of Q in Δ.)
12. A retracted initial thesis (assertion) cannot be reinstated as such.
13. Each move and each dialogue is limited with respect to the number of its symbol tokens. (We shall not here formulate the precise rules to this effect).

Win-Loss Rules :

1. At the end of the dialogue, if X has retracted its initial thesis P, X is the loser and the other party the winner with respect to P; however, if the other party has got P in its set of concessions by the end of the dialogue, X is the winner and the other party the loser with respect to P. In all other cases the dialogue is a draw with respect to P.

2.4.6 Bench-Capon

The Toulmin Dialogue Game (TDG) [13] is a Rescherian-type dialogue game having three participants, two players and a referee. The participants work cooperatively towards a position in which a claim is agreed and the players interactions have facilitated the construction of a supporting argument structure for that claim. TDG is specified using a state transition diagram shown in figure 2.6 to illustrate the procession of moves and player roles, and a schema for each illocutionary act which defines the pre, post and completion conditions for each move. TDG makes a number of novel contributions including a wider range of illocutionary acts and a new type of data store.

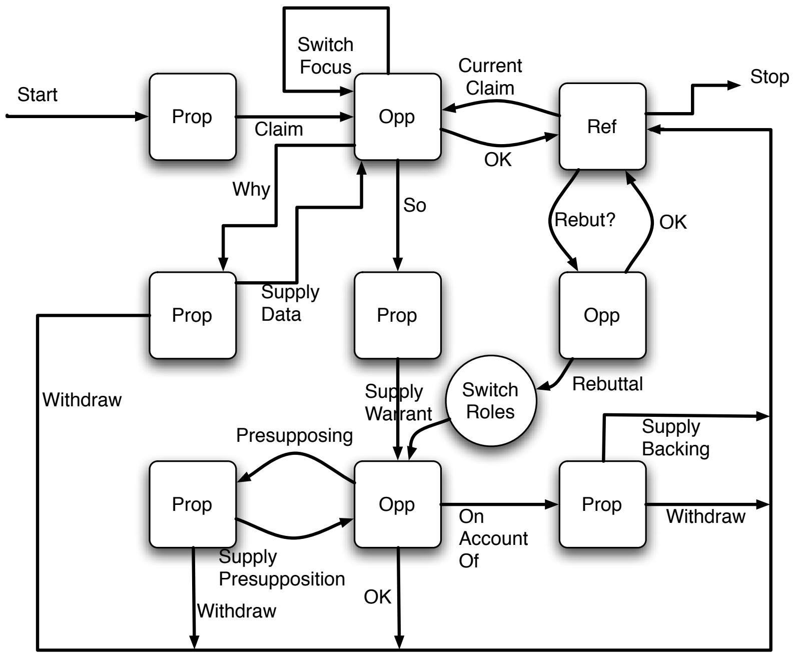

Figure 2.6: State Transition Diagram for the Toulmin Dialogue Game

The Toulmin layout of argument [129] suggests a structure for arguments in which the component propositions of the argument can occupy various roles. Bench-Capon identifies this as corresponding more closely to the way in which arguments are presented in real-world argumentation and proposes a modified argument schema, illustrated in figure 2.7. The modified Toulmin argument schema removes the qualifier, incorporates a presupposition role, and facilitates the chaining together of individual arguments such that the claim for one argument can fulfill the data role of another argument. This model of argument structure is explicitly supported by TDG through its wide range of illocutionary acts. TDG incorporates the notion of a claim stack, a shared data structure to which is added each new claim that the players make and whose contents are altered by the moves the players make during a dialogue. The presence and position of an individual claim within the claim stack is used to prescribe legality conditions for the moves of the game in addition to those conditions usually used to specify such things.

Rules of TDG

Claim (C) :

Description :
P asserts that C

Preconditions :
P has control of the dialogue

Postconditions :

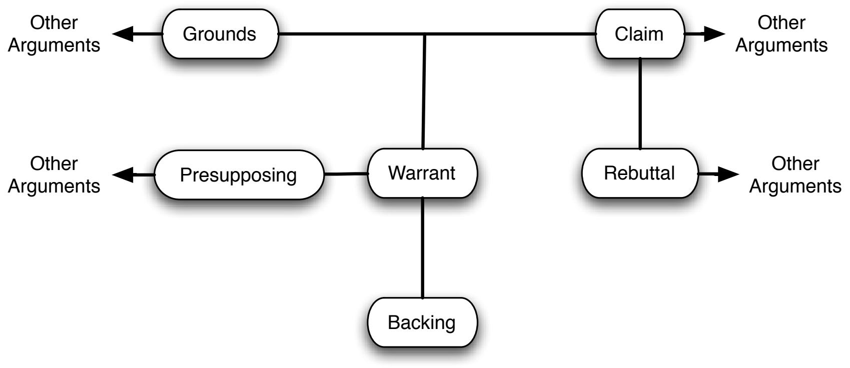

Figure 2.7: Bench-Capon's modified Toulmin layout

O has control of the dialogue
C is pushed onto the claim stack
P is commited to C

Completion Conditions :
C is popped from the claim stack

Why (C) :

Description :
O seeks data supporting C

Preconditions :
O has control of the dialogue
C is top of the claim stack

Postconditions :
P has control of the dialogue

Completion Conditions :
C is not top of claim stack

OK (C) :

Description :
O accepts C

Preconditions :
O has control of the dialogue
C is top of the claim stack

Postconditions :
C is popped from the claim stack
O is committed to C
O is not committed to not C
If not C is on claim stack, it is removed
Referee has control of the dialogue

Completion Conditions :
None

So (C) :

Description :
O requests the warrant for C

Preconditions :
O has control of the dialogue
O is not committed to if D then C, for any D for which he is not committed to ¬D
C is top of claim stack

Postconditions :
P has control of the dialogue

Completion Conditions :
C is not top of the claim stack

Presupposing (C) :

Description :
O requests the warrant for C

Preconditions :
O has control of the dialogue
If D then C is top of the claim stack

Postconditions :
P has control of the dialogue

Completion Conditions :
If D then C is popped from the claim stack

On Account Of (C) :

Description :
O requests the backing for the warrant of C

Preconditions :
O has control of the dialogue
If D then C is top of the claim stack
P has issued a supply warrant (C)

Postconditions :
P has control of the dialogue

Completion Conditions :
If D then C is popped from the claim stack

Supply Data (C) :

Description :
P asserts that D and that D supports C

Preconditions :
P has control of the dialogue
O has issued a Why (C)
C is top of the claim stack

Postconditions :
P is committed to D
D is pushed on the claim stack
O has control of the dialogue

Completion Conditions :
D is popped from the claim stack

Supply Warrant (C) :

Description :
P asserts that If D then C

Preconditions :
P has control of the dialogue
O has issued a So (C)
C is top of the claim stack

Postconditions :
P is committed to If D then C
If D then C is pushed onto the claim stack
O has control of the dialogue

Completion Conditions :
If D then C is popped from the claim stack

Supply Presupposition (C) :

Description :
P asserts that S

Preconditions :
P has control of the dialogue
O has issued a presupposing (C)
If D then C is top of the claim stack

Postconditions :
P is committed to S
P is committed to If not S then not C
S is pushed onto the claim stack
O has control of the dialogue

Completion Conditions :
S is popped from the claim stack

Supply Backing (C) :

Description :
P says that B is the authority for his argument for C

Preconditions :
P has control of the dialogue
O has issued an on account of (C)
P has issued a supply warrant (C)
If D then C is top of the claim stack

Postconditions :
R has control of the dialogue
O is committed to If D then C
If D then C is popped from the claim stack

Completion Conditions :
None

Withdraw (C) :

Description :
P withdraws his commitment to C

Preconditions :
P has control of the dialogue
C is top of the claim stack

Postconditions :
C is popped from the claim stack
P is not committed to C
R has control of the dialogue

Completion Conditions :
None

Switch Focus (C) :

Description :
O wishes to consider a claim not currently top of the claim stack

Preconditions :
C is not top of claim stack
C is on the claim stack
O has control of the dialogue
O is not committed to C

Postconditions :
C is moved to top of claim stack
O has control of the dialogue

Completion Conditions :
None

Current Claim (C) :

Description :
The referee passes control to the opponent of the current claim

Preconditions :
C is top of the claim stack
Ref has control of the dialogue
Player A is committed to C

Postconditions :
Player B has control of the dialogue

Completion Conditions :
None

End :

Description :
The referee terminates the dialogue

Preconditions :
Ref has control of the dialogue
The claim stack is empty

Postconditions :
The dialogue terminates

Completion Conditions :
none

Rebut (C) :

Description :
Player is invited to rebut an implicit commitment

Preconditions :
Referee has control of the dialogue
C is top of claim stack
Player is not committed to C
Player is not committed to if D then C for some D
Player is commmitted to D

Postconditions :
Player has control of the dialogue
Player is opponent, other player is proponent

Completion Conditions :
C is not top of claim stack

Rebuttal (C) :

Description :
Player provides a rebuttal, D, of C

Preconditions :
Player has control of the dialogue
Player is not committed to C
Other player is committed to C

Postconditions :
D is pushed onto the claim stack
Player is committed to ¬C
Player is committed to D
C is pushed onto claim stack
Player is committed to if D then ¬C
If D then ¬C is pushed onto claim stack
Player is proponent, other player is opponent
Opponent has control of the dialogue

Completion Conditions :
¬C is no longer on the claim stack

It should be noted that there have been both criticisms of dialectical games and recognition of possible limitations to this approach. Cummings describes formal dialectic as a circumscription of rationality, in the Putnam tradition, when applied to fallacy theory [22]. Walton also perceives some limitations in current dialectical games when applied to certain types of argumentation such as ethotic arguments which rely for their evaluation in part on the good character (*ethos*) of the speaker. The extant dialectical games do not provide sufficient resources to enable an adequate evaluation of ethotic arguments [136]. Nevertheless the investigation of dialectical games has not matured sufficiently to require alternative approaches to be sought. New applications for dialectical games are being suggested and explored, for example in the areas of AI and MAS, and new games can be developed to provide the necessary machinery to investigate a wider range of argumentative dialogue features. There are also many possible games that can be formulated from the non-overlapping features of the extant games. For example, only the Walton-related games make use of dark-side commitments stores so an exploration of a Girle DL based game with dark-side commitments might prove to yield dialogues with interesting properties.

2.5 Autonomous Agents & Multiagent Systems

Agents are software entities capable of intelligent, autonomous, and goal-oriented behaviour. Agents are identified with the modern face of artificial intelligence [110, pp. 4] and are often described in terms of mentalistic notions such as knowledge, belief, intention, and obligation. Such a mentalistic approach is often identified with Agent Oriented Programming (AOP) [118], a software engineering paradigm for building agent-based systems. Wooldridge [146, pp. 1-3] identifies a number of trends in the history of computing such as ubiquity, interconnection, intelligence, delegation, and human-orientation. The development of computer systems that can cope with the challenges posed by these trends has lead to the emergence of multiagent systems; societies of interacting, autonomous agents pursuing their individual and joint goals. MAS have four identifying characteristics [127, pp. 80] such that; each agent has incomplete information, there is no global control of the system, data is decentralised, and computation is asynchronous. This allows large problems to be tackled without risking a central controlling agent that could become a bottleneck or single point of failure for the system.

In order to achieve their goals agents peform actions. Actions are performed either in reaction to perceived changes in the environment or proactively to bring about changes in the environment based upon the agents own beliefs. Wooldridge and Jennings identify autonomy, as well as reactivity, and pro-activity as contributing towards a weaker, but popular, notion of agency [147]. Selection of the requisite actions requires agents to perform effective reasoning with regards to their beliefs, their environment, and the effects of their actions upon their environment. For a number of reasons it is not completely straightforward for an individual agent to unilaterally identify a goal, determine a sequence of actions that will satisfy that goal, and perform those actions. These reasons can include a lack of environmental resources, dearth of individual agent capabilities, or insufficient knowledge. Because an agent in a MAS is not acting alone it may have to compete with other agents for access to environmental resources, such resources might be as mundane as computer processing power, memory capacity, or communication bandwidth. According to Sycara [127, pp. 79] individual agents have bounded rationality, they are limited by their knowledge, resources and perspective. To overcome this, and to increase the agents problem-solving capabilities, organisations are created so that it is not necessary for an agent to have all of the capabilities required to meet its goals. If another agent in the organisation has the necessary capability then that agent can execute its capabilities on the agents behalf. Because agents are situated within dynamic environments it is assumed that an agent will not have perfect information about the environment, and that the information it does currently have may be out of date, inaccurate or incomplete. It is also assumed that because agents are individuals their thoughts are their own and that an agent does not know the contents of another agents knowledge base as this would violate the requirements of individuality.

Overcoming these kinds of problems motivates the need for inter-agent communication. When resources are scarce agents must communicate with each other to determine which agents get access to the resources. When agents need some action performed that they are personally incapable of performing then the agent must get the other agent with the requisite capability to agree to perform the action. Where an agent does not have the necessary knowledge about some aspect of the environment it needs a way to get that information from an agent that does have it. In each of these cases it is necessary for the agents concerned to communicate with each other or else give up some of the essential elements associated with agency such as autonomy or distributed control.

2.6 Agent Communication & Interaction

The last section introduced the notion of MAS as a society of intelligent, autonomous agents with incomplete information or limited capabilities. The need for communication between the individual agents was motivated but inter-agent communication is not solely the passing of data around between the agents. If correctly designed and implemented, effective communication mechanisms can aid the computational efficiency, reliability, extensibility, manageability, and robustness of the MAS [127, pp. 80]. To achieve this requires the design of higher-level information-oriented communication mechanisms between the agents. Such a communication mechanism enables the computational efficiency of a MAS to be improved by exploiting the concurrency of operation of individual agents. This is achieved by transmitting high-level information between agents rather than low-level data. Transmitting higher-level information also enables the agents to reduce the amount of communication required thus reducing the communications bandwidth required by the system. Communication enables the agents to dynamically coordinate their actions so that any failures in the MAS can be routed around thus improving the efficiency of the system. A MAS is more easily maintained and extended if inter-agent communication occurs in a high-level language because the external communications between agents and the information thus exchanged is not dependent upon the internal operation of the agents. If an agent is defective or if a replacement agent with enhanced capabilities is developed then it can be replaced without contributing any downtime to the system as a whole. Finally, by utilising the correct communication mechanisms the robustness of a system can be improved so that agents are capable of tolerating uncertainty in the information that they reason with and about.

Agent Communication Languages (ACLs) are specified to help achieve efficient and expressive higher-level communications between agents. An ACL enables agents to interact using analogues of human activities to guide the communicative process. ACLs are to a large degree based upon concepts identified in *speech act theory*. Some utterances can be identified that are not constative, they do not describe or report anything, but making the utterance is, or is a part of, the performance of an action. For example in executing a will, the sentence "I give and bequeath my watch to my brother" does not describe the act of giving the watch to the brother, neither does the sentence state that the action is being performed, uttering the sentence is actually performing the action of giving the watch to the brother [8, pp. 5-6]. Austin describes such utterences as *performative utterances*. Austin also identifies three attributes associated with performative utterences, these are the locutionary act, the illocutionary force and the perlocutionary effect [8, pp. 94-102]. The locutionary act is the act of making an utterance such as saying "you can't do that" whilst the illocutionary force describes the act associated with making the utterance such as protesting somebody's action. Finally the perlocutionary act is the effect of having performed an illocutionary act, in this case stopping somebody from doing something. Austin makes the case that uttering certain kinds of sentence is also to perform an act and that the successful utterence of the act, avoiding certain infelicities [8, ch. 3-4], brings about some real effect upon the state of the world.

Searle advances Austin's conception of illocutionary acts by proffering a theory of speech acts in which, for a range of illocutionary acts a number of rules are identified, satisfaction of which are a necessary precondition for the successful performance of the illocution [115, pp. 66-67]. The kinds of rules identified by Searle that govern the use of particular speech acts are the propositional content rules, the preparatory rules, the sincerity rules, and the essential rules and are specified in order to identify the "necessary and sufficient conditions" for successful performance of a speech act. An early use of speech act theory is suggested by Cohen and Perrault [21] who use techniques from AI planning research to specify pre-conditions and post-conditions for a range of speech acts. These conditions are then used to allow an agent to reason about the utterances it can make within a dialogue. The development of ACLs has been greatly influenced by speech act theory. Two such ACLs are The Knowledge Query and Manipulation Language (KQML) [33] and the Foundation for Intelligent Physical Agents Agent Communication Language (FIPA ACL) [34].

KQML was designed as an "outer wrapper" for constructing messages to enable agents to exchange information expressed in the Knowledge Interchange Format (KIF) [37]. To achieve this a KQML message is assigned a performative, a summary of which can be found in [146], and a range of parameters, attribute-value pairs that define meta-information about the message such as defining the recipient or recording that the current message is in reply to a previous message. KQML was designed specifically to enable heterogeneous agents, with different internal representations, to communicate by exchanging sequences of such messages where domain specific knowledge is encoded within a KIF sentence. A number of criticisms were made of KQML, summarised in Wooldridge [146, pp. 175], including that the set of performatives was too large, *ad hoc*, not sufficiently well specified to avoid ambiguity in disparate implementations, and that neither the transport mechanisms for exchanging messages nor the semantics of the performatives themselves were well enough specified. The biggest criticism from the perspective of a formal dialectical approach to agent communication, and from an agent communication approach in general is the lack of support for commissives, speech acts that enables one agent to make a commitment to another agent. Cohen and Levesque [19, 20] see the ability to manipulate commitments as a basic requirement of social action in a MAS.

The criticisms of KQML led to the development of the superficially similar FIPA ACL. The aims of FIPA are to create a standard communication language for autonomous agents which, similar to KQML, utilises a message-wrapper approach and does not mandate a specific language for representing message content, but which addresses the criticisms levelled at KQML by implementing an improved semantic framework in FIPA ACL over that in KQML [63]. Whilst not mandating a content language, FIPA ACL utilises the *Semantic Language* (SL) to specify the semantics of the performatives in terms of the beliefs, uncertain beliefs, desires, intentions, and actions of the agents concerned. To achieve this each ACL message is mapped to a formula of SL in order to specify a feasibility constraint which the message sender must satisfy before sending the message and a rational effect that is achieved by sending the message. Despite the attention paid to avoiding the failures of KQML, a number of criticisms have been directed towards the FIPA ACL. For example McBurney *et al.* [79] suggest that; because the aim of FIPA ACL is to model purchase negotiation dialogues it is deficient when applied to other dialogues with different goals because the participants cannot indicate the type of dialogue being entered into; the range of performatives is insufficient for sufficiently expressive argumentative dialogue; there is little in the way of dialectical structure to control sequences of utterances, or dialogue rules to govern the dialogue at a level higher than that of individual utterances. Another drawback of the FIPA ACL is that it is based upon private semantics, for example records of information internal to a given agent such as beliefs, which are difficult to verify in a heterogeneous agent setting. The dialectical game approach instead utilises commitment stores which provide a public semantics. The protocol for communicative agent interaction, as specified in a dialectical game, establishes public stores of information that can be maintained and inspected by all participants in the dialogue without requiring that the autonomy of the agents concerned be violated.

To mitigate some of the criticisms levelled at ACLs a number of approaches have been taken such as to create higher level interaction protocols that govern the types of messages that can be sent at any given point during a dialogue. Many of these protocols have been based on ideas from argumentation theory and were to a large extent spurred by the publication of Walton and Krabbe's work 'Commitment in Dialogue' in 1995 in which a partial dialogue typology is suggested [138, pp. 66] together with a protocol framework for engaging in critical pursuasion dialogues [138, ch. 4].

Walton and Krabbe's typology of dialogue types has proven to be influential within agent communication research where it has intuited the notion that there are numerous types of dialogue that an agent might engage in and that such an agent would be best served by utilising an interaction protocol tailored to the particular situation. To this end there have been a wide range of interaction protocols proposed that are based on particular dialogical situations identified either by type, goal, domain, or social considerations so that agents can select a conversation protocol suited to the current task [62]. Dialogue protocols that are used within agent research include protocols for inquiry dialogues [75, 74], deliberation dialogues [47], and negotiation dialogues [61, 95, 80, 78, 102]. Walton and Krabbe's dialogue typology has also informed recognition of the different types of interaction that agents might need to engage in to act within a particular problem domain. Although they do not suggest particular protocols for interaction, Dignum *et al* use the dialogue typology to inform recognition of the kinds of interaction that occur in cooperative problem solving (CPS) [26] and to aid in the formation of coallitions to tackle such problems [25]. More generally there are a number of computationally based protcols which could be applied to agents but which have been developed with non-MAS applications in mind. These include Lodder's DiaLaw protocol for legal reasoning [66, 65] and Bench-Capon's Toulmin dialogue game [13, 14], a protocol for regulating dialogue between human participants engaged in legal reasoning that is based on the argument schema of Toulmin [129] and that could be adapted to regulate inter-agent communication. An approach that is currently popular is to utilise formal dialectical games as the basis for interaction protocols between agents.

2.7 Argumentation in MAS

One form of argumentation has long been utilised in MAS and is based upon a conception of negotiation as one of the basic forms of interaction required by agents to resolve conflicts [125, 126, 94, 95]. This approach has been named argument based negotiation (ABN) and has been the basis for a lot of research into agent communications and interaction. More recently a richer conception of argumentation has been adopted within MAS research including the use of philosophically based dialogue games and the recognition of more interaction contexts than that of negotiation.

The two main approaches to argumentation in MAS can be distinguished based upon the distinction between two different senses of the term argument due to O'Keefe [91]. The two senses of argument are termed argument$_1$ and argument$_2$, where argument$_1$ refers to a particular kind of utterance, the argument that one makes, and argument$_2$ refers to a type of interaction in which argument$_1$ utterances are made. In O'Keefe's approach there is a difference between the process of argumentation as an interaction carried out during a dialogue, and the argument artifacts that are the currency of an argumentation process. The distinction between argument$_1$ and argument$_2$ can be seen in the ways that argumentation has been used in MAS where the construction of arguments has been tackled quite separately to the notion of argumentation as an interaction process, although not wholly separately because to exchange arguments between separate autonomous agents requires at least some sort of protocol to structure the interaction even if the protocol is very basic.

There is a very influential approach to building a relational structure of arguments which is due to Dung [28]. Dung introduces *argumentation frameworks* as a way to structure the attack and defeat relationships between arguments such that an argumentation framework is a pair consisting of a set of arguments and a binary relation between members of the set of arguments such that it can be said of any argument in the framework whether it attacks any other. Dung does not specify the structure of arguments themselves prefering to leave them abstract and concentrates instead on arguments whose roles are defined by their attack relationships to other arguments and the notion of acceptability of such arguments based on their roles and relationships. Elvang-Goransson *et al* introduce an argumentation system which specifies how arguments can be structured [32]. An argumentation system is a pair consisting of a formula, h, of a propositional language, and a set of formulas, H, of the same propositional language. There are additional constraints over the formulas associated with h and H but generally it can be said of the argument (H, h) that h is supported by H. These approaches been extended to facillitate preference ordering between arguments which builds on the argument structure of Elvang-Goransson *et al* and which incorporates a notion of undercutting attacks, defenses, and preferences between arguments [2]. A comprehensive survey of such logical approaches to argument can be found in Chesñevar *et al.* [18].

Many approaches have been taken to argument$_2$ type argumentation which either specify individual protocols for interaction or generic frameworks for specifying a range of protocols. Particular protocols for interaction, based on argumentative dialogue games, that have been suggested for use in MAS include a protocol for agent purchase negotiations [80], a protocol for deliberation dialogues [72]. The framework approach includes the generic framework of Maudet and Evrard [71], and the formal frameworks due to McBurney and Parsons [73, 77]. A drawback of these frameworks is that they specify flexible ways to implement a range of dialogue game protocols built from a range of generalised parameters and features of argumentative dialogue but do not take into account specific features of existing dialogue game protocols such as Mackenzie's DC which has been very popular, albeit in a number of slightly different and amended guises.

An integrated approach to a system of argumentation for MAS that integrates argument$_1$ and argument$_2$ is due to Parsons *et al* [96]. This incorporates a framework for Dungian arguments with the addition of preferences, undercutting attacks and defenses, and a generic framework for specifying dialogue game protocols based on work by Maudet and Evrard [71]. Another integrated approach is that of Bentahar *et al* which integrates the argumentation system due to Elvang-Goransson *et al* with a formalism for representing conversations called the commitment and argument network (CAN). A common theme of these integrated approaches is the attempt to provide a semantics for generic themes of dialectical argumentation that use a Dungian framework for representing arguments.

Generic approaches to argumentation have motivated the investigation of the attributes and properties of dialogue game protocols and their resultant dialogues. McBurney and Parsons identify a range of issues and challenges for the use of dialogue games in MAS [76] which include how to specify the semantics of dialogue game protocols, how to identify formal properties such as termination or computational complexity, for a given protocol, and how to support the design and assessment of new protocols. Jakobovits and Vermeir present a formalism of two-person dialogues that is used to investigate winning criteria under several dialogue types [49]. As it becomes easier to develop and deploy dialogue game protocols in MAS a pressing issue arises of how to compare and evaluate these different protocols.

2.8 Testbeds & Evaluative Techniques

The notion of testbeds, standard problems, and standard scenarios are tools that have been used in many areas of research as a measure of standard attainment, to see how much progress has been made, as a means to compare new against existing ideas and theories, and as a tool for explaining new ideas in easier terms. In section 2.2 the use of chess and other popular games as a measure of progress in AI was explored but there have been other approaches such as the use of various logical puzzles and mathematical problems. The Blocks World is a primitive scenario used to evaluate planning techniques [110] that is a simplified version of a standard scenario called SHRDLU used to evaluate research in constraint satisfaction programming, computer vision and natural language understanding [142]. McCarthy uses the Missionaries and Cannibals problem to illustrate a number of issues surrounding the use of circumscription to formalise common sense human reasoning [82]. The baseline Missionaries and Cannibals problem is as follows:

> Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals outnumber the missionaries on either bank of the river, the missionaries will be eaten. How shall they cross the river? [82, pp. 4]

McCarthy subsequently devises about 20 variants of the Missionaries and Cannibals problem which are deployed as a *Drosophila* for studying the use of elaboration tolerance in AI [84]. In this way McCarthy is able to test a solution strategy against the baseline problem before elaborating the problem with new information which means the original problem can be stress tested. The elaboration approach can lead to a library of variations on a single problem. Another approach is to have a library of different benchmark problems that can be deployed such as the list of 25 non-monotonic benchmark problems suggested by Lifschitz [64]. This approach has been used to test automated theorem provers using the Thousands of Problems for Theorem Provers (TPTP) library [124].

It has also been recognised that there should be some evaluation of the problems and scenarios to ensure that the results attained are the kind of results that are expected. Elio and Pelletier for example establish a range of human benchmarks [31] for the Lifschitz list in order to determine how closely the results from the AI software matched human ability.

Vreeswijk investigates the use of benchmark problems as tools to prove the correctness of defeasible logics and suggests an interpolation theorem that states that for every set of problems there is a defesible logic that satisfies those problems [131]. Vreeswijk suggests that solving more and more benchmark problems will not lead to an ultimate defeasible logic but rather to a logic that satisfies that set of problems. Following Vreejswijk's reasoning, satisfying various benchmarks and performing well in particular testbeds should not be the driving force in a field because the result is a system that performs well in those circumstances but that does not necessarily generalise to additional problems or lead to universal applicabililty. However testbeds give a way to measure certain aspects of a system that are otherwise difficult to appraise. Benchmark problems particularly enable the performance and capabilities of new systems to be comparatively evaluated against existing systems but the driving force behind the development should be to tackle particular aspects of the problem domain.

There is a need for equivalent approaches in computational argumentation and dialectcs systems. To this end Dignum and Vreeswijk investigate the use of a testbed to facilitate the exploration of issues in, and development of theories of, multi-party dialogues [27]. Their testbed uses a MAS and incoprporates a blackboard metaphor to facilitate communication between multiple agents who are all party to the same dialogue. Each agent is provided with an initial knowledge base to enable it to engage in dialogues within the system. The concern is mainly to demonstrate that a testbed can be constructed that enables multiple agents to engage in dialogue and that such a testbed is sufficiently flexible to allow a range of parameters to be set and a wide-range of issues to be investigated. The main drawback with this particular test-bed is that the protocol for interaction is very simple and there is not the facility to test the performance of a range of protocols against the framework.

2.9 Summary

This chapter has given an overview of research literature relevant to an exploration of dialectical games in MAS. The topics that have been explored either provide necessary background, which motivates the study of dialectical games and MAS, or underlies the approach taken in later chapters.

Chapter 3

Analysis

THREE basic issues in dialectical games can be identified. Firstly, many dialectical games have been developed and yet there is not a clear definition of what constitutes a dialectical game. Secondly there is no consistent approach to representing dialectical games. Individual game developers utilise their own representational formats, albeit with some overlap between them. Finally, many developers produce games that introduce new game components, or new ways of using existing components, but there has been no attempt to draw all of these features together and to delineate the resulting *game space*.

The first issue means that it is difficult to determine exactly what is and what is not a dialectical game. For example, if a game includes rules that specify a context of interaction, e.g. that one move should follow another, then is that game a dialectical game or merely a protocol? Similarly, if all of the rules that are related to commitment are removed from Mackenzie's DC [70] then is this DC variant still a dialectical game?

The second issue means that it is difficult to determine when two dialectical games are actually functionally identical but superficially different because they are represented differently to each other. Johnson *et al.* [50] investigate this issue and introduce the idea of equivalence between protocols. Equivalence has a number of forms and can be formulated in terms of the rules of the game, the legal states of player commitment that can occur during a dialogue, the states that can lead to the termination of a dialogue, and in terms of the dialogues themselves. A different approach, that has a broad range of benefits, is to identify a common way to represent dialectical games so that a rule formulated in one game that is equivalent to a rule formulated in another game is represented in a similar fashion in both games. It is such an approach that is developed in chapter 4 but the basis for that approach is developed here by investigating how existing games represent their rules.

The third issue means that dialectical games are considered in terms of individual sets of rules rather than as a space of potential games. By using a common representation that incorporates the range of features found in existing games such a space can be delineated and new games, new points in the game space, can be more easily identified and investigated. It is important that developers of new games can take advantage of the entire range of components that the extant games have introduced.

This chapter surveys a range of dialectical games and analyses their rules and components in order to identify what a dialectical game is and how it is represented, what components such a game is built from, and what the shape of the resulting game space is like. It begins by providing a basis for the need to unify approaches to game representation. A high level survey is conducted that investigates how existing dialectical games are currently represented. Following the high level survey an analysis of each game is then performed and the results of this analysis are collated and used to construct a list of desiderata for a unified representation of dialectical games.

The aim of this analysis is therefore to achieve the following;

1. Identify techniques for analysing dialectical games,
2. Identify the range of components used by extant games,
3. Identify the range of representational schemes used by extant games,
4. Identify the range of components that a unified representation of dialectical games requires.

The desiderata identified in this chapter form a necessary and important basis for the movement from individual *ad hoc* game development and implementation towards solid, robust, repeatable, and well-engineered software systems that can be adopted by the wider agent community as a part of their approach to inter-agent communication.

3.1 A Basis for Unification

The adoption of dialectical games in MAS is an increasingly popular pursuit however current approaches to implementation of arguing agents are *ad hoc* relying on individual developers to either specify games or to adopt an extant game. Software implementations are scarce. Often the authors of games do not provide a reference implementation so each time the game is adopted the entire game is re-implemented. Although some games, such as DC, have become popular, there has been no attempt to develop a representation of those popular games which can be shared between different developers. Furthermore the extant games are usually specified in natural language without a formalism. What this means for the developer is that their implementation relies upon an interpretation of the original authors intentions for how the game should be implemented, assuming the author even intended a computational implementation. This is partly due to the ambiguity inherent in natural language and also partly due to some games never having been intended for computational implementation. Hamblin's and Mackenzie's games for example were developed to illustrate aspects of fallacy research by enabling the researcher to either analyse a real world text and show the structure of interactions that could bring about a given fallacy, or else were intended to enable the researcher to demonstrate through the generation of dialogue according to a set of rules, the mechanism by which certain features of dialogue occurred. The result of this is that developers can produce different implementations of the same game, each based upon different interpretations. Additionally there are often a number of interpretations that can be taken, each of which translates, in a strict sense, into a different game. So the question becomes less of which games does an agent play, but which interpretation of a given game does the agent play.

These issues can be made more clear by considering a heterogeneous MAS. Such a MAS is constructed from multiple intelligent autonomous agents, all developed separately and having different owners and operators. In such a MAS it is important that agents can communicate otherwise there is not a cohesive MAS but a collection of independent agent programs. If the protocols that the agents use to facilitate their communications are not clearly and unambiguously represented then there arises the possibility that individual agents may be using incompatible protocols. This can occur if the agent's developers interpret the rules differently to one another. If it were possible for an agent to configure its internal representations of a dialectical games to match that used by another agent to whom it is in communication with then there is the potential for agents to have a much greater capability for interaction.

A recent trend in MAS research has been towards frameworks such as JADE[35], Jackdaw[67], and JACK [42]. MAS developers no longer have to implement their agents from first principles because they can utilise an existing framework that implements the core agent functionality. This leaves more time to focus on the specific problem at hand. This kind of trend is repeated in many areas of software development. As a concept becomes popular so frameworks are developed to support the utilisation of those concepts in new software. This kind of trend enables new ideas to move from *ad hoc* individual implementations towards standards, enabling interoperation and robust, repeatable implementations. There is a need for a similar approach *vis-à-vis* dialectical games. This is due to the increased focus on theoretical aspects of such games, the popularity of the application domains of arguing agents and computational dialectics, and the relative paucity of tools for getting from the theory to implemented software without repeating the steps taken, albeit seldomly, by others.

In the case where there are multiple interpretations of a given set of rules, each interpretation can be considered a distinct game ready for implementation. For example in PPD_0 [138], rule 3(e) specifies that following a challenge the set of legal moves consists of an elementary argument or a retraction of commitment (of whichever strength) only, but rule 8 specifies that the extractor move can only be used if the preceding move was a challenge or retraction of commitment. In this case rule 8 contradicts rule 3(e) because rule 8 allows an extractor to follow a challenge but rule 3(e) prohibits it. In this case at least two distinct games could be implemented, one in which the extractor move may follow the challenge move and one in which this is not the case. Although a unified framework does not of itself enable an implementer to determine the authors original intent in formulating a particular rule, it does enable multiple interpretations to be made explicit and explored in an efficient manner and the most appropriate variant according to the developers needs can be adopted. As a result, the effect of such ambiguity problems on implementational considerations is greatly reduced. Additionally given a unified implementation framework with a common runtime the costs to implement each variant are considerably reduced.

Whilst such a framework reduces the amount of repetition that each developer must do it doesn't help a developer to select which game to implement. Whilst Mackenzie's DC has been a popular choice, possibly because it is better known, there is little to support that game being a better choice for adoption in a MAS than Hamblin's H, or Walton's CB, for example. One of the benefits of a unified framework, a framework that supports a wide range of games, is that it would make the migration from one game to another easier to achieve. Developers could implement games using a framework and select a game after testing each of their prospective candidate games with less effort than having to implement each game afresh.

When many games are able to be played by a single agent, dialectical game use can become more of an issue of runtime game selection. Instead of the developer being required to select the best game or games *apriori* to suit all of the dialogical contexts in which the agent might have to act, an agent can be made capable of selecting the game based on the context in which it is required and loading the rules of the game as needed. Agents thus become runtime extendable rather than fixed entities that require to be recoded to add new dialectical capabilities.

An added benefit is that the use of standard components to implement capabilities such as dialectical game playing reduce the risks of incompatible implementations. Because a core aspect of agency is communication within a social environment with other agents, and those other agents may have been developed by other parties, the use of a shared, possibly standardised, component reduces the risk of such incompatibilities. It does not remove the risk however but merely reduces it as there are other capability areas where agents can become incompatible such as in the syntax that agents use to express their knowledge. For example, three agents enter into a dialectical game and all three have the same dialectical game component. However, the first agent expresses its knowledge as propositional formulas, the second as predicate statements, and the third as natural language statements. Although the agents can play the same game, they cannot communicate in a compatible fashion.

Yet another benefit of a unified framework is that as new concepts are added to the notion of a dialectical game so the framework can be extended to allow the new game to be represented but also for the new concept to become a standard component of yet more games. For example, H incorporated a number of elements which have become the basic components of formal dialectical games although Hamblin did not define limits for what a formal dialectical game should consist of. As new games have been developed so the range of basic components has increased. New illocutionary acts and rules governing their use have been introduced. New types of commitment store such as the set of assertions and the set of mere concessions [138], and the dark-side commitment set [133], have been suggested to stratify the contents of the players commitments stores. Wholly new dialogue artifacts such as the claim stack [13] have also been suggested. The aim of introducing these elements has been to enable formal dialectical games to better perform in the task to which they were applied. In adding these elements to the range of components available for building dialectical games, not only does it result in a wider range of possible games but allows a developer to more easily develop a system that is tightly coupled to a particular problem or situation. The idea of a dialectical game then becomes less that of a fixed set of rules for an isolated game and more of space of possible games delineated by a range of game components and component interactions. In this space many dialectical games can be constructed merely by recombining the elements from which existing games have been constructed. This can enable better games to be more easily developed to explore issues in argumentation and multiagent dialogue.

The way that a dialectical game is represented can form a barrier to comprehension and thus to the adoption or utilisation of that game. If a developer cannot easily comprehend how a game will proceed, or if it is difficult to determine the requirements of a given move or the effects of actually playing that move at any given point in a dialogue then there is a good chance that a better presented game will be adopted. From an ease of use perspective, there should not be a high cognitive burden to utilising a game. It can also be difficult to determine where two games differ in the formulation of their rules, to determine this requires close reading and comparison between the games. For example the rules of DL, DL2 and DL3, as shown in section 2.4.4, are all very similar, with similar numbers and types of locutions and rules, but they are sufficiently different to each other for their author to name each as a distinct system. Once the differences are determined it is not clear however that the differences in rule formulation of each system actually affect the resultant dialogues, and if there is such a resultant effect, how pronounced it is. This is not an isolated case either because the same could be asked of the CB family of games from section 2.4.3 or indeed of any of the dialectical games outlined in section 2.4. In fact the problem is more pronounced when games from different authors are compared because at least the CB game and the DL games, within their own families, share a common layout and presentation. The issue of comparison between disparate games has not been investigated to any depth in the literature although [50] and [79] take some steps in that direction by defining a notion of equivalence between dialogue games and setting out some possible attributes of dialogue games. Issues such as these can be tackled through adoption of a unified schema for representing games in addition to the use of a computational framework for implementation.

A desirable quality of a unified representation is that it support the implementation of existing games from the literature. This is not a straightforward task however because of differences in specification and instances of ambiguity, such as the incompatibility between rules 3(e) and 8 of PPR_0, and errors which come to light during detailed investigations. It is important though because the games were developed to solve clear goals and represent a significant investment of research effort which should be built upon. H and DC for example were instrumental in building models of the action of the *petitio* fallacy in dialogue [44, 144, 145, 70]. Moreover DC has been suggested as a system to regulate the interactions in a dialogue based human computer interface [86]. DL and DL2 have been applied to belief set construction and revision [39, 40]. IDL3 and DE have been proposed as the basis for dialogue capabilities in computer-based learning [41, 148].

The aims of a unified specification can be summarised as follows:

1. To provide reference implementation for extant games that do not have a public computational implementation.
2. To provide an API to facilitate the rapid design, implementation, and deployment of games in computational contexts.
3. To reduce repetition of effort.
4. To reduce time to implementation.
5. To increase the ease of migration between games.
6. To provide a platform for evaluation, testing, and comparison of games.
7. To simplify the development of new games.

Most importantly, such a unified approach helps developers to tackle the question of which game to implement. Rather than sinking valuable time into developing the necessary architecture to support an implementation, a developer who adopts the use of a unified representation and implementation framework can devote more time to exploring existing and new games rather than reinventing the wheel. Such an approach though is predicated on the notion that extant games can be unified into a single meta-level game. This requires that the extant games are analysed to perform the basis for such development, an activity that is pursued over the remainder of this chapter.

3.2 High-Level Summary

This section investigates what constitutes a dialectical game in a definitional sense. For example, what components must a dialectical game incorporate and what restrictions are there over those components. Once dialectical games have been defined an investigation is made of the ways in which existing games have been represented. Each game is then investigated to identify the kinds of components that it uses and any restrictions that are imposed over the use of those components. Finally a summary of the components and features found in extant games is presented.

A Basic Definition

Hamblin gives a general working definition for formal dialectical games. These are defined as, "a regulated dialogue involving a number of participants who speak in turn in accordance with a set of rules that specify the form of what is said relative to the context and previous occurrences in the dialogue". Hamblin further refines this definition to specify:

1. The number of participants is set to two. Because the central concern of [44] is specifically with two person dialogues the matter of discriminating over the direction of utterances is dispensed with and the assumption is made that all utterances are directed towards the speaker's opponent.

2. Previous occurrences refer to the specification of an interaction between the players. In order to establish the circumstances and conditions under which a fallacy might occur, Hamblin creates a system where playing a move affects what moves can be played in subsequent turns. This enables a sequence of moves to be used to model the flow of utterances involved in a particular aspect of dialogue such as the progression of question-answer moves in an information seeking dialogue or the challenge-defence moves of a persuasion dialogue.

3. The context refers to the contents of the players commitments stores. Whilst the previous occurrences are used to designate when a move is a member of the set of *potentially* legal moves, the commitments of the players can be used to further exclude particular locutions from that set.

Hamblin says of the rules of formal dialectical games that they may prescribe, prohibit or permit, may be directed towards particular players and may be conditional on previous occurrences. The rules of the system specify the kind of things that the players can utter during their turn but not the exact content or formulation of those utterances. By uttering locutions and associated content in accordance with the rules, players incur and retract commitment with respect to their, and their opponent's, utterances. From this naive analysis, three core characteristics of Hamblin's formulation of dialectical games can be recognised. These are the interaction context, the dialogue transcript and the notion of commitment. These components form the core of the original Hamblin-type Formal Dialectical games.

1. Interaction Context - The interaction context is the specification of a protocol for the dialogical interactions between the players. All interactions are effected by the application of moves. The interaction context provides part of the specification for individual moves within the system.

2. Transcript - The transcript is the artifact referred to as the dialogue history. The transcript is constructed by the players whilst engaging in dialogue according to the rules.

3. Commitment Model - The players introduce statements to the dialogue through their utterances. Players maintain a public position in relation to the statements made by themselves and their opponents. Commitment is used to model the position that the players maintain in relation to the statements made. The players position, in terms of their burden of incurred dialogical commitment is handled by way of commitment stores, a publicly inspectable collection of the commitments of each player. The commitment model plays two important parts in the specification of dialectical games. Firstly, commitment provides an additional dialogue artifact which supplements the transcript. Secondly, commitment provides a counterpart to the interaction context for the purposes of specifying individual moves.

3.2.1 Representing the Rules of Dialectical Games

Dialectical games have been described in two ways, through natural language description or through a formal description. The majority of games make use of a natural language description, occasionally this is supplemented with some formal description of important areas. DC is the only system that is presented entirely through a formal description, although this is in the appendix of [70] and supplements the natural language description in the main body of the paper. Additionally a number of modes of presentation have been utilised to set out the rules of the dialectical games. These are:

Function Oriented Groupings

The rules of H in Hamblin's original specification [44] are arranged into three groups which deal with locutions, commitment, and the structure of dialogue interactions. Locution rules set out the range of legal locutions in the system, commitment rules set out the effects of the locutions on the players commitment stores, and the structural rules set out the conditions under which a locution can be legally uttered. This broad structure is maintained in the games of Mackenzie, and of Girle, although with some variations in terminology. For example, the group of rules that Hamblin referred to as syntactical rules are variously referred to as rules of dialogue, by Mackenzie, and interaction rules, by Girle. The games of Walton and Krabbe introduce a further grouping, that of termination rules, to account for the conditions under which a dialogue should terminate. Walton also introduces two new rule groupings. These are the consistency rules, a group of rules which aim to enforce consistency over the content of player's commitment stores, and the resolution-mechanism rules, which groups rules relating to their use of contradiction-resolution locutions.

The majority of distinct, named games are introduced using the locution/structure/commitment/termination group layout so it is familiar to developers of dialectical games. The main drawback is that it is difficult to determine the legality requirement or effects for any given move. The locution rules set out which moves are available in a system but they are only potentially legal until their requirements are satisfied. To determine what the requirements are for any given move requires inspection of at least the structural moves group which might contain a large number of individual rules. During a dialogue game a particular move isn't selected merely because it can be legally played, unless of course the speaker has been manoeuvred into such a situation where a move has to be played to avoid forfeiting the game, but also because the move will bring about some desirable effect in the game. To determine these desirable effects requires the player to also the commitment rules, and possibly re-evaluate the structural rule in relation to effects rather than requirements. As can be seen from the presentation of various games in section 2.4 there may be a large number of rules which regulate a dialectical game and hence this presentation style is not conducive to user comprehension.

Hamblin used three groupings of rules. The first group was not labelled but can be referred to as locutional rules. The other groupings were labeled syntactical rules and commitment store operations. In addition to the core system Hamblin provided supplemental rules which could replace specific core rules. This enables either for the capabilities of the core system to be extended or for the focus of the system to be tightened and brought to bare on particular problems. Mackenzie's system DC employs the same basic structure as H. DC specifies locutional rules, commitment rules and rules of dialogue. The so-called rules of dialogue are analogous to the syntactical rules of H. These rules can be categorised as follows:

1. Locution Rules
2. Structural Rules
3. Artifact-Store Rules

4. Termination Rules

Dialectical games make good use of some concepts from speech act theory. This approach provides useful tools and concepts for analysing the rules of dialectical games. Speech act theory identifies locutionary form, illocutionary act, felicity conditions and perlocutionary force. These account for the form of an utterance, the act performed in expressing that utterance, the conditions required for commission of the act to be felicitous, and the effects on the hearers of the expression of the utterance.

These concepts can be used to describe the mode of action of dialectics wherein there is a formulation of what is said, there is the act performed in saying it, the requirements for the utterance to be considered legal, and the effect on the participants to the conversation. In real-world utterances it is not always clear what the illocutionary and perlocutionary acts are, whereas in a dialectic system these aspects are made explicit through the specification of moves and locutions.

Illocutionary acts correspond to the locution types of a dialectical game and are formulated in terms of the locutional rules for the game. The requirements for, and the effects of making, a move within a dialectic system, are an explicit representation of the felicity conditions and perlocutionary force of a speech act and represented in the structural rules for a game.

The locutional rules set out a taxonomy of the locutions allowed by a system. Each locution may specify certain conditions under which it is legal, although this is not mandatory. An essential part of a locution is it's effect. Without an effect a locution doesn't actually make a change to the game so an effect is an essential part of a locution's specification. The majority of game specifications, with the exception of TDG, do not however set out each locution, along with its legality conditions and rational effects, in one place to make the game simpler to comprehend but present each move split across the entire specification for the game. This can make it difficult to determine when exactly a move is legal and also, if the locution can be legally played, what effects that should have on the rest of the game. The set of locutional rules themselves in any given game specification generally set out no more than just the form of the locution, in terms of the performative verb and the statements that can be applied as content alongside the performative.

All of the extant games specify structural rules to some degree. These are the rules that specify the dialectical structure of a game, how the legality of a given move is dependent upon the moves played in earlier turns, and in turn can affect the moves that may be played in subsequent turns. With respect to their dialectical structure, moves can be described as dialectically-restricted, meaning that any responses are restricted to a particular subset of move types, or dialectically-unrestricted if they specify no mandatory response types. There are three varieties of game with respect to their structural rules;

1. Dialectically *open* games are those which do not mandate any dialectical structure so any move can be legally played at any point, and the playing of one move in turn *t* does not restrict the moves that the player can make at *t+n*. There no examples of completely open dialectical games within the literature because all dialectical games have at least a subset of their moves for which a set of allowable responses is specified.

2. Dialectically *closed* games are those which, for every move, specify a set of mandatory response types that is a subset of all of the moves specified by the game. An example of a dialectically closed game is the Toulmin Dialogue Game [13] which for every single move specifies a set of responsive moves.

3. The majority of extant dialectical games are in the *partially-open* category in which only some of the moves specify some form of a mandatory response. On average, examples of partially-open games specify dialectical structures for only about half of their moves, the rest of which are unrestricted. The partially closed-game represents a compromise between the open game which allows any move type to be played at any point, and the closed game which always restricts the types of available moved, which enables a game to be more flexible whilst also ensuring that certain dialectical interactions occur.

Most games identify a group of rules called commitment rules which deal with how commitment is incurred and retracted. This is the aspect referred to as the commitment model in section 3.2. Commitment though is only one of a range of artifacts that can be recorded during a dialogue and used to structure game rules. Commitment rules are therefore identified as a subgroup of a more general group of *artifact-management rules*. The artifact-management rules deal with how artifacts of dialogue are created, stored, examined and manipulated. The range of artifacts is small and currently incorporates the following three types, commitments, claims, and locutions.

Hamblin introduced the notion of commitment and commitment stores as a tool to aid in consistency maintenance. The aim was to ensure that rules could be specified such that each new statement made by a player was not inconsistent with earlier statements made by the player during the dialogue. This was achieved by creating a store for each player, the contents of which were the running tally of commitments incurred by that player. Hamblin suggests that to maintain consistency in a players statements, it is required "of each new statement that it may be added without inconsistency to" the player's commitment store. The notion of commitment stores has been extended to account for the different types of commitment that a player can make. Walton introduces the veiled commitment store to account for additional types of commitment and this notion is further refined in Walton and Krabbe's PPD_0 which makes use of dark-side commitments. Whilst Hamblin's commitment store was suggested to keep track of commitments that are incurred as a result of a player's explicit utterances, the veiled and dark-side stores account for a more psychological model of certain commitments, those that a player has, which are not necessarily explicitly expressed during a dialogue. However a player is required to maintain consistency between their explicit commitments as a result of their utterances, and those prior commitments that are in their veiled store. In this way, Walton and Krabbe attempt to provide rules to ensure that things that a player says within a dialogue are consistent with the prior beliefs of the player before the dialogue. Another type of artifact introduced by Bench-Capon in TDG is the claim which is maintained in a claim stack [13]. Claims are similar to commitments insofar as they are created when certain moves are made and are recorded in a store of similar artifacts. Claims are also used to formulate rules for the game similar to commitments and commitment store contents. The players use particular locutions during a dialogue to push and pop claims onto the claim stack and to otherwise manipulate them. The use of a claim stack rather than merely a store of claims means that claims are dealt with in a particular order during a dialogue. The initial claim takes the first position on the claim stack and subsequent claims are pushed onto the stack on top of the initial claim. As the dialogue continues and any issues associated with each claim are resolved so the dialogue eventually returns to the original claim. At this point the participants have already explored many of the underlying arguments related to the initial claim through handling those other sub-claims.

The transcript of a dialogue is the sequence of locutions that have been uttered. This sequence is ordered according to the turns that the players have taken. Each locution can be considered an artifact of the dialogue with the transcript being another artifact store, albeit one which is not explicitly represented in any of the extant games. The transcript, as a record of all previous moves played during the game, is however handled implicitly in a number of games where rules specify a legality condition regarding earlier moves. For example structural rules 3a to 3e of PPD_0 all specify conditions which affect whether a given move is legal based upon previous moves, e.g. if the preceding move was resolve(P,Q) then the responsive move must be either no commitment (P) or no commitment (Q).

The number and range of artifact stores is likely to increase as new games are developed. This enable those new games to better satisfy the issues which they are developed to tackle. This is supported by Hamblin's recognition that "though, presumably, the brain of an actual speaker must contain some remote analogy of a commitment-store, it contains much else besides". It is the functionality of the elements of the"*much else besides*" which artifact stores of various kinds try to capture.

The variety of termination rules, those rules which set out how and when a dialogue can be brought to a close, are signifiers of the range of application domains to which dialectical games have been applied.Games that are more focussed upon the issues relevant to the occurrence of fallacies during a dialogue, H and DC for example, are less concerned with the issues of starting and finishing said games than with the dynamics of the ongoing dialogue. In these cases it is assumed that the dialogue is underway when the fallacy or problem occurs. Often in these cases fragments of dialogues are of interest rather than entire dialogues.

Similarly games that are designed to enable human participants to engage in rule-governed dialogue assume that either the participants can agree that they have reached the end of the dialogue and as a result can stop talking, or else that they have established limits to the dialogue beforehand. In either case the dialectical game relies upon external factors, such as human capabilities of reasoning, rather than internally specified game rules. Machines unfortunately do not possess the same capabilities to determine when to exit a dialogue, or how long the dialogue should be. Computational dialectical applications, especially those aimed at dialogue production rather than analysis, require that some form of termination state can be both specified prior to the dialogue and identified once it is reached.

The lack of rules governing the opening of dialogues is not a serious problem as the conditions required to begin a dialogue can be inferred from other aspects of the situation. For example, if there is at least one move which can be performed then that move can be used as an initial locution to open the dialogue. The rest of the dialogue then proceeds according to the rules which specify legal responses in light of the opening move. It is useful to have a specification for the initial state of a games data stores but where this is not the case it is assumed that such stores are empty.

The lack of termination rules is more ominous in the context of computational dialectics. Without termination rules the agents engaged in a dialogue must have a means to withdraw from the dialogue once their goals have been achieved, or if they determine that said goals cannot be achieved given the current state of affairs, or if the underlying assumptions on which the dialogue is predicated have changed. The alternative is that agents remain stuck in a dialogue that they cannot leave resulting in a waste of communicative and computational resources. This necessitates that any developer adopting a dialectical game must ensure that it includes rules for determining when the dialogue is complete. If a game is adopted that does not include such rules then some alternative means of identifying dialogue termination must be used, possibly through a meta-level game. Termination rules are therefore an issue that developers should take into account during the selection of a dialectical game for MAS applications. Dialectical games that do not specify such rules are therefore under-specified with respect to the MAS application domain.

A variety of methods can be found for specifying dialogue termination rules including;

1. Enumeration of dialogue artifacts and predetermined limits
2. Commitment state of participants
    (a) Incur commitment to opponents thesis
    (b) Incur commitments from which the opponents thesis follows
    (c) Retraction of commitment
3. Out of bounds

An enumeration of dialogue artifacts can refer to the number of moves, turns, statement symbols, or any other such artifact generated during the course of a dialogue. Walton's CB-based games utilise this method, requiring that a dialogue terminate after a predetermined number of moves. Similarly Walton and Krabbe's PPD_0 requires that a dialogue be limited with respect to the number of its symbol tokens. Both approaches provide a way to limit the length of a dialogue and determine a point at which the dialogue should be drawn to a close. Other approaches that can be used, especially in more controversy oriented dialogues, are the commitment state of the participants. Once a player incurs commitment to their opponents initial thesis, or retracts commitment from their own thesis then that point marks a state at which the associated dialogue can be terminated. Players could continue past this point to determine any unexpressed assumptions that underlie the current state however. Walton also specifies in CBZ that a player who breaks the rules by uttering a locution that is illegal and out of bounds at the current point loses the game as a result.

In addition to identifying the termination states of a dialogue, a related issue is the determination of the termination status with respect to each participant. In a ludic sense this corresponds to the determination of which player has won the game. The termination status of a complete dialogue can correspond to three situations;

1. The game is won with respect to a given player,
2. The game is lost with respect to a given player,
3. The game is drawn with respect to a given pair of players.

Whilst the utterance of an out of bounds locution leads to forfeit of the game for the uttering player, the commitment state of the participants at the termination point can also be used to determine whether a player has won or lost. A player who retracts commitment to their initial thesis or commits to their opponents thesis has lost the game with respect to that thesis as is demonstrated in PPD_0. Another way to determine termination status is in terms of the number of points accumulated during the dialogue. CB awards a point to each player who accepts as a commitment a statement of their opponent. The winner of a CB dialogue is the player with the greatest accumulation of points at the end of the dialogue. Such approaches, to determining both termination state and termination status, are required of games that are to be applied in a generative computational context.

Function oriented groupings are the most common way to specify a dialectical game but the is no agreement as to which grouping should be included as standard in the description of a dialectical game. Additionally, because most games are expressed in natural language using function oriented groupings, they are not explicit enough to represent dialectical games so that ambiguities can be eliminated.

Rule Operation Tables

The rules of the DL3 system are presented using function oriented groupings. Additionally, to aid comprehension of the system, Girle [41] introduces the *rule operation table* which presents, for each locution in the game, the alterations which must be made to the players commitment stores and the legal responses in the next turn. An example rule operation table can be found in table 3.1. The benefit of this approach is that it aids a player to comprehend the effect of playing a given move. In addition the effects of playing the moves can be easily compared across the range of moves for a given system. A drawback of this approach is that although each legal locution is laid out, the rule operation table does not include the legality requirements for each locution, so although a player can easily see what the effect of making a move is, they still have to resort to the original rules to determine when the locutions can be uttered.

S LOCUTION at Step *n*	**S** STORE	**H** STORE	**H** RESPONSE
categorical statement: *P*	+ *P*	+ *P*	
reactive statement: ∵*R*	+ *R* + *If R then S* + ⟨ ··· *Why S?* ⟩ –*Why S?*	+ *R* + *If R then S* + ⟨ ··· *Why S?* ⟩ –*Why S?*	
term declaration: *t*	+*Ft*	+*Ft*	at some later point *Ft* or ∼*Ft*
withdrawal –*P*	–*P*		
withdrawal –(∃*x*)*Fx*	–(*Qx*)*Fx* –(∃*x*)*Fx*	–(*Qx*)*Fx* –(∃*x*)*Fx*	
tf-question *P?*	–*P*		one of *P*, ∼*P*, *τP*, *or* –*P*
wh-question (*Qx*)*Fx*	+(∃*x*)*Fx*	+(*Qx*)*Fx* +(∃*x*)*Fx*	one of t, ∼(∃*x*)*Fx* *τ*(∃*x*)*Fx* or –(∃*x*)*Fx*
challenge *Why S?*	–*S* +*Why S?*	+*S* +*Why S?*	one of *acceptable* ∵ *R* or ∼*S* or –*S* or a resolution demand as in (*v*)
resolution *Resolve P*	+*P* +*Resolve P*	+*P* +*Resolve P*	withdraw part *P* or state part *P* as in *(vii)*

Table 3.1: Rule Operation Table of DL3

Conditional Descriptions

The Toulmin dialogue game (TDG) [13] is laid out in terms of the pre-conditions, post-conditions and completion conditions for each move. An additional description is included for each move although this does not introduce extra functionality and serves merely to provide a useful mnemonic for human users of the system. This is a good way to present a formal dialectical game from a human comprehension standpoint. The layout is move-centric. For each move that a player can make the conditions setting out legality, rational effect, and successful completion are grouped in a single unit. For a person playing the TDG it is a simple matter to find out which moves can be played in response to any other move and the effects of doing so. A drawback is that it is difficult to fit rules which have a global effect into this layout. If a rule is to be applied after every move, such as to determine the termination status of the dialogue by examining the players commitment states, then a formulation for the termination conditions must be added to the completion conditions for every move. This is wasteful and increases the size of the specification.

Locution Attribute Table

The *locution attribute table* is a novel method developed in this thesis for presenting the moves of a dialectical game. The rationale was to develop a tool to support the locutional analysis of dialectical games to overcome the shortcomings of the function-oriented groupings, in which the rules pertaining to each move are spread throughout the rules of the entire game, and the rule operation table, which deals only with the effects of moves and not the legality conditions associated with them. The locution attribute table is constructed using three columns. The first column stores the locution, the second column stores the desideratum for the associated locution indicated in the first column, and in the third column the effects of the locution indicated in the first column being successfully uttered. A locution is successfully uttered, within the confines of a dialectical game, if and only if the requirements for that locution, as defined in the *desideratum* column are satisfied. The table can be *read* using the following technique. For each entry in the locution column, *if* the conditions specified in the desideratum column are specified *then* the effects specified in the effects column are applied.

Because the requirements of each move are included in the table, this approach tackles the biggest perceived drawback of the rule operation table as an analytical tool. In addition the locution attribute table is more flexible than the rule operation table because it allows for more than one commitment store per player and does not specify that the range of effects only include commitment set updates and responsive moves. Some nomenclature is specified to efficiently represent the elements that might occupy the columns of the table;

Locution Each locution allowed by a system should have an entry in the locution column. The locution is specified in terms of the illocutionary act, followed by the bracketed content variable(s). For example, in a system which has a statement move where the illocutionary act is named “statement” and the move manipulates a single content variable named S_x, the associated locution is denoted *statement(S_x)* in the locution column of the table. If more than one form of the statement move is allowed then a locution is entered into the locution column for each form of the move. For example, if there is a second form of the statement move in which an argument is stated and this is achieved using two content variables, S_x and S_y, where there is a conditional relationship between the two content variables such that $S_x \rightarrow S_y$, then this is denoted by *statement($S_x \rightarrow S_y$)*.

Desideratum The range of legality requirements are specified in terms of earlier events in the dialogue and the state of the players commitments, possibly in relation to the current move.

Past Events This relates to earlier locution events in the current dialogue. Because each locution is assumed to be expressed in turn, t, events which occured in the preceding turn are referred to as events at *t-1* and events which occurred at any point preceding the current turn are referred to as events at *t-n*. For example, if the requirement for a locution states that the preceding turn was a challenge locution then the entry in the desideratum column might be as expressed as *t-1 = Challenge(S_x)*. Likewise, if the requirements state that a previous move was *Challenge(S_x)* then this is denoted by *t-n = Challenge(S_x)*. Furthermore if the requirements state that the challenge move must previously have been uttered by a particular player then this is denoted by a subscript on the move which indicates the player, *H* for the listener or *S* for the speaker as follows: *t-n = Challenge(S_x)$_H$*, which specifies that an earlier move was *Challenge(S_x)* uttered by the listener. Finally, the $\neq$ symbol is used to indicate that a turn did not contain a particular move, for example, *t-n $\neq$ Challenge(S_x)$_H$* indicates that the listener has not uttered *Challenge(S_x)* in an earlier turn.

Commitment State This relates to the current commitment states of the dialogues participants. Commitment stores are denoted by the term *CStore*. If a commitment store is of a named type, such as a set of assertions or a set of mere concessions, the name of the store is indicated in brackets as follows, *CStore(Ass)* for the commitment store containing the set of a players assertions. To indicate the owner of the store a subscript containing the initial of the player is appended, for example, the commitment store containing the set of concessions of the Speaker is written as follows, *CStore(Con)$_S$*.

To specify that a commitment store must contain a particular commitment the set contains symbol ($\in$) and the set does not contain symbol ($\notin$) are used. For example, to specify that the commitment S_x is in the speaker's commitment store the following expression is used,

$\{S_x\}$∈CStore$_S$.

Content: Structure & Relations Requirements might be made that pertain to the structure of the content part of a locution or over the relationships that might hold between specific instances of statement variables. Relationships are argument-theoretic relationships between statement variables such as a requirement that S_x is grounds for S_y in some argument such that $S_x \rightarrow S_y$.

Effects :

Future Events Similar to the nomenclature for past events, future events are denoted using *t+1* for the next turn and *t+n* for some future move.

Commitment Updates Commitment stores are referred to in the same way as for commitment states. Instead of checking the current contents of the commitment stores though updates either add or remove commitments from the stores. To add S_s to the speaker's commitment store the following nomenclature is used, CStore$_S$ + $\{S_x\}$. Similarly to remove commitment the subtraction symbol is applied.

3.3 Individual Games

Even a cursory inspection shows that the rules that comprise the extant games express a limited number of situations and contexts and that the richness of dialectical games stems from the ways in which these notions are combined. This section identifies that range of situations, the objects that they relate to and the ways in which those objects are manipulated through the use of locution attribute tables, completed for each game of interest. Each game is investigated in terms of its locutions, through the completion of a locution attribute table, and through investigation of its wider set of rules and features. Particular attention is paid to any new components that a given game introduces.

3.3.1 Hamblin

Hamblin proposes an initial formulation of rules for the game H. In addition Hamblin provides additional rules, to enable a game to be tailored to suit application to a number of fallacies and situations, and an extended commentary on the features of dialectical games in general [44, ch. 8]. The features and components that are introduced include the following;

1. Rule Action - Whether the rule serves to prescribe, prohibit, or permit an action.

2. Languages - The object language and rule language.
3. Locutional From - The typical form that a given locution can take in terms of the relationship between its performative verb and its content.
4. Locutional Action - The kind of effect that a locution has in terms of whether it manipulates the contents of artifact stores or specifies a given dialectical progression.
5. Types of Dialogical Commitment - Explicit, Implicit, Potential, Suspended.

Hamblin identifies the ways that rules can act in a dialectical game. Rules can be used to prescribe, prohibit or permit a given act. The acts that rules relate to are the linguistic acts that the players can perform. The rules may specify that certain acts are directed towards certain players or are conditional upon particular features of the dialogue history. Hamblin suggests that permissive rules be avoided and frames H within the convention that everything not specifically prohibited is permitted. It should be noted that a rule which prescribes a particular locutional-act can equally be expressed as a rule that prohibits everything but that act.

Hamblin identifies two languages which dialectical games are concerned with, the *object-language* and the *rule-language*. The object-language is the language of expression used by the speakers during the dialogue, and the rule-language is the language used to state the rules of the system. Hamblin uses the notion of rule-languages to identify attributes which can be ascribed to particular games. For example, Hamblin defines a system as *rule-consistent* if the rules are formulated such that no act is simultaneously prohibited and prescribed. Additionally Hamblin identifies a number of characteristics which can be ascribed to formal dialectical games. The rules of a *semantically consistent* system are such that a player cannot be forced to utter a contradiction, a *semantically unforced* system is such that no player can be forced to utter a non-tautological statement, and a *semantically open* system is such that a player cannot be forced to utter any given statement whether that statement is tautological or not. Hamblin's describes H as both rule consistent and semantically open. The rule consistency of H is self-evident on examination of the rules, there is no rule that simultaneously prescribe and prohibits a particular act. H is semantically open because the players are able to utter 'No commitment' for any statement at any time and thus avoid all commitment.

At the highest level of representation H consists of two players, named *Black* and *White*, who take turns to utter locutions. The categories of locutions that are allowed are listed in the locutional rules. Locutions are expressed in the form of a locution name and a number of arguments which are supplied to the locution in the form of statement variables. All locutions in H have an arity of either 1 or $n{:}n{>}0$, meaning that a locution either takes a single argument or takes one or more arguments. Although it is not listed individually in the locutional rules the locution 'Statements S, T' is distinct from the 'Statement' locution both in terms of syntax and semantics. For this reason the 'Statements' locution should really be included as a distinct locution in the locutional rules of H. In addition to the basic locutions listed in the locutional rules, syntactical rule S1 allows a compound locution which is formed from the 'No Commitment' and 'Why' locutions. It is not specified that the same content must be supplied to both members of a compound locution although Hamblin's example dialogue shows the same statement variable associated with both members locutions of the compound locution. All of the locutions of H fit into two overlapping groups, those which have some sort of commitment effect and those which specify some form of responsive locution. These are two of the basic rule actions that Hamblin identifies in the general discussion of rules and are formulated to prescribe particular behaviours and outcomes given particular actions.

The 'Statement', 'No Commitment', 'Question', and 'Why' locutions all incur commitment on the part of the speaker. When a participant incurs commitment directly of their own volition then that commitment is *explicit*. Excepting the 'No Commitment' locution which only alters the speaker's commitment, the 'Statement', 'Question', and 'Why' locutions also incur commitment on behalf of the listening party. This form of commitment where a player causes the addition of a commitment to another player's commitment store is *implicit* commitment . The resolve locution does not specify any commitment effect. By enabling a player to incur commitment on behalf of their opponent, H ensures that both parties are fully engaged in the dialogue and that each party must be active to discharge any commitments that their opponent incurs for them to which they do not wish to be committed. Locutions can thus be classifed in terms of their commitment effect, whether and how the locution incurs or retracts commitment on behalf of any of the participants. The exact commitment effect of any given locution is not always atomic. Contrasting the effects of the 'No Commitment' locution and the 'Statement' locution illustrates this point. The 'No Commitment' locution removes a specified commitment from the speaker's commitment store. Once the locution is uttered the commitment operation takes place and that player's commitment store is updated. Locutions uttered in subsequent turns do not affect the effect of the commitment operation associated with the ‘No Commitment’ locution. From the perspective of the turn in which the ‘No Commitment’ locution is uttered, the effects of the locution are *atomic*. However in the case of the ‘Statement’ locution the exact commitment operation carried out as a result of uttering the ‘Statement’ locution depends to some degree upon the response of the listening player in the next turn. Uttering a ‘Statement’ locution incurs commitment immediately on behalf of the Speaker and possibly incurs commitment on behalf of the listener. If the listener utters a denial, or a ‘No Commitment’ locution in the subsequent turn then the commitment is not included in the listener’s commitment store. In this case from the point of view of the listener, the commitment is a *potential* commitment and the commitment operations of the ‘Statement’ locution are *non-atomic* with respect to the turn in which the originating ‘Statement’ locution was uttered. The existance of non-atomic locutions means that a Speaker cannot know in advance what the exact commitment effect of such a locution will be. Hamblin does not discuss these ‘potential’ commitments explicitly but introduces them within the rules for the ‘Statement’, ‘Statements’, ‘Question’, and ‘Why’ locutions.

A fourth form of commitment, in addition to explicit, implicit, and potential commitment, is introduced within the rules of H. This form is the *suspended* commitment which occurs when addition of a commitment to a players commitment store is delayed pending the outcome of a given dialectic sequence. The particular dialectical sequence which yields a suspended commitment occurs when the ‘Statement’ and ‘Why’ locutions are uttered. If a ‘Statement’ locution is replied to with a ‘Why’ locution then the implicit commitment associated with the ‘Statement’ is suspended until the Hearer either accepts or rejects the reasons offered for the initial statement. At this point the suspended implicit commitment is either added to the Hearer’s commitment store, if the the Hearer has accepted the reasons, or the commitment is discharged, if the Hearer does not accept the reason. The main difference between the potential and suspended commitments is that a given commitment can be suspended indefinitely if the Hearer repeatedly requests justification by uttering a ‘Why’ locution. It is only at the end of the Statement-Why-Statements sequence that the suspended commitment is either accepted or rejected, however a number of intermediate commitments may also have been suspended during the sequence, all of which may or may not be incurred as commitments depending on the outcome of the sequence. A potential commitment however is either explicitly rejected or implicitly accepted during a single two-turn round between the players.

The final form of commitment that can be distinguished from Hamblin’s treatment of formal dialectic is axiomatic commitment. Hamblin suggests, but does not explicitly formulate, rules for the establish-ment of axioms which are present in both players commitment stores prior to the beginning of a dialogue. Axioms cannot be removed from the player's commitment stores and only affect the dialogue by restricting the players from asking a 'Why' locution if the subject of the locution is an axiom or a substitution instance of said axiom.

H both allows and positively encourages retraction of commitment. The encouragement comes from the notion of implicit commitment, without a participant actively retracting commitments their opponent would be able to burden them with any and all commitments. A player can remove particular commitments from their commitment store by uttering the 'No Commitment' locution. No provision is made for adjusting commitment stores in light of any retractions from them. Hamblin does mention that there may be consequential alterations after a retraction but does not elaborate. On a more general level it is suggested that specific formulations of retraction rules should be created for specific instances of formal dialectical games. Hamblin recognises that an adequate dialectical system can operate without requiring consistency amongst the contents of the participants commitment stores and assumes that a rational system can be sustained by allowing participants to recognise and remedy inconsistencies when they are pointed out. The approach that H takes therefor is to separate the notions of consistency of commitments from the notion of retraction of commitment so that retracting a commitment does not retract anything else, not even logically equivalent commitments.

Locutions can also be classified in terms of their dialectical status within the system. A locution can specify a particular set of valid responses. Whether or not a locution makes this specification indicates the dialectical status for that locution and whether that locution has a place in any given subsequent sequence of utterances. The 'Statement' and 'No Commitment' locutions do not prescribe a particular locutional response, the speaker can utter any location in the next round following a 'Statement' or 'No Commitment' so long as the selected locution is legal. Such locutions can be described as *dialectically open* to indicate that they do not prescribe any responsive locutions. Conversely the 'Question', 'Why', and 'Resolve' locutions are *dialectically closed* and prescribe a restriction over the set of legal responsive locutions. Whether a locution is dialectically open or closed indicates the status of the locution within the dialectic system and the specifics of any locution's dialectical status indicate abstractly how that locution can contribute towards a dialogue's profile [59]. The dialectical status of any given move within a game contributes directly to whether the game itself is dialectically closed, open or partially-open.

For the basic H system, the tri-partite layout of H into locutional rules, syntactical rules, and commitment-store operations appears both simple and sufficient but this is deceptive because it hinders the comprehension of each locutions legality requirements and resultant effects which are spread out through the specification of the system. For example, to determine the legality of uttering the 'Why' locution requires that at least three distinct points in the rules have to be examined. Although in the case of H it is useful to separate the syntactical rules and commitment-store operations, the split quickly begins to break down as witnessed by the unwieldy formulation of operations for the 'Statement' locution. When a 'Statement' is followed by a 'Why' locution the commitment effects for the 'Statement' locution are altered because of the non-atomic, inter-locutional commitment effect of the 'Statement' locution so there is an interplay between the legality requirements of a locution, the dialectical relationships of the locution, and the resultant commitment-effects of the locution all of which must be comprehended and balanced to determine that the locution is both legal and desireable. This suggests that listing the two types of rule separately is not the best way to represent the rules of a system as the locutions begin to include more legality requirements.

Hamblin describes H as a "Why-Because system with questions" and postulates that the simplest feasible system, a "Why-Because system", would omit the question locution, whereas a simple "question and answer system" could be built by omitting the why locution. This suggests that Hamblin conceives of a range of formal dialectical games, each formulated to tackle a particular problem. To achieve this Hamblin introduces a range of additional rules which are suggestions for ways to incrementally improve H so that it hinders various forms of sophistical behaviour and prohibits the commission of certain fallacies.

Additional rule A1 is formulated to ensure that questions are not used to make statements by requiring that the arguments supplied to a 'Question' locution are already commitments of both participants. The reasons for this are twofold, firstly because making a statement has a characteristic commitment effect and questions could be considered as different to statements and so should not have the same commitment effect, and secondly because rule A1 is an example of the type of rule that might be required to avoid committing the fallacy of many questions. Because questions and statements are similar in H in terms of their commitment effects another addition is suggested in A2 to give statements a particular character by requiring that they are only used to impart information. If a statement is already a commitment of the hearer then restatement does not introduce any new information to the dialogue. Rule A3 introduces a view of questions as inquiries so that a question can only be asked if the speaker has not previously established the subject of the inquiry. Rule A4 is a weakened version of rule A2 to enable statements to be used to garner admissions such that any statement uttered is neither a commitment of the speaker or of the hearer. Rule A5 is the corresponding weakened version of rule A3 to enable questions to be characterised as admission elicitators. If a question is asked under rules A5, in the guise of an inquiry for information and as an admission elicitator, then the answer cannot be amongst the commitments of the speaker, because of the inquiry aspect, but neither can it be in the commitment store of the hearer, because of the elicitation aspect. Correspondingly under rule A4 a statement can occupy both an admission role and as a means to impart information, so a statement can only be made if it is not in the speakers commitment store, because that statement has already been elicited, and not in the hearer's commitment store because it is information the hearer already knows. Hamblin recognises that the characterisation of argumentation in H is rudimentary. Rule A6 is used to give the 'Why' locution a character such that the hearer is invited to convince the speaker. This requires that the participant to whom the 'Why' is directed be committed to the statement in question and that the participant uttering the 'Why' not be committed to the statement at issue. To avoid *petitio principii* rule A7 is formulated so that participants can only develop arguments which follow from the existing commitments of both participants. In relation to rule A7 which is particularly strong, because it only allows arguments to be developed one step at a time, rules A8a and A8b are presented which provide a different approach to the *petitio* fallacy. Instead of requiring that all elements of an argument be commitments of both participants, rules A8a and A8b aim to establish only that the most basic premises are shared commitments and that whenever possible the players question each other using 'Why' locutions and only ever accept a commitment as the result of an argumentative exchange. A8a is a stronger instantiation of this approach than A8b and requires that any commitments of the hearer that are not in the speakers commitment store must be questioned with the 'Why' locution whenever possible. A8b merely weakens A8a to include only commitments of the hearer which are part of a chain of statements-premise arguments. Rule A9 introduces a new form of commitment, the concession, which recognises the difference in status between explicit and implicit commitments. The commentary associated with A9 suggests but does not formulate rules to account for different behaviours of the 'Why' locution in light of the different burder of proof associated with asserting a commitment as opposed to merely conceding a commitment.

Table 3.2 shows the locution attribute table for the basic rules of H. The features of H are summarised as follows;

Commitment Stores: Each player has a dynamic commitment store

Commitment Store Contents: Commitments may be added or retracted from nominated commitment stores. Commitment stores may be inspected to identify their current contents

Locutional Content Form: Requirement can be made that the content of a location is expressed as a disjunction, a conditional, or a negation of statement variables.

Locutional Form: A location can consist of a performative verb associated with either a single statement variable, exactly two statement variables, or multiple statement variables dependent upon the location. Support for partnered locutions in which two specified locutions are allowed to be expressed in the same turn by the same player contrary to the single location per turn rule.

Locutional Range: Explicit support for five location types

Participants: Supports two players

Responses: For certain locutions a set of mandatory alternative responses is specified.

Turn Structure: One location per turn with a strict alternation of players between turns

3.3.2 Mackenzie

The games DC and DD contribute a number of new component states, requirements and effects that can be used to specify a dialectical game. The commitment effect of some moves such as the *Statements* move is expressed in terms of earlier moves that have taken place in the immediate previous turn. Specifically, the commitment effect of the *Statements* move is dependent upon the move played in the last turn. DC also introduces the notion of commitment with respect to a particular location rather than commitment just to the statement content of the move. This is exemplified by the "commitment to challenges" that allow players to incur a location into their commitment store rather than just the content portion of the location. The range of states that can be used to decide whether a move is legal is greater in DC compared with H. The legality of a move can depend upon the logical relationships between relevant commitments in the players commitment stores, for example one of the requirements of the itResolve move is that the content of the move is a conjunction of inconsistent statements from the listener's commitment store. This kind of requirement enables a game to prescribe that a move is legal if its content is of a particular form, for example that the content is a conjunction of statements, or consists of established statements

Locution (in turn *t*)	Desideratum	Effects
Statement(S_x)		CStore$_S$ + $\{S_x\}$ CStore$_H$ + $\{S_x\}$ CStore$_S$ – $\{\neg S_x\}$ CStore$_H$ – $\{\neg S_x\}$
Statements(S_x, S_y)	*t-1* = Why(S$_x$)	CStore$_S$ + $\{S_x\}$ CStore$_S$ + $\{S_y\}$ CStore$_H$ + $\{S_x\}$ CStore$_H$ + $\{S_y\}$
Denial($\neg S_x$)		CStore$_S$ + $\{\neg S_x\}$ CStore$_H$ + $\{\neg S_x\}$ CStore$_S$ – $\{S_x\}$ CStore$_H$ – $\{S_x\}$
No Commitment(S_x, S_{x+1}, ..., S_{x+n})		CStore$_S$ – $\{S_n\}$
Question(S_x, S_{x+1}, ..., S_{x+n})		CStore$_S$ + $S_x \vee S_{x+1} \vee ... \vee S_{x+n}$ CStore$_H$ + $S_x \vee S_{x+1} \vee ... \vee S_{x+n}$ Denial $\neg(S_x \vee S_{x+1} \vee ... \vee S_{x+n})$ No Commitment($S_x \vee S_{x+1} \vee ... \vee S_{x+n}$) *or* No Commitment(S$_x$, S$_{x+1}$, ..., S$_{x+n}$) *or* Statement(S$_x$) *or* Statement(S$_{x+n}$)
Why(S$_x$)		CStore$_H$ + $\{S_x\}$ Denial($\neg S_x$) No commitment(S$_x$) *or* Statement(S$_{x+1}$): $S_{x+1} \equiv S_x$ *or* Statements(S_x, S_y): $S_y \rightarrow S_x$
Resolve(S$_x$)		No Commitment (S$_x$) *or* No Commitment ($\neg S_x$)
Compound: No Commitment(S_x, S_{x+1}, ..., S_{x+n}) + Why(S$_x$?)	the conjunction of *desiderata* for each elemental locution of this compound locution	the conjunction of *effects* for each elemental locution of this compound locution

Table 3.2: Locution Attribute Table for H

that contribute towards a particular state of the game components, for example that the content is a conjunction of statements all of which are in the listener's commitment store.

DC was developed in response to Woods and Waltons use of a game formed from H plus the additional rules A6 and A7 [144] to explore *petitio principii*. Woods and Walton identify cumulativity as a crucial factor involves in the occurrence of the *petitio* fallacy in a dialectical context. Cumulativity in this context is in regard to values that the uttered propositions are claimed to possess. A system is defined as cumulative with respect to a given element if once that element is established it remains so. For example a system which is cumulative with respect to a players commitments requires that once a player incurs a commitment they can no longer retract that commitment. The system that Woods and Walton examine is identified as non-cumulative and not prohibitive of circular dialogues. IN response Mackenzie developed DC which is described as non-cumulative in the Woods and Walton sense but prohibits *petitio*. In addition Mackenzie introduces a modified version of DC named DD which replaces the CR_S rule with a new rule CR_{SS}. Whilst DC is described as prohibiting *petitio* and non-cumulative, the modified system named DD is described as "non-cumulative in an even stronger sense"[70] because whilst DC was non-cumulative with respect to statements it was cumulative with respect to locutions, specifically the challenge locution. DC is non-cumulative with respect to both statements and locutions

Prakken identifies Mackenzie's DC as "historically influential, mainly for its set of locutions" [98]. DC has thus been subsequently used as the basis for a number of dialectical games, either through amendments to the original DC rules or through adoption of the DC locutions. The dialectical game of Moore and Hobbes [86], and Yuan's game DE [148] are both games based on amended versions of DC. Yuan proposes DE formulated for use in human-computer debate. This incorporates Walton and Krabbe style commitment stores similar to those found in the game PPD_0 [138]. Two types of commitment are identified, the assertion and the concession, and a commitment is classified depending upon the way in which the commitment is incurred when a game move is played. The conditions under which a move is made set out which type of commitment is incurred as an effect of the move. Other than the incorporation of different commitment store models, DE differs little from DC in terms of the range of components and artifacts which the games rules manipulate.

Table 3.3 shows the locution attribute table for Hamblin's game DC. The features of DC are summarised as follows;

Commitment Stores: Each player has a dynamic commitment store which is initially empty

Commitment Store Contents: Commitments may be added or retracted from nominated commitment stores. Commitments with respect to the statement variables that form locutional content. Commitment to locutions specifically the challenge locution. Combinations of commitments that form immediate consequence conditionals can be identified along with their constituent elements. Commitment stores may be inspected for presence or absence of a given commitment.

Locutional Form: A locution consists of a performative verb associated with a single statement variable. Support for partnered locutions in which two specified locutions are allowed to be expressed in the same turn by the same player contrary to the single locution per turn rule.

Locutional Range: Explicit support for five locutions

Participants: Supports two players

Responses: For certain locutions a set of mandatory alternative responses is specified.

Turn Structure: One locution per turn with a strict alternation of players between turns

3.3.3 Walton

Walton makes a number of contributions to dialectical game research in [133]. These include extensions of existing dialectical game components, new components, and consideration of the way a game is played. Other work by Hamblin, Mackenzie, and Woods and Walton was more concerned with how the rules of a game either prohibit or permit certain behaviours. In contrast, Walton here begins to investigate the notion of terminating states and winning conditions and introduces the concept of a win strategy, a dialectical path through the dialogue leading to the player's thesis. Although this notion is important to analysis of dialogues in terms of how a player reaches their goal within a given game, it is not suggested how such a player would actually contstruct such a win strategy. Walton's other contributions in the context of CB are the veiled commitment store and point scoring.

Whilst CB is formulated as a minimal ruleset for investigating strategic game playing other issues are examined also. CB+ for example is formulated to regulate inconsistency in the players commitments. The type of consistency that the '+' rule regulates is that which Walton identifies as *ambivalence* and represents "a kind of insincerity or irregularity in playing the game". It occurs when a player incurs

Locution (in turn *t*)	Desideratum	Effects
Statement(S_x)	$\{S_x\} \notin$CStore$_S$ *and* $\{S_x\} \notin$CStore$_H$	CStore$_S$ + $\{S_x\}$ CStore$_H$ + $\{S_x\}$
Statement($\neg S_x$)	$\{\neg S_x\} \notin$CStore$_S$ *and* $\{\neg S_x\} \notin$CStore$_H$	CStore$_S$ + $\{\neg S_x\}$ CStore$_H$ + $\{\neg S_x\}$
Statement($S_x \rightarrow S_y$)	$\{S_x\} \notin$CStore$_S$ *and* $\{S_x\} \notin$CStore$_H$	CStore$_S$ + $\{S_x \rightarrow S_y\}$ CStore$_H$ + $\{S_x \rightarrow S_y\}$
Defence(S_x)	t-1 = Challenge(S_y) *and* $\{$Why(S_y)$\} \notin$CStore$_H$	CStore$_S$ + $\{S_x\}$ CStore$_S$ + $\{S_x \rightarrow S_y\}$ CStore$_H$ + $\{S_x\}$ CStore$_H$ + $\{S_x \rightarrow S_y\}$
Withdrawal(S_x)		CStore$_S$ – $\{S_x\}$
Question(S_x)		t+1 = Statement(S_x) *or* t+1 = Statement($\neg S_x$) *or* t+1 = Withdrawal(S_x)
Challenge(S_x)		CStore$_S$ + $\{$Why $S_x\}$ CStore$_S$ – $\{S_x\}$ CStore$_H$ + $\{S_x\}$ t+1 = Withdrawal(S_x) *or* t+1 = Resolve($S_y \rightarrow S_x$) *or* t+1 = Defence(S_y)
Resolve($S_x \wedge S_{x+1} \wedge S_{x+n}$)	((t-1 = Withdrawal(S_{x+1}) $\vee$ t-1 = Challenge(S_{x+1})) $\vee$ $S_x =_{form} S_2 \rightarrow S_3$: $S_2 \in$CStore$_H$) $\vee$ $S_x \in$CStore$_H$	t+1 = Statement(S_y)
Resolve($S_x \rightarrow S_y$)	$S_y \in$CStore$_S$	

Table 3.3: Locution Attribute Table for DC

commitment to a statement but later retracts that commitment when it becomes apparent that maintaining the commitment may prove to be a liability. What is interesting about this system is that instead of prohibiting undesirable behaviour outright, the rules specify a sanction that is imposed if the player exhibits the undesirable behaviour. In this case the undesirable behaviour is when a player indicates that they are not committed to a statement which follows from their existing commitments. Instead of being prohibited, the player is allowed to engage in this behaviour but at the risk of losing their current score of points which might result in losing the dialogue outright.

The veiled commitment store is introduced in CBV and is the first time that a private commitment store is introduced into a dialectical game. Such a private commitment store is used as a technique for consistency maintenance between the commitments incurred during a dialogue and those that a player may have incurred outside the dialogue. For example if the contents of a private commitment store are established prior to a dialogue then the rules may require that all publicly incurred commitments during the subsequent dialogue are incurred consistent with the contents of the private store.

Table 3.4 shows the locution attribute table for Walton's game CBZ. The features of CBZ are summarised as follows;

Commitment Stores: Each player has a dynamic commitment store and a dynamic veiled commitment store

Commitment Store Contents: Check the content of players commitment stores for the existance of particular commitments. Identify immediate consequence conditionals within a commitent store

Dialogue Result: Dialogue is "lost" by a player who breaks the rules. Dialogue is won by player whose initial thesis is included in opponents commitments. Dialogue is drawn otherwise.

Dialogue Termination: Dialogue lasts for a predetermined number of turns. If both veiled commitment stores empty then the dialogue can terminate.

Locutional Form: Locutions may marry a single performative verb with either a single statement variable, a pair of statement variables, or multiple statement variables.

Locutional Range: Explicit support for five locutions

Participants: Supports two players

Responses: For certain locutions specify a set of mandatory alternative responses.

Turn Structure: Player utter one locution per turn.

Locution (in turn *t*)	Desideratum	Effects
Statement(S)		CStore$_S$ + {S}
		If {¬S} ∈ LightStore$_S$ then CStore$_S$ + CStore$_S$ + {¬S}
		If form of S is a conditional and the antecedents of S are in CStore$_H$ then CStore$_H$ + {S}
Withdrawal(S)	S is not the consequent of an imc	CStore$_S$ + {S}
		If {S}∈ LightStore$_S$ then t+1 = resolution(S, ¬S)
Question(S)		t+1 = Statement(S) or t+1 = Statement(¬S) or t+1 = Withdrawal(S)
Challenge(S)		CStore$_S$ + {Why S} t+1 = Statement(T) or t+1 = Withdrawal(S)
Resolution(S, ¬S)	{S}∈CStore$_H$ or {¬S}∈CStore$_H$	t+1 = Statement(S) or t+1 = Statement(¬S)

Table 3.4: Locution Attribute Table for CBZ

3.3.4 Girle

Whilst Girle's games are broadly similar to H and DC in terms of the type of requirements and effects associated with each game, they also introduce a couple of refinements to existing components. The most noticeable refinement is the introduction of a wider range of locution types than previous games. This range is because DL3 handles differences between related locutions explcitly whereas other games can handle similar moves implcitly under the same locutiom. DL3 for example distinguishes between various types of statement to differentiate between the logical form of a given statement in the *logical statements*, the context in which a given statement is used whether as an assertion in the form of a *categorical statement* or as an argument relation between statements that express grounds in the form of a *reactive statement*. Conversely, in DC the Statement locution actually covers two related but different locutions, the assertion of some statement and the denial of a statement which are legal under different conditions and have different effects.

The other refinement is similar to the introduction of commitment with respect to locutions introduced by DC. Similarly DL3 introduces the idea of a justification sequence as a form of commitment. Justification sequences are multipart commitments construcuted from individual statements and have the following form;

$$< antecedent, conditional, consequent, challenge - location >$$

Justification sequences enable a player to incur structures into their commitment stores such as < P, If P then Q, Q, Why Q? > which are used to record how particular positions are established during a dialogue.

DL3 makes requirements regarding the structure of knowledge about the domain in which a dialogue takes place. The wh-Question locution requires that for each wh-question there is an associated answer to the question that is placed into the participants commitment stores. The participants then, in subsequent turns, state their position with respect to the wh-question and its answer. This requires an association in the knowledge base between wh-questions and their answers.

Tables 3.5 and 3.6 show the locution attribute tables for Girle's game DL3, the features of which are summarised as follows;

Commitment Stores: Each player has a dynamic commitment store. Commitments may be added to and removed from commitment stores

Commitment Store Contents: Commitments occur in a number of forms, as justification sequences, locutions, statements, and can be distinguished by type. The form of content of a commitment can be identified with respect to whether it forms a conditional, negation or locution. Commitments can be added and retracted and the stores can be checked for the presence or absence of a particular commitment. Identify immediate consequence conditionals, refer and identify constituent parts: conditional, consequent, antecedent. Specify that content has a particular form or state in relation to other game components (e.g. stated grounds should be acceptable).

Locutional Form: Locutions consist of a performative verb associated with a single statement variable

Locutional Range: Explicit support for nine locutions

Participants: Supports two players

Responses: Specify a set of mandatory alternative responses. Specify the form of content of a response in relation to other expressions. Instead of dealing with expressed content as a unit deal with the elements that constitute the unti or hold argument or logical relations to the content.

Locution (in turn *t*)	Desideratum	Effects

Categorical Statement (P)	T-1 $\neq$ Challenge (P)
	{ P } $\notin$ CStore$_S$
	{ P } $\notin$ CStore$_H$
		CStore$_S$ + {P}
		CStore$_H$ + {P}

Reactive Statement (P)	{ Why Q? } $\in$ CStore$_S$
	{ Q } $\in$ CStore$_S$
	{ P } $\notin$ CStore$_S$
	{ P } $\notin$ CStore$_H$
		CStore$_S$ + {<P, if P then Q, Q, Why Q?>}
		CStore$_H$ + {<P, if P then Q, Q, Why Q?>}
		CStore$_S$ – { Why Q? }
		CStore$_H$ – { Why Q? }
		CStore$_S$ + {P}
		CStore$_H$ + {P}

Logical Statement (P)	{ P } $\notin$ CStore$_S$
	{ P } $\notin$ CStore$_H$
		CStore$_S$ + {P}
		CStore$_H$ + {P}

Term Declaration (P)	{ P } $\notin$ CStore$_S$
	{ P } $\notin$ CStore$_H$
		CStore$_S$ + {P}
		CStore$_H$ + {P}

Withdrawal (P)	P is not an IMC
	P is not the conditional of an IMC
		CStore$_S$ – { P }
		CStore$_S$ – {<P, if P then Q, Q, Why Q?>}
		CStore$_H$ – {<P, if P then Q, Q, Why Q?>}
		CStore$_S$ – { Q }

		If P associated with (Qx)Fx then CStore$_H$ – { P } and CStore$_S$ – { (Qx)Fx } and CStore$_H$ – { (Qx)Fx }

		if T-1 = Challenge(P) then
		CStore$_S$ – { Why P? } and
		CStore$_H$ – { Why P? }

tf-Question (P)		CStore$_S$ – { P } t+1 = Statement(P) or t+1 = Statement(not P) or t+1 = Withdrawal(P) or t+1 = Statement of Ignorance(P)

Table 3.5: Locution Attribute Table 1 for DL3

Locution (in turn *t*)	Desideratum	Effects
wh-Question (P)		CStore$_S$ + {wh-question(P)} CStore$_S$ + {wh-answer(P)} CStore$_H$ + {wh-answer(P)} t+1 = Term Declaration(t) or t+1 = Withdrawal(wh-answer(p)) t+1 = Denial(wh-answer(P)) t+1 = Ignorance(P)

Challenge (P)		CStore$_S$ + {Why P?}
		CStore$_H$ + {Why P?}
		CStore$_S$ – {P}
		CStore$_H$ + {P}
		CStore$_S$ – {<P, if P then Q, Q, Why Q?>}
		CStore$_S$ – {<Q, if Q then P, P, Why P?>}
		t+1 = Withdrawal(P) or
		t+1 = Denial(P) or
		t+1 = Resolution Demand(P) or
		t+1 = Reactive Statement(Q) or

Resolution Demand (P)	[0] P∈CStore$_H$ and form(P,conjunction)	t+1 = Withdrawal (Q): Conjunct(Q,P)
	[1] form(P, Q→R) and Q∈CStore$_H$ and imc(R, Q) and t-1 = withdraw(P)	t+1 = Withdrawal(S): Conjunct(S,Q) or t+1 = Statement(R)

Table 3.6: Locution Attribute Table 2 for DL3

3.3.5 Walton & Krabbe

PPD$_0$ uses strict turn progression and allows players to utter multiple locutions per turn. Locution rule 2 includes a six-tuple layout to demonstrate the structure of allowable moves per turn. Each slot in the tuple may contain more than one element of each type and not all slots must be filled. This means that players may make multiple moves per turn so long as the requirements associated with each move are met. Rule 13 suggests that there is a limit to the number of locutions that can be uttered per turn and to the length of each dialogue, however specific rules to this effect are not formulated.

Table 3.7 shows the locution attribute table for Hamblin's game PPD$_0$. The features of PPD$_0$ are summarised as follows;

Commitment Stores: Set of Assertions, Concessions, & Dark-side Commitments for each player

Commitment Store Contents: Ax & Cx are dynamic, can add & remove commitments from Assertions and Commitments. Dark-side stores are static throughout dialogue. Check whether a particular store currently contains a given commitment. Identify initial thesis

Responses: For some locutions, specify the set of mandatory responses

Dialogue History: Require that specific moves have been played in the past

Dialogue Result: Determined by state of commitment stores at completion. Win-lose-draw state assigned to players

Inter-Turn States: Require that moves be played per turn to bring about a particular state before the next player takes their turn

Locutional Form: Locutions consist of a performative verb associated with either a single statement variable or two statement varibles for specified locutions. Require that two statements are explicit contradictories when used as content of a locution.

Locutional Range: Explicit support for nine locutions

Participants: Two players: white & black

Turn Structure: Multiple locutions can be uttered per turn. Length of turns and dialogues is restricted according to a defined number of tokens (however this number is undefined)

Locution (in turn *t*)	Desideratum	Effects
Assert(S_x)		CStore(Ass)$_S$ + $\{S_x\}$ CStore(Con)$_S$ + $\{S_x\}$
Concede(S_x)	$S_x \in$CStore(Con)$_S$ *and* ($S_x \in$CStore(Ass)$_H$ *or* (t-1 = Request(S_x) *or* t-1 = Serious(S_x)))	CStore(Con)$_S$ + $\{S_x\}$
ElemArg($\Delta \rightarrow S_x$)	$\Delta = \{S_1 \wedge ... \wedge S_n\}$ *and* $S_x \notin$ CStore(Con)$_H$ *and* t-n = Challenge(S_x) $\forall S_x \in \Delta$: $S_x \notin$CStore(Con)$_H$	CStore(*Ass*)$_S$ + $\{S_1 \wedge ... \wedge S_n\}$ CStore(Ass)$_S$ + $\{S_1 \wedge ... \wedge S_n \rightarrow S_x\}$ CStore(Con)$_S$ + $\{S_1 \wedge ... \wedge S_n\}$ CStore(Con)$_S$ + $\{S_1 \wedge ... \wedge S_n \rightarrow S_x\}$ t+1 = Concede(S_x) *or* t+1 = Challenge(S_x)
Request(S_x)	($\nexists S_x$: $S_x \in$ CStore(Ass)$_S$ *and* $S_x \in$ CStore(Con)$_H$) *and* $S_x \notin$ CStore(Con)$_H$	t+1 = Concede(S_X) *or* t+1 = NoCommitment(S_x)

Serious$_1$(S_x)	t-1 = NoCommitment(S_x) *or*
	t-1 = Challenge(S_x)
	$S_x \notin$CStore(Dark)$_H$
		t+1 = Concede(S_x) *or*
		t+1 = NoCommitment(S_x)

Serious$_2$(S_x)	t-1 = NoCommitment(S_x) *or*
	t-1 = Challenge(S_x)
	$S_x \in$CStore(Dark)$_H$
		t+1 = Concede(S_x)

Resolve(S_x, S_y)	$S_{x\ contradicts}$ S_y *and* ($S_x \wedge S_y$)$\in$CStore(Con)$_H$	t+1 = NoCommitment(S_x) *or* t+1 = NoCommitment(S_y)
Challenge(S_x)	$S_x \in$CStore(Ass)$_H$ *and* t-n $\neq$ Challenge(S_x)$_S$ *and* $S_x \notin$CStore(Con)$_S$	t+1 = ElemArg($\Delta \rightarrow S_x$) *or* t+1 = NoCommitment(S_x) *or* t+1 = RetractAssertion(S_x)
No Commitment(S_x)	$S_x \notin$CStore(*Dark*)$_S$ *and* t-n = Serious(S_x)$_H$ *and* ((t-1 = Request(S_x) *or* t-1 = Serious(S_x)) *or* $S_x \in$CStore(Con)$_S$	CStore(Ass)$_S$ – $\{S_x\}$ CStore(Con)$_S$ – $\{S_x\}$
RetractAssertion(S_x)		CStore(Ass)$_S$ – $\{S_x\}$

Table 3.7: Locution Attribute Table for PPD$_0$

3.3.6 Bench-Capon

The Toulmin Dialogue Game (TDG) is distinct from the other games discussed thus far in a number of respects, most noticeably its representation but also due to its use of a disinct argument model to underpin the representation of arguments within a dialogue, the use of a new type of artifact store, and its novel turn-taking structure.

TDG makes use of both a state transition diagram to represent the dialectical progression between moves, and a set of rules describing each locution in terms of preconditions, postconditions and completion conditions which in turn describe the state of game components. Whilst other games are represented such that the legality conditions for making, and the resultant effects from playing, a move are spread throughout the rules into the commitment, structural, locutional rules and so on, TDG is concise and well ordered in its representation. For each move the legality of playing it can be easily determined through examination of the associated preconditions. This makes the TDG representation much easier to comprehend and lessens the chance that a requirement or effect may go unnoticed. However, not all of the information about each move is stored in these operationalised rules. The associated STD is essential to determine which responses are allowed at any given point in the game. This means that the specification for TDG is in two parts, one of which is represented by a diagram.

TDG is the first game to explicitly support an underlying argument model and uses Toulmin's layout of argument to this end. In other games the notion of an argument model is left underdeveloped usually requiring no more than that statements can stand in relationship to each other such that a set of statements, P, can act as premises in support of a conclusion, c, such that P→c. The use of Toulmin's layout has informed the selection of locution types allowed in TDG so that each component part of the layout is represented by a distinct locution.

As wells as the basic Hamblin-type commitment store for each player, TDG utilises a shared claim stack onto which statements are pushed or popped in the manner of the standard computer science data structure [58]. The claim stack becomes a means to control the flow of a dialogue as the initial claim is pushed onto the stack and subsequent locutions possibly cause new claims to be pushed onto the stack as the participants explore the underlying arguments relevant to the initial claim. Once all subsequent claims are resolved so the participants eventually return back to the intial claim having explored the underlying arguments with respect to it. Apart from its internal stack structure, the claim stack also differs from the regular commitment store insofar as it represents a shared, public artifact store whereas all stores used in other games have been either individual private or individual public stores. The introduction of the claim stack represents a broadening of the notion of a commitment store into a more general store of artifacts established during the dialogue.

The turn taking structure of TDG is novel, whilst it shares the traditional single move per turn structure of other games, it does not share the same turn progression. TDG utilises three participants, two of whom are locutors within the dialogue and third of which is a referee. Because it has three participants TDG is not strictly within the definition of a formal dialectical game laid down by Hamblin which sets the number to two, however it is within the general definition of dialectical games having a number of participants. There are two roles available to the locutors, the proponent and the opponent roles which are assigned such that if one locutor is in the proponent role then the other locutor is in the opponent role. Control of the dialogue, meaning determination of which player can move next, is assigned depending upon which particular moves are played rather than based upon the mere fact that a move is played so it is somebody else go. This kind of turn progression is termed liberal, in contrast to the strict progression of other games.

Tables 3.8 and 3.9 show the locution attribute tables for Bench's game TDG. The features of TDG are summarised as follows;

Claim Stack: Can push tokens onto the stack, pop them from the stack, inspect token at top of stack, remove tokens from stack, require stack is empty, check a token is on the stack

Commitment Stores: Commitment can be incurred or retracted

Commitment Store Contents: Require that a commitment exists having the form of a conditional (e.g. if D then C), negation (e.g. not C), or combination thereof. Verify that a particular player has incurred a particular commitment

Dialogue Control: Can be assigned to roles, Can require a particular player to be in control. Can set dialogue status to complete

Dialogue History: Require that specific moves have been played in the past

Locutional Form: That content is a conditional or a negation

Locutional Range: Explicit support for sixteen locutions. Locutions consist of a performative verb associated with a single statement variable

Responses: For each move specify what the set of responses must be

Roles: Proponent, Opponent, Referee. Assignment of roles to particular players

Miscellaneous: Each locution has an associated description

3.4 Summary of Components and Features of Dialectical Games

The games that have been examined in this chapter utilise a range of components, each of which can be arranged relative to each other and manipulated in a variety of ways. This section seeks to summarise this range of components sufficient to realise the desiderata of a unified framework for representing and playing such games. It should be noted that none of the dialectical games examined here, with the exception of Bench-Capon's TDG, were developed with the primary aim of supporting a computational implementation. This is evidenced by the sparseness of desiderata that can be identified for each game, for example, the locution attribute table for Hamblin's game H displays no desideratum for any locution other than the *Statements(S_x, S_y)* locution. As a result the aim of this analysis has been primarily to identify the range of components and features, as evidenced by the collated desiderata, that are made use of within extant dialectical games. A seconday aim has been to identify the requirements of a system that can support the implementation of each of the examined games within a unified framework. To achieve both of these aims requires that the components be identified and their limits mapped so as to provide a basis for unification.

A dialectical game is played by setting up its components into a legal initial state and then by manipulating that state according to a set of rules until a desired termination state is reached. The components of a game are, in a ludic sense, the game boards and tokens that the players manipulate by taking their turn to play allowed moves. The remainder of this section will summarise the identified range of components and the ways that extant games allow those components to be interacted with.

The only way that a player can directly manipulate a component is through their making a move during their turn. Turns can be structured in a number of ways but the two main approaches are concerned with the number of moves that a player can make during a single turn and how, once a player has taken their

Locution (in turn *t*)	Desideratum	Effects
Claim (C)	Prop has control of dialogue	t+1$_{Opp}$ = Switch Focus (C) or t+1$_{Opp}$ = OK (C) or t+1$_{Opp}$ = So (C) or t+1$_{Opp}$ = Why (C) Opp has control of dialogue ClaimStack + { C } CStore$_{Prop}$ + { C }
Why (C)	Opp has control of dialogue { C } top of ClaimStack	t+1$_{Prop}$ = Supply Data (C) or t+1$_{Prop}$ = Withdraw (C) Prop has control of dialogue

OK (C)	Opp has control of dialogue
	{ C } top of ClaimStack
		t+1$_{Ref}$ = Current Claim (C) or
		t+1$_{Ref}$ = Rebut (C) or
		t+1$_{Ref}$ = End
		ClaimStack – { C }
		CStore$_{Opp}$ + { C }
		CStore$_{Opp}$ – { $\neg$C }
		ClaimStack – { $\neg$C }
		Referee has control of dialogue

So (C)	Opp has control of dialogue { If D then C } $\notin$ CStore$_{Opp}$ { C } top of ClaimStack	t+1$_{Prop}$ = Supply Warrant (C) Prop has control of dialogue
Presupposing (C)	Opp has control of dialogue { If D then C } top of ClaimStack	t+1$_{Prop}$ = Supply Presupposition(C) or t+1$_{Prop}$ = Withdraw (C) Prop has control of dialogue
On Account Of (C)	Opp has control of dialogue { If D then C } top of ClaimStack t-1 = Supply Warrant (C)	t+1$_{Prop}$ = Supply Backing (C) or t+1$_{Prop}$ = Withdraw (C) Prop has control of dialogue
Supply Data (D)	Prop has control of dialogue t-1 = Why (C) { C } top of ClaimStack	t+1$_{Opp}$ = Switch Focus (C) or t+1$_{Opp}$ = OK (C) or t+1$_{Opp}$ = So (C) or t+1$_{Opp}$ = Why (C) CStore$_{Prop}$ + { D } ClaimStack + { D } Opp has control of dialogue
Supply Warrant (C)	Prop has control of dialogue t-1 = So (C) { C } top of ClaimStack	t+1$_{Opp}$ = Presupposing (C) t+1$_{Opp}$ = OK (C) t+1$_{Opp}$ = On Account Of (C) CStore$_{S}$ + { If D then C } ClaimStack + { If D then C } Opp has control of dialogue
Supply Presupposition (S)	Prop has control of dialogue t-1 = Presupposing (C) { If D then C } top of ClaimStack	t+1$_{Opp}$ = Presupposing (C) t+1$_{Opp}$ = OK (C) t+1$_{Opp}$ = On Account Of (C) CStore$_{Prop}$ + { S } CStore$_{Prop}$ + { If $\neg$S then $\neg$C } ClaimStack + { S } Opp has control of dialogue

Table 3.8: Locution Attribute Table 1 for TDG

Locution (in turn *t*)	Desideratum	Effects

Supply Backing (C)	Prop has control of dialogue
	t-1 = On Account Of (C)
	t-2 = Supply Warrant (C)
	{ If D then C } top of ClaimStack
		t+1$_{Ref}$ = Current Claim (C) or
		t+1$_{Ref}$ = Rebut (C) or
		t+1$_{Ref}$ = End
		Referee has control of dialogue
		CStore$_{Opp}$ + { If D then ¬C }
		ClaimStack – { If D then ¬C }

Withdraw (C)	Prop has control of dialogue { C } top of ClaimStack	t+1$_{Ref}$ = Current Claim (C) or t+1$_{Ref}$ = Rebut (C) or t+1$_{Ref}$ = End ClaimStack – { C } CStore$_S$ – { C } Referee has control of dialogue
Switch Focus (C)	{ C } ¬top of ClaimStack { C } on ClaimStack Opponent has control of dialogue { C } ∉ CStore$_{Opp}$	t+1$_{Opp}$ = Switch Focus (C) or t+1$_{Opp}$ = OK (C) or t+1$_{Opp}$ = So (C) or t+1$_{Opp}$ = Why (C) { C } to top of ClaimStack Opponent has control of dialogue
Current Claim (C)	Referee has control of dialogue { C } top of ClaimStack { C } ∈ CStore$_{Prop}$	t+1$_{Opp}$ = Switch Focus (C) or t+1$_{Opp}$ = OK (C) or t+1$_{Opp}$ = So (C) or t+1$_{Opp}$ = Why (C) Opponent has control of dialogue
End	Referee has control of dialogue ClaimStack = Ø	DialogueStatus = Complete
Rebut (C)	Referee has control of dialogue { C } top of ClaimStack { C } ∉ CStore$_S$ { If D then C } ∉ CStore$_S$ { D } ∈ CStore$_S$	t+1$_{Opp}$ = OK (C) or t+1$_{Opp}$ = Rebuttal (C)

Rebuttal (D)	Player has control of dialogue
	t-1 = Rebut (C)
	{ C } ∉ CStore$_S$
	{ C } ∈ CStore$_H$
		t+1$_{Opp}$ = Presupposing (C)
		t+1$_{Opp}$ = OK (C)
		t+1$_{Opp}$ = On Account Of (C)
		ClaimStack + { D }
		ClaimStack + { C }
		CStore$_S$ + { ¬C }
		CStore$_S$ + { D }
		CStore$_S$ + { If D then ¬C }
		ClaimStack + { If D then ¬C }
		Speaker$_{Role}$ = Prop
		Speaker$_{Role}$ = Opp
		Opponent has control of dialogue

Table 3.9: Locution Attribute Table 2 for TDG

turn, the next player to play is decided upon. The limit for a turn in terms of the number of allowed moves is usually one as evidenced in H and DC for example. PPD_0 however allows players to make multiple moves per turn. Determining how the next player to take their turn is selected occurs in two ways. In the first there is always a strict turn-taking rhythm such that after one player takes their turn, the next player takes theirs and so on, so that the players always take their turns to play in the same order. For example in CB, if there are two players white and black and white plays the inital move then black takes the next turn. In such cases turn taking is deterministic such that for any player at any given turn number it can be determined whose turn it is to play. The other style of turn taking assigns the next player to play based upon the moves that have been played previously. This style of play is in evidence in TDG which has three players. Most turns in TDG are taken by the Opp and Prop players however occasionally, after certain moves have been played by either player, the referee takes a turn.

The essence of a dialectical game is concerned with the uttering of locutions which can have a range of associated performative verbs and content tokens. Although every locution has exactly one performative the number and arrangement of content tokens can vary between locutions. Locutions can also specify that the content tokens themselves express certain statements which have a given logical form or stand in a particular relation to other previously established content tokens. The range of performative verbs is usually related to the context of dialogue that the game is concerned with. Whilst Searle and Vanderveken [116, pp. 182–216] identify a large number of performative verbs in the English language, only a small selection have been explored in the context of dialectical games. Content tokens can range from a single token through a single set of tokens to multiple sets of tokens that stand in relation to each other. Multiple token arrangements have been used to enable resolution request type locutions, where two content tokens are immediately inconsistent, and argument relationships where a set of tokens stand as grounds in relation to a single token which acts as a conclusion. The statement expressed in a token may be required to have an argument theoretic or a logical relationship to statements expressed in other specified content tokens. Given an existing token a locution may require that its content be expressed in the form of a conjunction, disjunction, negation or conditional, or that it is a premise of conclusion of some related argument.

The boards on which a dialectical games are played are basically a set of information stores that are established and manpulated during a game. As the game progresses tokens are moved between these boards. Most games make use of two kinds of board. The first is the transcript of the dialogue itself which stores a record of the players utterances. The transcript is structured as an indexed sequence of turns, beginning at turn 0, such that a new turn is added to the end of the sequence to correspond to each turn taken by a player. The second type of board is the commitment store onto which certain tokens are moved in accordance with the effects of the players moves. Commitment stores may be public, such that the contents are visibile to all players, or private, so that only the owner can investigate their contents. They may also be static, such that their contents are fixed during the game, or dynamic, so that as the game progresses their contents may alter. The most common kind of commitment store is the dynamic public store which can be found in all of the dialectical games examined here. The static private store is found in CB, whereas the dynamic private store is found in PPD_0. A variant on the commitment store is found in TDG which uses a claim stack in addition to the standard commitment store. The main difference with the claim stack is that it is structured internally as a "last-in-first-out" type data structure and is shared between the participants rather than being owned by an individual player.

Individual commitments which form the contents of a games commitment stores have a range of forms. The most basic commitment is commitment with respect to a content token as a expressed in a locution. Some games may also support commitment with respect to locutions themselves. For example DC allows players to incur commitment with respect to challenges. Another form of commitment is the immediate consequence conditional introduced by Mackenzie in [69] which has the form "P and If P then Q so Q". The final type of commitment found in the extant games is the justification sequence used by Girle in [41] which have the form "antecedent, conditional, consequent, challenge-locution" and are designed to enable the commitment store to record particular sequences of locutions between the players.

In order to move the tokens between boards, to utter a locution that is added to the transcript or to incur a commitment with respect to a given token requires that the participants play moves according to the rules. The rules specify the legality of a given locution at each discrete point in the game. The legality of a locution is determined based upon a number of parameters including previously uttered locutions, the contents of the players commitment stores, the allowable form of the content tokens, and the roles that the players occupy.

Once a locution has been uttered it can have two basic kinds of effect, firstly by specifying a set of manipulations to game components, and secondly by specifying a set of mandatory alternative responses. Game component manipulations either move tokens about the boards or assign roles to particular players.

For example the statement locution of DC moves a commitment corresponding to the content token of the locution into the commitment stores of both players. Mandatory alternative responses are a set of locutions that form the set of legal responses a player can make to a given locution. Each response usually specifies a particular form for a responsive locution where the content of the response is in relation to the current locution.

These desiderata are summarised as follows;

Number of moves per turn May be either a single move or multiple moves per turn.

Turn progression Turns may be organised in a strict manner or a liberal manner.

Locution: Performative Performatives broadly identify the kinds of locution that a game supports

Locution: Content Token: Arrangement The content of a locution can contain single or multiple content tokens

Locution: Content Token: Status Logical and argument theoretic relations between tokens

Boards: Dialogue transcripts record the sequence of turns. Other boards store sets of tokens and are either individually owned or shared, have publicly visible contents or private contents, and are either static or dynamic.

Tokens: Commitment to content tokens, locutions, immediated consequence conditionals, justification sequences.

Move Legality: Formulated in terms of prior moves, commitment store contents, form of content tokens, and player roles.

Locutional Effects: Manipulations Move commitments into and out of commitment stores and assign roles to players.

Locutional Effects: Responses Specify a set of mandatory responses that can be played subsequent to a given locution.

3.5 Summary

This chapter has investigated the current state of dialectical game research through a high-level survey of the field as a whole and the analysis of a range of extant games. Individual games have been examined in terms of the components from which they are built, the manipulations to which those components can be subjected, and the locutions that game players can use to directly manipulate those components. The results of this analysis were then collated and presented in the form of a set of desiderata for a unified dialectical game framework.

Chapter 4

The Architecture for Argumentation

WHILST dialectical games are becoming increasingly popular and finding application in a wide diversity of problem domains this popularity has not thus far been translated into tools and methodologies to support development and deployment of such games. The current state of the field uses a mixture of *ad hoc* specificational approaches and individual implementations and requires a period of tool building and consolidation to support the movement towards a field in which games are specified and implemented in a robust and reproducible fashion. This chapter introduces the Architecture for Argumentation (A4A), a framework for the development, representation, and computational implementation of dialectical games. The A4A characterises all dialectical games in terms of turn-taking games in which the dialogue transcript and the players positions are represented by a collection of boards, and in which the players utterances become the tokens of the game that are manipulated about the boards. Because individual dialectical games may use these components in different ways, or may even specify components in addition to the standard components, A4A game schemas are used to specify individual games in terms of their extensible or optional components and the rules for interacting with those components. A software framework implements the meta-game and the game schemas to enable dialectical games to be implemented within a software application. Figure 4.1 shows the relationship between these components.

Additionally this chapter proposes a change in thinking about dialectical games away from consideration of individual games towards a space of dialectical games in which a vast range of games can be represented and from which can be selected the most suitable game in a given context. The shape of this space is delineated by the range, organisation and manipulations of game components, and is itself

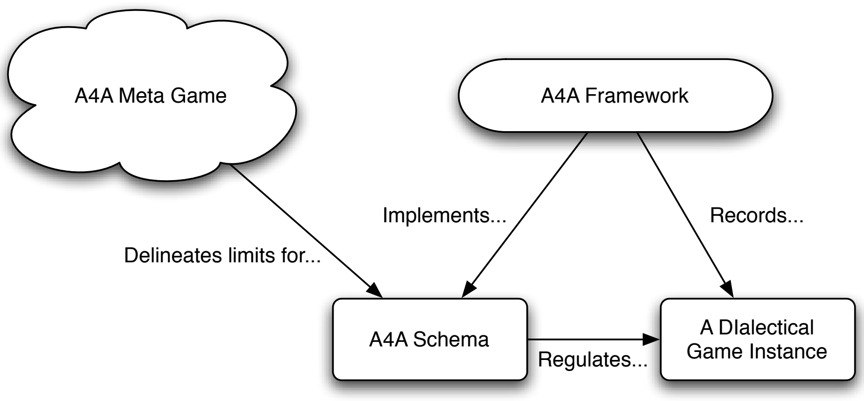

Figure 4.1: The major components of the A4A and their fundamental relationships.

situated within the, currently undefined, space of possible ludic games. The relationship between these game spaces is shown in figure 4.2. The A4A is constructed around a meta-game that delineates a game space bounded by the components of a number of extant games. The A4A game space is therefore a sub-space of possible dialectical games. It provides a strong initial position from which to explore new games by expanding the range of legal components and manipulations.

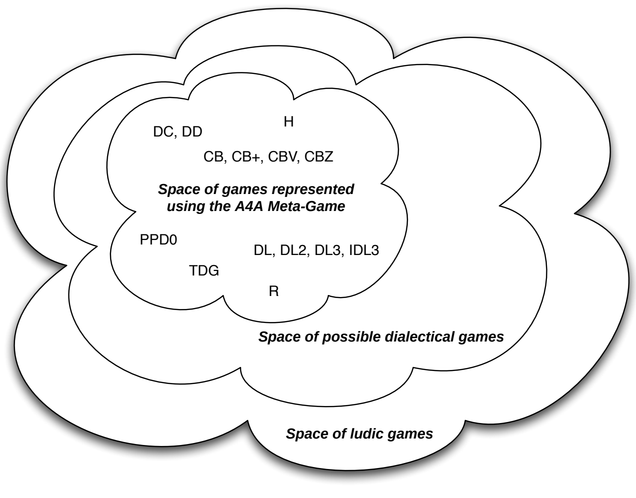
DC, DD
H
CB, CB+, CBV, CBZ
Space of games represented using the A4A Meta-Game
PPD0
DL, DL2, DL3, IDL3
TDG
R
Space of possible dialectical games
Space of ludic games

Figure 4.2: The relationship between different game spaces.

4.1 The A4A Meta-Game

The A4A meta-game is so called because it provides a context in which the range of boards, tokens, players, and moves of dialectical games are abstractly represented. The actual components required by any given dialectical game are specified in that games associated game schema and not all of the components of the meta-game are necessarily required or utilised by any single game. The components of the meta-game can be divided into three groups, those that are standard components and present in every dialectical game, those that are standard components but whose exact form varies from game schema to game schema and are extensible, and those that are optional components, and are present in a dialectical game only if they are specified in the games schema.

The boards represent the space in which a game takes place, and it is the state of the game boards that are used to formulate legality conditions for the players moves. The primary board is the dialogue transcript, a record of the moves made by each player in order of the sequence of turns that have occurred. Other boards correspond to the information stores established during a game as required by the games rules. Whilst H only required the commitment store as a record of incurred player commitments, other games use variations on this concept for a number of purposes. These variations can be grouped together under the term *artifact stores* to account for their variety of purposes, contents, manipulations, and internal structures. A game requires a transcript as its primary board and may optionally specify a range of artifact stores as secondary boards.

The game tokens, the elements that are manipulated between the games boards when a player makes a move, occur in a range of forms. There are two types of standard token, the locution token and the content token. The locution token corresponds to an instantiated move that a player has uttered during their turn. It is the individual locutions that a player utters which are recorded in the transcript. The locution token itself is partly composed from content tokens which represent the statements expressed by the player during the dialogue. The rules of the game specify whether there are any additional tokens and how the tokens are manipulated about the boards.

Players are represented within the meta-game such that each dialogue participant corresponds to an individual player of the game. Roles are assigned to each player initially, and these roles may be assigned or reassigned to other players during the course of the dialogue if the rules so permit. Roles are used to regulate turn taking, player position with respect to the dialogue, and player position with respect to a given argument within the dialogue. The players take turns to manipulate game components in order to bring about a goal state. The game components can only be directly manipulated by the players and such manipulations are only allowed for a player who makes a legal move during their turn. Moves are enacted by the players uttering a locution from the set of legal locutions at that point.

At any given point in a dialogue, the locutions that a player can legally utter can be determined for each active game schema. If a player utters a locution from the set of illegal locutions then this is a case of rule breaking. With the exception of Walton's CBZ, the extant games do not include a formulation of rules to handle such occurrences and even in CBZ the formulation is minimal. This belies the nature of many dialectical games which are formulated for use by human locutors who can "step out of the dialogue" temporarily to deal with rule infractions before returning to the dialogue. This is one deficiency of existing dialectical games when they are moved to the computational domain, they do not include rules to deal with such situations and rely upon new additional rules being developed, which essentially changes the original game, or they require a meta-level control structure in which the game is encapsulated, or else an embedded sub-dialogue. In both cases, merely knowing the rules of the dialectical game is not sufficient to enable heterogeneous agents to interact robustly because they must also know about the extensions to enable them to recover from errors and rule breaking. Even without explicit rules to enforce sanctions against rule breaking, locutions in the set of illegal locutions cannot be played at that point in the dialogue.

The members of the set of mandatory locutions form a disjunction of alternative responsive locutions that a player can make in response to the locutions uttered in earlier turns. For example, specifying that the next move must consist of one locution from a set of alternative responsive locutions. The members of the set of legal locution that are not mandatory are the set of merely legal moves. A player can only utter a merely legal move if the set of mandatory locutions is empty. This means that a player cannot play a merely legal locution unless there are no mandatory alternative locutions which must be uttered first.

For example if $L^M = \{ p, q, r\}$ then the moves available to the current player are $p \vee q \vee r$. L^M is therefore a subset of L^L since all mandatory moves must be legal. The members of L^L that are not members of L^M are the set of merely legal locutions L^{ML} such that $L^{ML} = L^L \setminus L^M$. L^{ML} is therefore the relative complement of L^M in L^L. The members of L^M however take precedence over the members of L^{ML} such that a participant can only utter a locution $l \in L^{ML}$ *if* $L^M = \emptyset$.

Artifact stores are game artifacts, that are manipulated about the games artifact stores according to the positions maintained by the players with respect to each artifact. The game schemas set out the number of artifact stores required, and the rules in each game schema set out how artifacts are related to uttered locutions and are added to, and removed from the artifact stores during a game.

4.2 The A4A Unified Game Schema

The unified game schema is used to represent a set of rules for playing a dialectical game. These rules are formulated in terms of meta-game components and make use of, at a minimum, the standard components although a game schema may also specify a number of the optional components as required to satisfy the needs of the dialogical context.

The unified game schema has a number of aims. These are to provide an extensible way to specify both existing and new dialectical games, to reduce the complexity of representation, to reduce the cognitive burden involved in comprehending new games, to unify the current disparate representational approaches, and to facilitate the development of a single runtime on which a range of games can be played. Using the game schema the development of new games becomes a matter of selecting the components that the game will use and specifying the ways in which those components can be manipulated.

The layout of the game schema is a hybrid of the function oriented groupings of rules found in H and the conditional descriptions used in TDG. The goal is to retain the benefits of the functional grouping presentation whilst also benefiting from the conditional presentation. This is achieved by gathering the game rules into similar groups so that the rules pertaining to the specification of game elements, the specification of global rules, and the specification of individual moves are in their own sections, whilst ensuring that the legality conditions and rational effects of each move are grouped together with the specification of the moves locutional form. The game schema is split therefore into three sections as follows:

Composition
  Name: ID
  Turns: $<$ Structure, Magnitude $>$
  Participants: $\{$ Init$_1$, Resp$_1$, ..., Resp$_n$ $\}$
  Stores: $\{$ Artifact_Stores $\}$

Structure
  $<$ **Name** $>$
  [1] Req:
  [1] Eff:

Interaction
  $<$ **Performative, Content** $>$
  [1] Req:
  [1] Eff:

In the first section the composition of the game is specified, in the second section the structural rules are specified, and in the third section the interaction rules are specified. The compositional section is used to specify the extensible components and optional components used by a specific instance of a dialectical game. Both Structural and Interaction rules are expressed using a head and a body. The head is used to identify the rule and the body is used to define the legal conditions under which the rule is applied and the effects of so applying it. The head of a rule is expressed differently depending upon whether the rule is a structural rule, in which case only the name of the rule is required, or if the rule is an interaction rule, in which case the performative and content form must be specified. The body of a rule is a set of one or more instances where an instance is constructed from a set of requirements and a set of effects. Because each rule may have a number of contexts in which it is applicable each instance of a rule is numbered with a subscript so that a particular instance of a rule may be referenced.

4.2.1 Instances

Instances are a central aspect of rule specification in the game schema and are used both to formulate conditions for when a rule is applicable and when a move is legal, and for describing the effects of those rules being executed. They have the following basic form;

[*Instance Number***] Req:**
[*Instance Number***] Eff:**

Requirements set out a range of conditions that must be met for the rule or move to be legally applied or executed. Each individual condition is separated from the others by a comma and individual conditions are evaluated separately. If any condition evaluates to false then the requirements have not been met. Similarly, the effects set out a range of manipulations of game components that are executed if the requirements are met. For structural rules if the requirements of a rule are met then the associated effects for that instance must be applied to the game's components. However for interaction rules it is required that the associated locution must be played by the current player for the effects to be applied. The requirements for an interaction rule therefore fulfill two roles, the first being to ensure that the current state of the game is sufficient for a move to be legally played, and secondly to ensure that the locution associated with the move is of a valid form.

Ideally instances should be formulated such that if a rule has more than one instance then those instances are mutually exclusive in terms of their requirements. This means that the requirements are specified so that there is no ambiguity as to which instance is applicable at any given point. This approach has the advantages of reducing the number of occasions of unintended effects from the playing of a move but also helps keeps each context in which a moves is played well defined in a single place. Instances may therefore be formulated in one of two manners, as exclusive or inclusive instances. Exclusive instances are formulated such that no two instances can be simultaneously valid. This means that there is no ambiguity about which instance to apply. Inclusive instances may overlap insofar as two or more instances may simultaneously be valid. To avoid any issue of ambiguity with regards to which instance should then be applied, the set of effects to apply is the union of effects from all of the valid instances of the locution at issue.

Requirements are used to set out a verifiable state of game components. These can include requirements over the contents of artifact stores, participant-roles occupation, relations between content tokens, and historical events within the dialogue. Individual conditions can be grouped together to form complex conditions using the conjunction, disjunction, and negation symbols, using parenthesis to indicate scope. The range of conditions supported by the A4A are described as follows;

Content Form This is used either to specify the internal structure of a given content token in terms of its logical form or else to specify a relationship that must hold between two content tokens. This is represented in the schema using the following nomenclature:

| form(token, relation) |

Where token is an established content token and relation may be one from the following set { conjunction, disjunction, negation, conditional, imc }. This may be read as "*the form of token is relation*".

Argument Theoretic Relations A requirement may be required that a content token stand in a particular argument theoretic relation with another content token. This is represented using the following nomenclature:

| argument(first_token, relation, second_token) |

In which first_token and second_token are established content tokens and relation is one element from the following set { premise, conclusion, disjunct, conjunct, antecedent, consequence, conditional}. Requirements of this type may be read as "*first_token is a relation of second_token*".

Artifact Store States This is used to specify the required contents or size of a given store at a particular point in the dialogue using the following nomenclature;

1. artifact $\in$ Store_Name$^{owner}_{timepoint}$
2. artifact $\notin$ Store_Name$^{owner}_{timepoint}$
3. |Store| = *n*

In the first case this requirement states that *artifact* is in a given artifact store at a particular point in the dialogue. Owner is the player who the artifact store belongs to and is specified in the set of participants in the compositions section of the game schema and timepoint is either current, past, or current - *n* where indicates a specific number of turns. In the last case the magnitude of *Store* must be equal to *n*, where *n* is an integer which is greater than zero. This condition is used to specify that a store must be of a certain size. Because artifact stores may be structured as sets, stacks, or queues, additional conditions are required to account for the special access mechanisms that stacks and queues make use of. Stack structure artifact stores may require that a particular artifact is or is not on the top or bottom of the stack as follows;

1. artifact $\overset{Top}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
2. artifact $\overset{\neg Top}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
3. artifact $\overset{Bottom}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
4. artifact $\overset{\neg Bottom}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$

The queue structured artifact store requires that an artifact may or may not be at the head or tail of the store as follows;

1. artifact $\overset{Head}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
2. artifact $\overset{\neg Head}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
3. artifact $\overset{Tail}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$
4. artifact $\overset{\neg Tail}{\rightarrow}$ Store_Name$^{owner}_{timepoint}$

Dialogue Transcript States A rule may require that a particular locution was uttered, possibly by a specific player, at some earlier point in the game.

1. Transcript$^{Player}_{timepoint}$ = Locutions
2. Transcript$^{Player}_{timepoint}$ $\neq$ Locutions

Where *Transcript* is the dialogue transcript, a sequence of turns, *Player* references either a player or a role, and *timepoint* indicates whether the last event, a previous event, or a particular previous event is at issue, and *Locutions* is a disjunction of previously uttered locutions.

Elapsed Turns Because a dialectical game is structured as a sequence of turns it can be necessary to investigate the current turn number.

1. Turn = n
2. Turn $\neq n$

At the beginning of a game the turn number is set to zero. As each turn elapses the turn counter is incremented such that the first turn is at n=1, the second turn is at n=2 and so on. Where *Turn* is the counter which records the number of turns that have elapsed since the game began and n is an integer greater than or equal to zero. In the first instance the requirement expresses that the current turn is equal to n and in the second instance the requirement expresses that the current turn is not equal to n. Identifying the number of turns that have elapsed enables termination states to be checked if the game uses termination rules such as those found in CBZ which specify that a game terminates after a predetermined number of turns.

Role Occupation The participants of a game can occupy various roles which are used to establish which player is in control of the dialogue and can therefore play a turn, and to specify whether a rule is applicable to a given player.

1. role $\in$ Roles^{Player}
2. role $\notin$ Roles^{Player}

Where role is a member of the set of Roles formed from { speaker, listener, proponent, opponent, referee } and Roles^{player} is the subset of Roles assigned to the nominated player. The set of players is defined in the game schema for a given dialectical game but has a minimum of two members.

Schema Activity A dialectical game is played according to a set of rules and these rules are encoded within a game schema. However a given game can be formulated to use several schemas, either serially such that the game moves from one schema to the next, or in parallel so that a number of schemas may be active at any one time. For any given game two sets of game schemas can be identified, the set of active schemas whose rules are currently in force, and the set of legal schemas which defines the set of all game schemas that can possibly become active during the current game if the correct states are encountered.

1. schema $\in$ Schema^{active}
2. schema $\notin$ Schema^{active}

In the first case this condition requires that a given game schema is in the set of active game schemas by Schema^{active}. In the second case this conditions requires that the nominated schema is not a member of Schema^{active}.

Effects are used to update the components of the game and can only be validly applied if their associated requirements are met subject to the distinction between structural and interaction rules. Effects fall into a number of distinct groups;

Artifact Store Updates A defining characteristic of dialectical games is that they manipulate various artifacts about a set of artifact stores. Typically these are commitment stores but other kinds of artifact may be defined. Artifact store updates are used to incur and retract artifacts from nominated artifact stores.

1. Store_Name^{owner} + (artifact)
2. Store_Name^{owner} – (artifact)

For a given store, identified by *Store_Name* and *owner* where the owner is a participant of the game as specified in the game schema. In the first case an artifact is added to the nominated store and in the second case an artifact is removed from the store. Again because artifact stores may be internally structured as sets, stacks or queues the + and – are also used to push and pop artifacts from a stack-structured store, and to enqueue and dequeue from a queue-structured store.

Responses One of the distinguishing features of dialectical games is that they are used to specify particular dialectical sequences such that after a given move there is a certain set of mandatory alternative responses that must be made. This effect is used to specify such a dialectical sequence.

1. $\text{Response}^{Player}_{timepoint}$ = responses
2. $\text{Response}^{Player}_{timepoint} \neq$ responses

Where *responses* is a disjunction of locutions, player is a member of the set of participants as defined in the game schema, and timepoint is one from {next, future, current+n}. In the case of current+n, n denotes a number of turns. In the first case, a set of moves that must occur in response is defined and in the second case a set of moves that must not occur in response. Because a locution may have a range of legal instances, and each instance is indexed, each constituent locution specified in a response set may indicate a particular instance of the locution that the player must respond with. This enables locutions having a particular form or structure, where that form or structure is encoded in an instance of the locution, to be specified as being legal responses rather than that any instance of the given locution is a legal response. To specify a particular instance of a locution within a set of responses a subscript is applied to the locution indicating the particular instance that is required. For example;

$\text{Response}^{Listener}_{timepoint} = \text{<Statement, \{p\}>}_1 \vee \text{<Statement, \{q\}>}_2$

In this case there are two instances of the Statement move and the response must be either a locution conforming to the first instance of the Statement move or a locution conforming to the second instance of the Statement move. A further formulation of responses is the branching response. This is only required in games that allow multiple locutions to be uttered per turn and enables the specification of rules that require that for each elements of a given collection, some response is made. This kind of response is laid out as follows;

[∀ token: condition] Response$_{timepoint}^{Player}$ = responses

This kind of response specifies that for every token satisfying the specified condition, the nominated player in the specified timepoint must utter one from the set of responses.

Role Assignments These effects are used to assign a specific role to a given player. The set of roles available to the game are denoted Roles, and the set of roles assigned to a given player are denoted by Roles^{Player} where player is one of the participants of the game.

1. Roles^{Player} + role
2. Roles^{Player} – role

In the first case a role is added to the set of roles assigned to the nominated player and in the second case a role is removed from the set of roles assigned to the nominated player.

Schema Assignments The set of rules that regulate a given dialectical game is encoded in the *game schema* associated with that game. There are a set of legal schemas associated with a given dialogue. At any given point during a dialogue one or more of the legal schemas may be active. The relationship between the sets of active and legal schemas is such that the set of active schemas is a subset of the set of legal schemas. Schema assignments are used to add or remove a specific schema from the set of legal schemas to the set of active schemas.

1. Schema^{active} + schema
2. Schema^{active} – schema

These assignments are used to define the effects of rules to specify that a given game schema is to be added or removed from Schema^{active}.

Game Status Assignments This assignment is used to set the status of the game to indicate that the associated dialogue is finished.

Game^{Status} = complete

The status of a game is incomplete from its inception until the rules assign the status complete to the game.

Conditional Effects Even once the requirements of a move have been met so that it may legally be played, a move can still have a variety of effects dependent upon the state of the game. Such conditional effects are formulated as follows;

| [*Conditions*] → [*Effects*] |

Where conditions is a formulation of requirements, and effects is set of effects. If the conditions are met then the effects are applied.

In summary, formulations of requirements and sets of effects are very important in the specification of dialectical games and are the mechanism that is used to determine the members of the set of legal locutions at each turn during a dialogue.

4.2.2 Game Schema: Composition

The standard components of the meta-game need not be specified within a game-schema because they are essential to a dialectical game. However in addition to the standard components, an instantiated game must specify the number of players, the turn structure, and any optional components, such as artifact stores, that the game requires. This is because the number of players, the turn structure, and the artifact stores are not fixed for all dialectical games but are dependent upon a particular formulation of rules for specific game. The composition section of a game schema is used to specify these non-standardised components. These are outlined as follows;

Name: ID
Turns: < Structure, Magnitude >
Participants: { Init, Resp$_1$, ..., Resp$_n$ }
Stores: { Artifact_Stores}

The name component is used to identify a schema from all other schemas. Ideally this would be a unique identifier however as a minimum it must be sufficient to uniquely identify the schema within the set of legal schemas for the current dialogue.

The turns component is used to specify how the turns of the game are structured and how many moves may be made per turn. The turn structure may be either strict or liberal. A strict turn structure is defined such that the player, init, is the first player, and is assigned the speaker role at turn t=0. After init has played their turn, the speaker role is removed from init and assigned to the next player in the set of participants. When the list of participants is exhausted the speaker role is reassigned to the init player. This process continues until *Game*^{Status} = *Complete*. In the case of the strict turn structure, the speaker and listener roles are assigned automatically as each player plays their turn. The liberal turn structure is used to facilitate turn taking where speaker-role assignment occurs dependent upon the particular move that is played rather than the mere fact that a role is played. This requires that each move specifies whether or not the speaker role is assigned to another player as a part of its effects. If a schema specifies a liberal turn structure but the moves do not specify any speaker role assignements then it will always the init player's turn and no other players will get a change to make a move. The liberal turn structure therefore requires careful development to ensure it is well structured. As well as specifying how turn taking is structured, the turn component also specifies how many moves may be played per turn. The magnitude of a turn may be either single meaning one move per turn, multiple meaning many moves per turn, or *n* where *n* is an integer greater than one.

The participants component is a set of participant identifiers and is used both to specify the number of players that the game supports and to uniquely identify the players within a game. The minimum number of participants is two, denoted *init* and *resp*. Each subsequent additional player is denoted resp*n* such that *n* increments for each new player starting with *n* equal to two. The player who initiates a dialogue is assigned the init identifier and all subsequent players are assigned identifiers from the set of participants in the order in which they join the game. Participant identifiers are fixed throughout a game such that once a player is assigned an identifier they retain that identifier until the dialogue is complete.

The store component is a set of artifact stores. If a game schema doesn't specify any artifact store then the set is empty. Artifact stores are defined using a four tuple of the following form;

| $<$ Type, Owner, Structure, Visibility $>$ |

In which *Type* identifies the type of the store, *Owner* identifies the participant that the store belongs to, *Structure* identifies the internal arrangement of artifacts, whether the stores is a set, stack or queue of artifacts, and visibility determines whether the contents are available to be inspected at all times. *Type* is a label that the schema uses to identify a type of artifact store such as whether the store is a commitment store as found in most game, an assertion store as found in PPD$_0$, or a claim stack as found in TDG. For each type of store that has a specific owner within the set of participants, *Owner* should be set to reflect that ownership. If a store does not have a specific owner then *Owner* can be left undefined. The specification of *Type* and *Ownership* must be such that every artifact store can be uniquely identified within a schema from all other artifact stores. There are three internal structures for an artifact store, as a turn-indexed set of artifacts, or as a stack or queue of artifacts. The visibility of an artifact store can be set either to public visibility or private visibility. The contents of a public store are visible at all times to all participants and act in the traditional commitment store manner as a public record of artifacts. The contents of a private store however are not visible by default to any of the participants including the owner. The artifacts stored in a private store are only identified through confirmation of the presence of individual artifacts if the schema so allows.

4.2.3 Game Schema: Structure

Structural rules are used to specify certain game states and the effects of those states being brought about. The distinction between structural rules and interaction rules is in their scope. Structural rules have global scope compared with interaction rules which have a localised scope. For example, an interaction rule corresponds to a particular move and the effects of an interaction rule are only valid at the time that the move is played. Conversely all structural rules must be inspected after each turn to determine whether any of them are applicable. Whereas the effects of a move are only applied if the move is played and the requirements of the move are met, the effects of a structural rules are applied as soon as the state specified in the rules requirements are met. Structural rules have the following basic structure;

$<$ **Name** $>$
[1] Req:
[1] Eff:

Where *Name* uniquely identifies the rule. Each rule has a set of one or more instances where each instance is a pair formed from a set of requirements and a set of effects. Each instance is numbered so that it can be efficiently referenced.

Although structural rules could be formulated to carry out many tasks these tasks currently fall into a limited variety of groups. These groups include initiation, termination, progression, adjustment, and sufficiency rules. Initiation rules are used to set out initial requirements and set-up for legal entry into a game controlled by a given schema. Conversely termination rules are used to set out the conditions under which a game can terminate. Progression rules are used to facilitate dialectical progressions, the movement within a game from one game schema to another game schema. A game schema can be developed to control each dialogical context that the participants can enter into. A game begins with a given schema for an initial context of dialogue then as the state of the game develops the context may change, at which point a more suitable schema can be used. This kind of game schema progression is used to enable dialectical shifts and embeddings, the movement from one dialogue type to another and consequently the shift from a sub-dialogue carried out according to the rules of one dialectical game to a sub-dialogue carried out according to the rules of another dialectical game. This type of progression is useful in a multiagent systems context. This is because software agents will likely act under constraints imposed by the amount of memory and computational power available to them, and it is good practise to avoid profligate waste of either. Therefore a communication mechanism that minimises the use of available resources is good. Small, robust, schemas that characterise a well-defined agent interaction and which support the move from one schema to another are useful in this context. Such schemas can form the basis of an efficient and scaleable communication mechanism that utilises resources appropriate to the current interaction, and which alters to account for the changing nature of the each unique, individual interaction.

Such shifts have been explored by Walton and Krabbe in [138] which proposes a type of dialectical embedding of a rigourous persuasion dialogue game within a permissive persuasion dialogue game. Other approaches to dialectical progressions include Reed's Dialogue Frames [103] and McBurney and Parson's "*Agent Ludens*" dialogue game layer model [73]. In their model McBurney and Parsons take the notion of embeddings further and explore a number of abstract combinations of dialogues which involve iteration, sequencing, parallelisation, and embedding. Walton and Krabbe meanwhile identify three modes of dialectical shift [138, pp. 104–108], as well as two speeds at which the shifts can occur [138, pp. 102]. Walton and Krabbe's modes include shifts between dialogue type, internal shifts within the same dialogue type, and shifts of flavour which affect how a given dialogue of a particular type is pursued. The speeds at which a dialectical shift can occur include the deplacement, an immediate shift from dialogue type to another, and the glissement, a gradual overlapping shift.

Walton and Krabbe further identify that such dialectical shifts may be licit or illicit. A licit shift occurs when the shift is constructive and agreed to by all parties. When a shift is concealed or otherwise inappropriate then it is illicit. Walton argues that a characteristic of many fallacies is that they occur where shifts in the dialogue are illicit [134]. As a result, to make the A4A schema applicable to current contexts of fallacy research requires that dialectical shifts and embeddings are supported. A benefit of this approach is that dialectical games can be developed that are concise and targeted towards specific contexts of dialogical interaction. As the context in which a dialogue occurs changes so the participants of the dialogue can progress to a different dialectical game schema tailored to the new context and so utilise the most suitable schema at each point in the dialogue. This avoids the need to develop a single monolithic dialectical game to cover all of the contexts in which dialogue occurs, preferring instead to take a modular approach in which each context has its own associated set of game schemas. Instead a schema can be selected or specified as required to satisfy the needs of the dialogical context.

Adjustment rules are used to specify states of the artifact stores which, if met, require the contents of the stores to be adjusted. For example Walton and Krabbe suggest the idea of stability adjustments [138] to change the contents of players commitment stores in light of their contents. If a given commitment is removed from the commitment store then other commitments which have a dependency upon the retracted commitment may also have to be removed as a result even if the dependent commitments have not been explicitly retracted. Adjustment rules are specified as a way of imposing some non-locutionally specific means to maintain consistency in the artifact stores of a game. Sufficiency rules are similar to adjustment rules and are proposes as a way to ensure that a certain state is reached at the culmination of each turn. If a turn allows players to make more than one move per turn then it may be required of the players that at the end of each turn they have uttered sufficient locutions to bring about a certain state of the game components, e.g. the moves made during the turn have been sufficient. This is based upon a notion found in Walton and Krabbe [138] in which players must, in their turn, utter moves such that, for example, if their opponent has uttered an argument, P, then the speaker must ensure that if they are committed to the premises of P and the warrant of P then they must also concede the conclusion of P. These kinds of rules cannot be added to the instances of any single locution because the conditions might depend upon the utterance of a number of, or combination of locutions. It is simpler to use an inter-locutional state that a player must bring about during their turn, hence the use of adjustment and sufficiency type rules.

4.2.4 Game Schema: Interaction

Interaction rules are used to specify the moves that players can make. Because legally playing a move is the only way for a player to directly manipulate game components it is essential that such moves are clearly, concisely, and consistently presented. An interaction rule specifies the performative associated with the move, the content tokens that can accompany the performative, and a set of instances defining the contexts in which the move can be played. The archetypal interaction rule is structured as follows;

> $<$ **Performative, Content** $>$
> **[1] Req:**
> **[1] Eff:**

Where *Performative* is a performative verb, and *Content* is a set of content tokens. Each move is distinguished by its locutional form which specifies the allowable configurations of performative verb and content token. Because the same location may have a variety of circumstances in which it is legal and a diversity of resultant effects, an instance number is used to reference different instances of the same location. Each instance of a locution is labelled with a number. Such instance numbers begin at zero and are incremented for each new instance.

The tradition in the development of dialectical games has been to have each move correspond to a performative verb or illocutionary act. There are a great many performatives available for each of which a number of moves could be formulated within an expansive dialectical game. Searle and Vanderveken in [116, pp. 182–216] explore thirty two English assertives, seventeen English commissives, twenty four English directives, twenty one English declaratives, and thirteen English expressives, many or all of which could be used as the basis from which to form moves in a dialectical game. However there have generally been no more than a half dozen different performatives included in the rules of the extant games. DC [70] for example, which has been described as possessing an influential set of performatives [98], includes only five moves which are centred around the following acts; statements, withdrawals, questions, challenges, and demands.

The *Content* slot of an interaction rule is used to define the number of tokens that the associated locution requires. Content tokens are variables that correspond to expressions that the players utter in the games object language and are represented by lower case roman letters. Upper case roman letters are used to represent sets of content tokens. Individual content tokens, s, t, u, &c. and sets of content tokens, S, T, U, &c. are used to express the content requirements of a locution so that the interaction rules can be used to express the locutional forms found in the extant games. Typical locutional forms include the following;

1. A locution consisting of a performative allied with a single content token:

   > $<$ **Performative,** $\{$ **s** $\}$ $>$
   > which can express, for example, the *Statement (S)* move of H:
   > $<$ **Statement,** $\{$ **s** $\}$ $>$

2. A locution consisting of a performative allied with exactly two content tokens:

   $<$ **Performative,** { **s, t** } $>$
   which can express, for example, the confronter locution, *Resolve (P, Q)*, found in PPD_0:
   $<$ **Resolve,** { **s, t** } $>$

3. A locution consisting of a performative allied with a set of content tokens, S, where the set is of arbitrary size. Notice that brackets are not required around the set as this form deals with the content as a whole rather than individual members of the set of content tokens:

   $<$ **Performative, S** $>$
   which can express, for example, the retraction locution, *"No Commitment S, T, ..., X for any number of statements S, T, ..., X (one or more)"*, found in H:
   $<$ **No Commitment, S** $>$

Additionally the requirements of a move can be used to specify that a content token has a particular form. This enables the form of moves such as the elementary argument, ΔsoP, found in PPD_0 to be expressed. The elementary argument has a statement, P, that acts as the conclusion and a set of statements, Δ, that acts as the premises. This can be represented in a locution of the following form:

$<$ **Performative,** {**s, t**} $>$
where the performative is "Elementary Argument", and there is a requirement that the content token, s, has the form of a conjunction of statements expressing the premises of the argument, and t is the conclusion.

The final locutional form found in the extant games is that of the compound move. In games that allow only a single move to be made per turn there is sometimes the requirement that this rule be relaxed. For example, in H a *"No Commitment"* locution may accompany a *"Why"* locution within a single turn. Such moves form a compoound from two existing locutions and utilise the existing specifications for each element of the compound. In the A4A schema such compound moves are expressed using the addition sign as follows;

$<$ **No Commitment,** {**s**} $>$ **+** $<$ **Why,** {**s**} $>$

Since both elemental locutions within the compound locution are already defined within the game schema it is not necessary for the body of each locution to be reiterated.

4.3 Example

A pair of dialectical games were developed [140, 141] to investigate the progression from persuasion to negotiation in the context of the fallacy of bargaining. In [138] the fallacy of bargaining is identified as occurring when participants are engaged in a dialogue which starts out as a persuasion type but that at some point during the course of the dialogue an illicit shift occurs from persuasion to negotiation.

The example of the fallacy of bargaining used by Walton and Krabbe involves a government minister of finance who has been caught profiting from certain tax exemptions. The minister argues that those tax exemptions should be allowed temporarily and not be penalised. The minister then goes on to propose to his critics that if they abstain from moving for penalties for the exemptions, then he will not oppose a bill that the critics will benefit from. In this case, instead of satisfying his burden of proof with respect to his position on the tax exemptions, the minister substitutes an offer for an argument, a move which is not permissible in persuasion dialogues. By making an offer during the persuasion dialogue the minister has reneged on his commitment to defend his position, *vis a vis* the tax exemptions, and caused an illicit shift to a negotiation dialogue.

However, the shift from persuasion to negotiation need not always be an instance of the fallacy of bargaining. As Walton and Krabbe recognise, illicit shifts occur when the shift is concealed or inappropriate and a fallacy can occur as a result, If the shift occurs in an open way, and is demonstrated to be appropriate then there is no need to characterise it as fallacious. Where conflicting participants in a dialogue have exhausted their persuasive arguments and are in a position that is unlikely to be resolved through continuation of the persuasion dialogue then it is acceptable for the participants to try some other way to break the deadlock. In an agent situation the failure to reach agreement can be undesirable, requiring that the agents re-plan which is computationally expensive. Given that both participants actually wish to resolve the conflict, which is the reason why they are still engaged in the dialogue at this point, a shift to another type of dialogue enables the participants to continue. If the shift is from a persuasion dialogue to a negotiation dialogue then the participants may be able to reach a practical settlement and so be able to move forward. Interestingly the converse situation, the embedding of persuasion dialogues within ongoing negotiation dialogues has been explored quite extensively by Sycara in relation to the PERSUADER system [126], and by Rahwan [101] in relation to argument-based negotiation in multiagent systems.

Two dialectical games were developed, Persuasion Protocol Zero (PP_0) and Negotiation Protocol Zero (NP_0) [140, 141], to demonstrate two points, firstly that not all shifts from persuasion to negotiation are necessarily illicit, and secondly, that in the context of agent communication, such a shift can not only be licit but that it can enable the participants of a dialogue to proceed towards a practical settlement when they would otherwise reach an impasse. These games also serve as a useful example of some of the features of the A4A. PP_0 is a minimal game for engaging in persuasion dialogues and is represented using the following A4A schema;

Composition
Name PP_0
Turns < Strict, Single >
Participants { init, resp }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >

Structure

<**Initiation**>
[1] Req: Turn = 0
[1] Eff: Schema^{Active} + (PP_0), $\text{Response}^{Speaker}_{Next}$ = <Request, {p}>

<**Progression**>
[1] Req: p∈ CStore^{init}_1, p∈ $\text{CStore}^{init}_{current}$, (q→p)∈ $\text{CStore}^{init}_{current}$, $\text{Transcript}^{resp}_{last}$ = < Reject, {p} >
[1] Eff: Schema^{Active} + (NP_0)

<**Termination**>
[1] Req: p∈ CStore^{init}_1, p∉ $\text{CStore}^{init}_{current}$
[1] Eff: Dialogue_{status} = complete
[2] Req: p∈ CStore^{init}_1, p∈$\text{CStore}^{resp}_{current}$)
[2] Eff: Game^{Status} = Complete

Interaction

< **Request,** {**p**} >
[1] Req: –
[1] Eff: $\text{Response}^{Listener}_{Next}$ = <Accept, {p}>∨<Reject, {p}>∨<Challenge, {p}>, $\text{CStore}^{Speaker}$+(p)

< **Accept,** {**p**} >
[1] Req: $\text{Transcript}^{Listener}_{Last}$ = <Request, {p}>
[1] Eff: $\text{CStore}^{Speaker}$ + (p), $\text{CStore}^{Speaker}$ – (¬p)

< **Reject,** {**p**} >
[1] Req: $\text{Transcript}^{Listener}_{Last}$ = <Request, {p}>
[1] Eff: $\text{Transcript}^{Listener}_{Next}$ = <Challenge, {p}> ∨ <Withdraw, –>, $\text{CStore}^{Speaker}$ + (¬p), $\text{CStore}^{Speaker}$ – (p)

< **Challenge,** {**p**} >
[1] Req: $\text{Transcript}^{Listener}_{Last}$ = <Request, {p} > ∨ <Reject, {p}> ∨ <Defense, {p, q} >
[1] Eff: $\text{Transcript}^{Listener}_{Next}$ = <Defense, {p, q} > ∨ <Reject, {p}> ∨ < Withdraw, –>

< **Defense,** {**p, q**} >
[1] Req: –
[1] Eff: $\text{CStore}^{Speaker}$ + {p}, $\text{CStore}^{Speaker}$ + {q}, $\text{CStore}^{Speaker}$ + {p→q},
$\text{Transcript}^{Listener}_{Next}$ = <Challenge, {p}>∨<Challenge, {q})>∨<Challenge, {p→q}>∨
<reject, {p}>∨<reject, {q}>∨<reject, {p→q} >∨
<accept, {p}>∨ <accept, {q}>∨<accept, {p→q}>

< **Withdraw, –** >
[1] Req: Transcript_{Last} = <Challenge, {p}>∨<Reject, {p}>
[1] Eff: Game^{Status} = Complete

PP_0 enables two players named *init* and *resp* to engage in a persuasion dialogue. Players can make one move per turn, starting with init. The turn structure means that turns proceed automatically, after one player makes their move, the next player has their turn and so on, such that it can be seen from examination of the current turn index which players move it is. The actual moves that are played cannot influence which player is assigned the speaker role in the next turn and thus cannot influence whose turn it is. Each player is assigned an artifact store named CStore. The remaining parameters specify that the store can contain a mixture of commitment types, for example a player can incur commitment to just the content of a move or to the entire move, that the store is a light side store [133] which stores a set of commitments and that the stores are to be shared between sub-dialogues of differing types. PP_0 incorporates three types of global rule. These rules specify the requirements for starting a new instance of a PP_0 sub-dialogue, the requirements for initiating a progression from an instance of a PP_0 sub-dialogue to a new instance of another sub-dialogue type, and the conditions for terminating a PP_0 dialogue.

When a new sub-dialogue of type PP_0 is begun, the initiation rules require only that the very next move, in this case the first move of the new sub-dialogue, must be a request. For a progression to be legal it is required that the player who played the first move of the PP_0 instance still be committed to their initial thesis, that *init* has expressed at least one argument in support of their initial thesis, and that the last move played in the immediate previous turn was a rejection of that initial thesis by the respondent. These conditions establish that a progression is legal at this point in the dialogue, and that the next move may be from the set of moves allocated to the NP_0 system. The current player may elect to continue in the current dialogue without progressing to another dialectic system. For example, the progression rules of PP_0 only establish that a transition is legal, not that it must occur. To actually initiate a progression at this point requires the player to make a legal move from the NP_0 move set according to the initiation rules for NP_0.

PP_0 allows six distinct moves. Each move specification incorporates a formulation of requirements for when the move is legal, and a formulation of effects that must be applied when the move is played. The request move is an utterance of the form "S?", and has no requirements. The effects of playing the request move are that the content of the move is added to the speaker's commitment store and that the legal responses are the accept, reject and challenge moves. The accept move enables a player to agree to a request and is of the form "OK S". Conversely the reject move enables a player to disagree with a request and is of the form "Not S". The challenge move is formulated to enable a player to get justification for a previous request, reject or defence move and is of the form "why S?". The defence move enables a player to defend their challenged position by providing a supporting statement of grounds and by stating an inferential link between the challenged position and the justifying statement. The withdraw move is essentially an utterance of the form “I withdraw from this dialogue”, and the rationale is to allow either player the opportunity to withdraw from the dialogue. If either player determines that the dialogue is unlikely to end successfully then it is more computationally efficient to leave the dialogue cleanly at the first subsequent opportunity rather than continue.

PP_0 only allows a player to manipulate the contents of their own commitment store and does not allow a player to incur commitments in their opponents commitment store. This is achieved through the formulation of effects for each move which only update the commitment store of the speaker. The only moves which incorporate a commitment effect are the request, accept, reject and defence moves. The challenge move does not incorporate a commitment effect, like the commitment to challenges of DC [70], but rather allows the receiver of the challenge to immediately withdraw from the dialogue without penalty. This enables the participants to produce a number of different justifications in response to a challenge by engaging in several iterations of the challenge-defence sequence. This enables some tactical play to emerge in PP_0 persuasion dialogue whereby a player can repeatedly challenge a statement to uncover the underlying justifications for that statement, but if the player is too persistent then their opponent may choose to withdraw from the dialogue entirely. To avoid withdrawal, it is incumbent upon the challenging player to determine when they are unlikely to be able to persuade their opponent and may have more success engaging in a negotiation dialogue instead. As established earlier, the progression rules set out only when it is legal to transition to a new sub-dialogue, not that that transition must occur.

A progression is only legal, at the very earliest, after a request has been made, an argument in support of the request has been made, and the request has still been rejected by the respondent. It is only in the event that the initiator has no argument to justify their position and must make an offer in lieu of a defence or withdraw from the dialogue, that it is in the initiators interests to move straight to a negotiation dialogue. Th progression rules enable the initiator to avoid the kind of fallacy of bargaining attributed to the minister of finance in the Walton and Krabbe example because the initiator has provided a defense of their initial thesis thereby discharging the burden of proof required to satisfy the persuasion dialogue and thereby avoid an illicit shift to a negotiation dialogue.

NP_0 is a protocol tailored towards negotiation-type dialogues. In a negotiation the players may make offers formulated to win acceptance of their goal from their opponent. The offers however need not pertain directly to the goal. Walton and Krabbe recognise in [138] that the swapping of one concession for another is a characteristic of negotiation. In the context of a multiagent system implementation, the agents may have many different capabilities, many of which are not pertinent to the issue at hand but which may be offered as part of a deal in order to get the goal accepted. This kind of dialogue is characterised by offer-counter offer sequences. The rules of NP_0 represented using the A4A schema are as follows;

Composition

Name NP_0
Turns < Strict, Single >
Participants { init, resp }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >

Structure

<Initiation>
[1] Req: (p)∈ CStore_1^{init}, (p)∈ $\text{CStore}_{current}^{init}$, (p)∉$\text{CStore}_{current}^{resp}$
[1] Eff: $\text{Response}_{Next}^{Speaker}$ = <Offer, {p, q}>

<Termination>
[1] Req: (p)∈ CStore_1^{init}, (p)∉ $\text{CStore}_{current}^{init}$
[1] Eff: Game^{Status} = Complete
[1] Req: (p)∈ CStore_1^{init}, (p)∈$\text{CStore}_{current}^{resp}$)
[1] Eff: Game^{Status} = Complete

Interaction

< Offer, {p, q} >
[1] Req: <Offer, (p, q)> ∉ $\text{CStore}^{Listener}$
[1] Eff: $\text{CStore}^{Speaker}$ + (p), $\text{CStore}^{Speaker}$ + (q), $\text{CStore}^{Speaker}$ + <Offer, {p, q} >
$\text{Response}_{Next}^{listener}$ = <Accept, {p}>∨<Reject, {p}>∨<Withdraw, (–)>)∨
<Offer, {p, r}> ∨<Offer, {s, q}> ∨<Offer, {t, u}>

< Accept, {p, q} >
[1] Req: $\text{Transcript}_{Last}^{Listener}$ = <Offer, {p, q}>
[1] Eff: $\text{CStore}^{Speaker}$ + (p), $\text{CStore}^{Speaker}$ + (q), $\text{CStore}^{Speaker}$ + <offer, {p, q}>

< Reject, {p, q} >
[1] Req: $\text{Transcript}_{Last}^{Listener}$ = <Offer, {p, q}>
[1] Eff: $\text{Response}_{Next}^{Listener}$ = <Offer, {p, r}> ∨<Offer, {s, q}> ∨<Offer, {t, u}>

< Withdraw, – >
[1] Req: $\text{Transcript}_{Last}^{Listener}$ = <Offer, {p, q}>∨<Reject, {p, q}>
[1] Eff: Game^{Status} = Complete

The initial setup for an NP_0 dialogue is similar to that for a PP_0 dialogue. Both systems utilise the same number and types of commitment store, the contents of which are preserved between progressions from one sub-dialogue to another. Both players retain their participant identifiers in an NP_0 sub-dialogue, that were established in the preceding PP_0 sub-dialogue, due to the formulation of progression rules. Only the initiator of the PP_0 dialogue is able to initiate a new NP_0 dialogue and hence retains their *init* identifier throughout both sub-dialogues. The similar setups are necessary to enable a clean progression from one sub-dialogue to the next, and a possible subsequent return to the original dialogue type. This approach also enables a consistent representation of supporting machinery between the two systems as required by the A4A.

The global rules for NP_0 specify initiation and termination rules. The initiation rules establish that the initiator has some initial thesis in their commitment store and that that same initial thesis is not present in the respondent's commitment store. The initiation rules also establish that an NP_0 dialogue must begin with an offer move in which the initiator states the goal that they are trying to achieve, in this case the goal is actually the initial thesis which was established at the very beginning of the encompassing persuasion dialogue, along with a proposal that they are willing to concede to get the goal accepted. An NP_0 dialogue can terminate when either the initiator has withdrawn their initial thesis, or the respondent has accepted the initial thesis, or the withdraw move is uttered.

NP_0 incorporates four moves which enable basic bargaining behaviour. The offer move, in the context of a negotiation over action, can be assumed to have the following form, "If you accept X, I will concede Y", where X is some goal that the offerer wants the offeree to achieve and Y is the concession that the offerer is willing to make to achieve X. The offer move requires that the speaker has not previously made the same bid. In the case above, all of X, Y, and the utterance *offer(X, Y)* will be added to the speakers commitment store, so NP_0 allows commitment to offers as well as commitment with respect to the individual statements that comprise the offers. The requirements for this move stop the speaker from repeating a bid that they have already offered.

The offer move can be followed in a subsequent turn by a counter offer. NP_0 recognises four varieties of offer. The first is the initial offer in a negotiation. The remainder are various types of counteroffer in which either, the goal remains the same and the proposal is altered, the goal is altered and the proposal remains the same, or the goal and the proposal are both altered. In the two instances of counteroffers where the goal is altered, it is assumed that the goal is a reduced or related version of the initial goal but the rules do not enforce this. Given the initial offer, "If you accept X, I will concede Y", it should be noted that in the counter-offers the participants are inverted so that the offer should be read as the inversion of the previous offer; for example the first variety of counteroffer is of the form, "I will accept X, If you concede Y'", the second variety is of the form, "I will accept X', if you concede Y", and lastly the final type of counteroffer is of the form, "I will accept X', If you concede Y'". Notice that because

NP_0 dialogues are not entirely symmetrical it is always the case that the goal refers to something that the initiator wants the respondent to accept and that the proposal refers to something that the initiator is conceding. After an initial offer is made the next move can be either outright acceptance or rejection of the offer, or one of the varieties of counteroffer. The accept move enables a player to agree to a given offer and adds the components of the offer and the offer itself to the speakers commitment store so that a player actively commits themself to accept an offer. The reject move enables a player to not accept a proposed offer. Finally the withdraw move is similar to that for withdraw in PP_0.

It should be noted that NP_0 includes no progression rules to govern either return to the parent persuasion dialogue or to enter a new instance of persuasion or negotiation dialogue as a child of the current NP_0 dialogue. This was because it is not required to demonstrate either the use or the utility of the progression from persuasion to negotiation during a dialogue. The machinery of the A4A architecture is sufficiently flexible to enables such transitions to be specified as required either in a manner similar to that used for PP_0 or by specification of a particular move which leads to a progression as part of the effects of playing that move.

4.4 Reformulation of Extant Games

Part of the motivation for developing the A4A was to enable a unified means to specify and implement a range of extant dialectical games. To this end, the remainder of this section presents a reformulation of a number of extant games into the A4A game schema;

4.4.1 Hamblin's 'H'

Composition
 Name H
 Turns $<$ Strict, Single $>$
 Participants { init, resp }
 Stores
 $<$ CStore, Init, Set, Public $>$
 $<$ CStore, Resp, Set, Public $>$

Structure
 Ø

Interaction

$<$ **Statement,** {**p**} $>$
 [1] Req: –
 [1] Eff: CStore^{Speaker} + (p), CStore^{Listener} + (p)
 [2] Req: Transcript$_{Last}$ = <Question, {q}>, form(p,disjunct,q)
 [2] Eff: CStore^{Speaker} + (p), CStore^{Listener} + (p)

< **Statements,** $\{$**p, q**$\}$ >
[1] Req: Transcript$_{Last}$ = < Challenge, $\{$q$\}$ >
[1] Eff: CStore^{Speaker} + (p), CStore^{Speaker} + (p→q), CStore^{Listener} + (p), CStore^{Listener} + (p→q)

< **No Commitment, P** >
[1] Req: –
[1] Eff: CStore^{Speaker} – (P)
[2] Req: |P|=1, Transcript$_{Last}$ = <Question, $\{$q$\}$>, relation(p,disjunct,q)
[2] Eff: CStore^{Speaker} – (P)

< **Question,** $\{$**p**$\}$ >
[1] Req: form(p, disjunction)
[1] Eff: CStore^{Speaker} + (p), Response$^{Listener}_{next}$ = <Statement, $\{\neg$p$\}$>∨<No Commitment, $\{$p$\}$>∨ <Statement, $\{$p$\}$>$_2$∨<No Commitment, $\{$q$\}$>$_2$

< **Challenge,** $\{$**p**$\}$ >
[1] Req: –
[1] Eff: CStore^{Listener} + (p), Response$^{Listener}_{next}$ = <Statement, $\{\neg$s$\}$>∨<No Commitment, $\{$p$\}$>∨<Statements, $\{$q, p$\}$>

< **Resolve,** $\{$**p**$\}$ >
[1] Req: –
[1] Eff: Response$^{Listener}_{next}$ = <No Commitment, $\{$P$\}$>∨<No Commitment, $\{\neg$P$\}$>

< **No Commitment, P** > + < **Challenge,** $\{$**p**$\}$ >

An aspect of H that is interesting is that its Statement and No Commitment locutions both have two different instances. In the first instance of the Statement locution there are no requirements and the effect is to add the content of the locution into bothe players commitment stores. In the second instance the requirement is that the previous move was a Question locution and that the form of the responsive Statement locution is that of a disjunct of the content of the previous locution. The second instance of the Statement locution is a specified response to the question locution as can be seen in the formulation of effects for that move. It is the same case with the two instances of the No Commitment move. The response to the Question locution requires that both instances are valid responses but with a different formulation of content. For example, in response to <Question, $\{$p$\}$> a player may retract commitment to the whole of p, or alternatively because p has the form of a disjunction, retract commitment to one of the conjuncts of p.

4.4.2 Mackenzie's 'DC'

Composition
Name DC
Turns < Strict, Single >
Participants $\{$ init, resp $\}$
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >

Structure

< **Initiation** >
[-] Req: Turn = 0
[-] Eff: CStore$^{Speaker}_0$ = Ø, CStore$^{Listener}_0$ = Ø

Interaction

< **Statement,** {**p**} >
[1] Req: ¬((p)∈CStore^{Speaker} ∧ (p)∈CStore^{Listener})
[1] Eff: CStore^{Speaker} + (p), CStore^{Listener} + (p)

< **Defence,** {**p**} >
[1] Req: Transcript$_{Last}$ = <Challenge, {q}>, (<Why, {p}>)∉CStore^{Listener}
[1] Eff: CStore^{Speaker} + (p), CStore^{Speaker} + (p→q), CStore^{Listener} + (p), CStore^{Listener} + (p→q)

< **Withdraw,** {**p**} >
[1] Req: ¬form(p,imc-conditional)
[1] Eff: CStore^{Speaker} – (p)
[2] Req: Transcript$_{Last}$ = <Resolve, {s}>, form(s, q→r), relation(p,conjunct,q)
[2] Eff: CStore^{Speaker} – (s)
[3] Req: Transcript$_{Last}$ = <Resolve, {q}>, form(q, conjunction), relation(p,conjunct,q)
[3] Eff: CStore^{Speaker} – (s)

< **Question,** {**p**} >
[1] Req: Ø
[1] Eff: Response$^{Listener}_{Next}$ = <Statement, {p}>∨<Statement, {¬p}>∨<No Commitment, {p}>

< **Challenge,** {**p**} >
[1] Req: ¬form(p,imc-conditional)
[1] Eff: CStore^{Listener} + (p), CStore^{Speaker} – (p), CStore^{Speaker} + (<Why, {p}>), Response$^{Listener}_{Next}$ = <No Commitment, {p}>∨<Resolve, {p}>∨<Defence, {q}>

< **Resolve,** {**p**} >
[1] Req: Transcript$_{Last}$ = <Challenge, {r}>∨<No Commitment, {r}>, form(p, q→r), {p}∈CStore^{Listener}
[1] Eff: Response$^{Listener}_{Next}$ = <Withdraw, {s}>$_2$∨<Withdraw, {r}>
[2] Req: form(p, conjunction)
[2] Eff: Response$^{Listener}_{Next}$ = <Withdraw, {q}>$_3$

Of note in DC is that the Statement locution has been split into two different locutions in the A4A to account for the contexts in which it is used. In the first context the locution is used merely as a way to assert a given content token. In the other context the locution is used as a defense of an earlier challenged content token. In DC both of these contexts are bound up into the Statement move but in the reformulation it made sense to reconstruct this locution into two separate locutions having different requirements and effects.

4.4.3 Walton's 'CBZ'

Composition
Name CBZ
Turns < Strict, Single >
Participants { init, resp }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >
< DStore, Init, Set, Private >
< DStore, Resp, Set, Private >

Structure

< **Termination** >
[1] Req: (p)∈CStore$^{init}_{1}$∧(p)∈CStore^{resp}

[1] Eff: Game^{Winner} = init, Game^{Loser} = resp
[2] Req: (p)∈CStore$_2^{resp}$∧(p)∈CStore^{init}
[2] Eff: Game^{Winner} = resp, Game^{Loser} = init

Interaction

< **Statement, {p}** >
[1] Req: –
[1] Eff: CStore^{Speaker} + (p), [(¬p)∈DStore^{Speaker}]→[CStore^{Speaker} + (¬p)]

< **Withdraw, {p}** >
[1] Req: –
[1] Eff: CStore ^{Speaker} – (p),
[(p) ∈ CStore ^{Listener}] → [Response $^{Listener}_{Next}$ = <Resolve, {p, ¬p } >],
[(p) ∈ DStore ^{Speaker}] → [CStore ^{Speaker} + (p)]

< **Question, {p}** >
[1] Req: –
[1] Eff: Response$^{Listener}_{Next}$ = <Statement, {p}>∨<Statement, {¬p}>∨<Withdraw, {p}>

< **Challenge, {p}** >
[1] Req: –
[1] Eff: CStore^{Listener} + (p), Response$^{Listener}_{Next}$ = <Withdraw, {p}>∨<Statement, {q}>

< **Resolve, {p, ¬p}** >
[1] Req: (p)∈CStore^{Listener}∨(¬p)∈CStore^{Listener}
[1] Eff: Response$^{Listener}_{Next}$ = <Statement, {p}>∨<Statement, {¬p}>

CBZ is interesting insofar as the rules don't specify a relationship between p and q when q is uttered in response to the challenge of p. It is assumed that the purpose of the challenge locution is to either gain grounds, t, that support s or to get commitment to s to be retracted.

It is difficult to formulate this game as a computational dialectical game for a couple of reasons. Firstly it doesn't provide rules for establishing the contents of the players dark-side commitment stores. Secondly, it suggests in commitment rule v that the participants may communicate outside of the dialogue but does not formulate rules for this. The same drawback can be identified in strategic rule (iv) which suggests that firstly the game can be ended by universal consent, and secondly that the players can agree to start a new game with the same commitment store contents. In neither case are rules formulated to determine how the participants, within a given game should achieve these agreements.

There are no rules for establishing the initial thesis of the players. It is assumed that in their first turn, each player states their position so as to establish an initial thesis. The initial thesis for the *init* player therefore is the content of the CStore^{init} at time step 1 and the initial thesis of the resp is the content of CStore^{resp} at time step 2. Strategic rule (ii) is not explicit enough with respect to showing that the initial thesis of one participant is an immediate consequence of the commitments of the other player

4.4.4 Girle's 'DL3'

Composition

Name DL3
Turns < Strict, Single >
Participants { init, resp }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >

Structure

Ø

Interaction

< **Categorical Statement,** {**p**} >
[1] Req: $\text{Transcript}_{Last} \neq$ <Challenge, {p}>, ¬((s)∈$\text{CStore}^{Speaker}$ ∧ (s)∈$\text{CStore}^{Listener}$)
[1] Eff: $\text{CStore}^{Speaker}$ + (p), $\text{CStore}^{Listener}$ + (p)

< **Reactive Statement,** {**p**} >
[1] Req: (<Challenge, {q}>)∉$\text{CStore}^{Speaker}$, (q)∈$\text{CStore}^{Speaker}$, (p)∉$\text{CStore}^{Speaker}$, (p)∉$\text{CStore}^{Listener}$
[1] Eff: $\text{CStore}^{Speaker}$+(p, if p then q, q, <Challenge, q>), $\text{CStore}^{Listener}$+(p, if p then q, q, <Challenge, q>), $\text{CStore}^{Speaker}$–(<Challenge, {q}>), $\text{CStore}^{Listener}$–(<Challenge, {q}>), $\text{CStore}^{Speaker}$+(p), $\text{CStore}^{Listener}$+(p)

< **Logical Statement,** {**p**} >
[1] Req: (p)∉$\text{CStore}^{Speaker}$, (p)∉$\text{CStore}^{Listener}$
[1] Eff: $\text{CStore}^{Speaker}$+(p), $\text{CStore}^{Listener}$+(p)

< **Term Declaration,** {**p**} >
[1] Req: (p)∉$\text{CStore}^{Speaker}$, (p)∉$\text{CStore}^{Listener}$
[1] Eff: $\text{CStore}^{Speaker}$+(p), $\text{CStore}^{Listener}$+(p)

< **Withdrawal,** {**p**} >
[1] Req: ¬form(p,imc-conditional), ¬form(p,conditional)
[1] Eff: $\text{CStore}^{Speaker}$–(p, if p then q, q, <Challenge, q>), $\text{CStore}^{Listener}$–(p, if p then q, q, <Challenge, q>), $\text{CStore}^{Speaker}$–(p), $\text{CStore}^{Speaker}$–(q),
[Transcript_{Last} = <Challenge, {p}>]→[$\text{CStore}^{Speaker}$–(<Challenge, {p}>), $\text{CStore}^{Listener}$–(<Challenge, {p}>)]

< **TF-Question,** {**p**} >
[1] Req: –
[1] Eff: $\text{CStore}^{Speaker}$–(p), $\text{Response}^{Listener}_{Next}$ = <Statement, {p}>∨<Statement, {¬p}>∨<Withdrawal, {p}>

< **WH-Question,** {**p**} >
[1] Req: –
[1] Eff: $\text{CStore}^{Speaker}$+(<WH-Question,{p}>), $\text{CStore}^{Speaker}$+(<WH-Answer,{p}>), $\text{CStore}^{Listener}$+(<WH-Answer,{p}>), $\text{Response}^{Listener}_{Next}$ = <Term Declaration, {p}>∨<Withdrawal, {p}>∨<Statement, {¬p}>

< **Challenge,** {**p**} >
[1] Req: –
[1] Eff: $\text{CStore}^{Speaker}$+(<Why,{p}>), $\text{CStore}^{Listener}$+(<Why,{p}>), $\text{CStore}^{Speaker}$–(p), $\text{CStore}^{Listener}$+(p), $\text{CStore}^{Speaker}$–(p, if p then q, q, <Challenge, q>), $\text{CStore}^{Listener}$–(p, if p then q, q, <Challenge, q>), $\text{Response}^{Listener}_{Next}$ = <Withdrawal, {p}>∨<Statement, {¬p}>∨<Resolve, {p}>∨<Reactive Statement, {q}>

The main difference between DL3 and other games presented here is that it has a greater range of locutions, in particular the statement locution is split into a number of different locutions to account for the range of contexts in which a statements may be uttered such as uttering a defense of a challenged statement. Such a defense is termed a Reactive Statement in DL3. The othe major difference is that sequences of artifacts may be manipulated about the player's commitment stores rather than just locutions and the content tokens of locutions.

4.4.5 Walton & Krabbe's 'PPD$_0$'

Composition

Name PPD$_0$
Turns < Strict, Multiple >
Participants { init, resp }
Stores
< Assertions, Init, Set, Public >
< Assertions, Resp, Set, Public >
< Concessions, Init, Set, Public >
< Concession, Resp, Set, Public >
< DarkStore, Init, Set, Public >
< DarkStore, Resp, Set, Public >

Structure

< **Initiation** >
[1] Req: Turn = 0
[1] Eff: CStore$_0^{Speaker}$ = Ø, CStore$_0^{Listener}$ = Ø

Interaction

< **Assert,** {**p**} >
[1] Req: –
[1] Eff: Assertions^{Speaker} + (p), Assertions^{Listener} + (p)

< **Concede,** {**p**} >
[1] Req: (p)∉Concessions^{Speaker}, (p)∈Assertions^{Listener}
[1] Eff: Concessions^{Speaker} + (p)
[2] Req: (p)∉Concessions^{Speaker}, Transcript$_{Last}$=<Request, {p}>∨<Serious, {p}>
[2] Eff: Concessions^{Speaker} + (p)

< **ElemArg,** {**p, q**} >
[1] Req: form(p,conjunction), (q)∉Concessions^{Speaker}, Transcript$_{Last}$=<Challenge, {q}>
[1] Eff: [∀ r: conjunct(r,p)] Response$_{Next}^{Listener}$ = <Weak Retraction, {r}>∨<Challenge, {r}>, Assertions^{Speaker}+(p), Assertions^{Speaker}+(p→q), Concessions^{Speaker}+(p), Concessions^{Speaker}+(p→q)

< **Request,** {**p**} >
[1] Req: (p)∉Concessions^{Listener}
[1] Eff: Response$_{Next}^{Listener}$ = <Concede, {r}>∨<Weak Retraction, {r}>

< **Serious,** {**p**} >
[1] Req: Transcript$_{Last}^{Listener}$ = <Weak Retraction, {p}>∨<Challenge, {p}>, (p)∉DarkStore^{Listener}
[1] Eff: Response$_{Next}^{Listener}$ = <Concede, {p}>∨<Weak Retraction, {p}>

< **Resolve,** {**p, q**} >
[1] Req: (p)∈Concessions^{Listener}, (q)∈Concessions^{Listener}
[1] Eff: Response$_{Next}^{Listener}$ = <Weak Retraction, {p}>∨<Weak Retraction, {q}>

< **Challenge,** {**p**} >
[1] Req: (p)∈Assertions^{Listener}, (p)∈Concessions^{Speaker}, Transcript$_{Last}^{Listern}$≠<Challenge, {p}>
[1] Eff: Response$_{Next}^{Listener}$ = <ElemArg, {–,p}>∨<Weak Retraction, {p}>∨<Strong Retraction, {p}>

< **Weak Retraction,** {**p**} >
[1] Req: ¬((p)∉DarkStore^{Listener} ∧ Transcript$_{Past}^{Listener}$≠<Serious, {p}>), Transcript$_{Last}^{Listener}$ = <Request, {p}>∨<Serious, {p}>
[1] Eff: Assertions^{Speaker}–{p}, Concessions^{Speaker}–{p}
[2] Req: ¬((p)∉DarkStore^{Listener} ∧ Transcript$_{Past}^{Listener}$≠<Serious, {p}>), (s)∉Concessions^{Speaker}
[2] Eff: Assertions^{Speaker}–{p}, Concessions^{Speaker}–{p}

< **Strong Retraction,** {**p**} >
[1] Req: –
[1] Eff: Assertions^{Speaker}–{p}

An interesting aspect of PPD$_0$ is that it allows more than one locution to be uttered by a player in their turn. This is the only game therefore that can use rules which allow a single move to specify a set of multiple alternative responses to be made. Single move games specify a set of alternative responses but because they only allow a single move to be played per turn only a single response can be made to a given locution. PPD$_0$ however allows multiple locutions to uttered in response to a single locution. An example of this kind of *branching response* can be found in the *ElemArg* move which specifies a response in relation to each each premise of the argument.

As was the case for Walton's CBZ an aspect of PPD$_0$ that is difficult to transfer to the computational domain without formulating additional rules to account for it is the dark-side commitment store. Specifically how the contents of the dark-side store should be established. Although PPD$_0$ can be played without stalling using empty dark-side commitment stores this is unsatisfactory because a computational implementation of the game should use all available components of the game.

4.4.6 Bench-Capon's 'TDG'

Composition
Name TDG
Turns < Liberal, Single >
Participants { init, resp1, resp2 }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >
< ClaimStack, –, Stack, Public >

Structure

< **Initiation** >
[1] Req: Turn = 0
[1] Eff: Roles^{init} + Speaker, Roles^{init} + Proponent, Roles^{resp1} + Listener, Roles^{resp1} + Opponent, Roles^{resp2} + referee, Response$^{Speaker}_{Next}$ = <Claim, {p}>

Interaction

< **Claim, {p}** >
[1] Req: –
[1] Eff: ClaimStack + (p), CStore$Proponent$ + (p), Roles^{Opponent} + Speaker, Roles^{Proponent} + Listener

< **Why, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, (p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: Roles^{Proponent} + Speaker

< **OK, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, (p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: ClaimStack – (p), CStore^{Opponent}+(p), CStore^{Opponent}–(¬p), Roles^{Referee} + Speaker

< **So, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, (q→p)∉CStore^{Speaker}, (p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: Roles^{Proponent} + Speaker

< **Presupposing, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, (q→p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: Roles^{Proponent} + Speaker

< **On Account Of, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, (q→p) $\overset{top}{\rightarrow}$ ClaimStack, Transcript$^{Proponent}_{Last}$ = <Supply Warrant, p>
[1] Eff: Response$^{Proponent}_{Next}$ = <Supply Backing, p> ∨ <Withdraw, p>, Roles^{Proponent} + Speaker

< **Supply Data, {p, q}** >
[1] Req: relation(p,grounds,q), Speaker∈Roles^{Proponent}, Transcript$^{Oppponent}_{Last}$ = <Why, q>, (q) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: CStore^{Proponent}+(p), ClaimStack+(P), Roles^{Opponent} + Speaker,
Response$^{Opponent}_{Next}$ = <Switch Focus, p>∨<OK , p>∨<So, p>∨<Why, p>

< **Supply Warrant, {p}** >
[1] Req: form(p, q→r), Speaker∈Roles^{Proponent}, Transcript$^{Opponent}_{Last}$ = <So, r>, (r) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: CStore^{Proponent}+(p), ClaimStack + (p), Roles^{Opponent} + Speaker,
Response$^{Opponent}_{Next}$ = <Presupposing, p>∨<On Account Of , p>∨<OK, p>

< **Supply Presupposition, {p}** >
[1] Req: Speaker∈Roles^{Proponent}, (q→r) $\overset{top}{\rightarrow}$ ClaimStack, Transcript$^{Opponent}_{Last}$ = <Presupposing, r>
[1] Eff: CStore^{Proponent}+(p), CStore^{Proponent}+(¬p→¬r), ClaimStack + (p), Roles^{Opponent} + Speaker

< **Supply Backing, {p}** >
[1] Req: Speaker∈Roles^{Proponent}, Transcript$^{Opponent}_{Last}$ = <On Account Of, p>, Transcript$^{Proonent}_{Past}$ = <Supply Warrant, p>, (q→p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: Roles^{Referee} + Speaker, CStore^{Opponent} + (q→p), ClaimStack – (q→p)

< **Withdraw, {p}** >
[1] Req: Speaker∈Roles^{Proponent}, (p) $\overset{top}{\rightarrow}$ ClaimStack
[1] Eff: ClaimStack – (p), CStore^{Proponent} – (p), Roles^{Referee} + Speaker

< **Swtich Focus, {p}** >
[1] Req: (p) $\overset{\neg top}{\rightarrow}$ ClaimStack, (p) $\overset{on}{\rightarrow}$ ClaimStack, Speaker∈Roles^{Opponent}, p∉CStore^{Opponent}
[1] Eff: ClaimStack + (p), Roles^{Opponent} + Speaker

< **Current Claim, {p}** >
[1] Req: (p) $\overset{top}{\rightarrow}$ ClaimStack, Speaker∈Roles^{Referee}, (p)∈CStore^{Proponent}
[1] Eff: Roles^{Opponent} + Speaker

< **End, {p}** >
[1] Req: Speaker∈Roles^{Referee}, |ClaimStack| = Ø
[1] Eff: Game^{Status} = Complete

< **Rebut, {p}** >
[1] Req: Speaker∈Roles^{Referee}, (p) $\overset{top}{\rightarrow}$ ClaimStack, (p)∉CStore^{Opponent}, (q→p)∉CStore^{Opponent}
[1] Eff: Roles^{Opponent} + Speaker

< **Rebuttal, {p}** >
[1] Req: Speaker∈Roles^{Opponent}, Transcript$^{Opponent}_{Last}$ = <Rebut, q>, (q)∉CStore^{Speaker}, (q)∉CStore^{Listener}
[1] Eff: ClaimStack + (p), CStore^{Speaker} + (¬q), CStore^{Speaker} + (p), ClaimStack + (q), CStore^{Speaker} + (p→¬q), ClaimStack + (p→¬q)
Response$^{Opponent}_{Next}$ = <Presupposing, p>∨<On Account Of , p>∨<OK, p>,
Roles^{Speaker} + Proponent, Roles^{Listener} + Opponent, Roles^{Opponent} + Speaker

TDG has a number of differences to the other games examined here. Most prominent is that it has on average many more requirements and effects for each locution. This is partly due to the use of the liberal turn structure which means that for every locution, it must be specified which player is in which role and also any alterations to the role assignment as a result of the move being played. The greater number of requirements and effects inherent in TDG is also due in part to the original aims of this particular game. TDG was developed as a specification for a dialectical game specifically aimed at a computational implementation [1]. Hence the specificity of the rules of TDG are much closer to the requirements of an A4A specification. Another prominent difference is the use of the claim stack to control the flow of the dialogue in light of the claims that are made. The claim stack artifact store does not specify an owner and is shared between the participants of the dialogue.

It is not possible to produce an exact reformulation of each game for a number of reasons. Primarily the original games have been formulated using natural language descriptions which are inherently ambiguous and each decision that is made by a game implementor with regards to an ambiguity can lead to a number of different formulations for the game. An even larger problem is that a number of games are not full systems insofar as they suggest rules but do not formulate particulars for their use. For example, CBZ and PPD_0 suggest rules for a maximum number of tokens, or locutions but do not formulate rules for specifying these numbers. They also make use of dark-side stores but do not specify how the contents of these stores may be established. More problematic are the rules that suggest alternative plays but formulate these in terms of player agreement without rules for establishing how that agreement is gained. For example the rule of CBZ that suggests that a move is legal only if the other player agree to its begin played. To fill in these gaps, or to decide on an interpretation with regards to an ambiguous rule risks altering the intent behind the rules of the original game thereby creating a variant game. One variant is very likely incompatible with others so any software that uses a fixed set of dialectical game rules will not be able to use the new formulation. However, one strength of the A4A is that the game schema approach is designed to be lightweight, flexible and not fixed, so that an application can easily load a different schema. Dialogue participants can therefore either agree on a schema to govern their interactions or if one application uses the A4A and the other is fixed with respect to its game, the A4A-based software can load the required schema to enable compatibility.

4.5 The A4A Software Implementation

An implementation of the A4A meta-game and game schema has been written using the Java programming language. The implementation is in the form of an application programming interface (API) rather than a standalone application. This enables the A4A to be more easily incorporated into a range of end-user applications and simplifies the deployment of dialectical games into JAVA based MAS. The selection of Java was made because of its widespread adoption in the development of internet based, argumenta-

[1]Learnt through discussion with the author of TDG.

tive dialogue and discussion applications such as the Parmenides discussion forum [7], its popularity amongst developers of multiagent systems [67, 42, 35], and its successful use in argumentation analysis software such as Araucaria [104]. These are all areas in which a Java based API for representing dialectical games can find utility. By using Java, the A4A could thus easily be incorporated into applications aimed at a range of differing deployments without requiring the developers to convert between disparate implementational languages. Figure 4.3 shows the class diagram for the A4A implementation. The A4A implementation can be interpreted in terms of four related components as follows;

1. Representation of the dialogue's participants
2. Recording the dialogue's transcript
3. Recording artifacts in the artifact Stores
4. Representation of the rules of the game through one or more game schemas.

Each of these components corresponds to a different sub-hierarchy of classes in the A4A implementation such that a dialectical game has a set of participants, a transcript, a set of artifact stores and a set of game schemas which govern how the participants, transcript and artifact stores interact. The *DialecticalGame* class forms the basis for any deployment of the A4A. It provides an interface to the A4A and serves as a unified point of access for any external code.

Figure 4.4 shows the class diagram for the representation of games schemas within the A4A implementation. This corresponds directly to the game schema laid out in section 4.2. The composition section of a game schema, used to specify the extensible and optional components, are recorded within the schema class. Each structural rule from the structural rules section of the game schema is recorded as an instance of the *structuralRule* class and stored in a structure object in the *Schema* class. Each interaction rule, as specified in the interaction rules section of the game schema, is recorded as an instance of the *MoveSpec* class.

4.5.1 Playing Dialectical Games

The A4A is designed to be embedded within, and to provide game representation capabilities to, a software application external to the A4A. This means that, for example, an agent may embed the A4A alongside other components to provide knowledge representation, reasoning, and planning capabilities

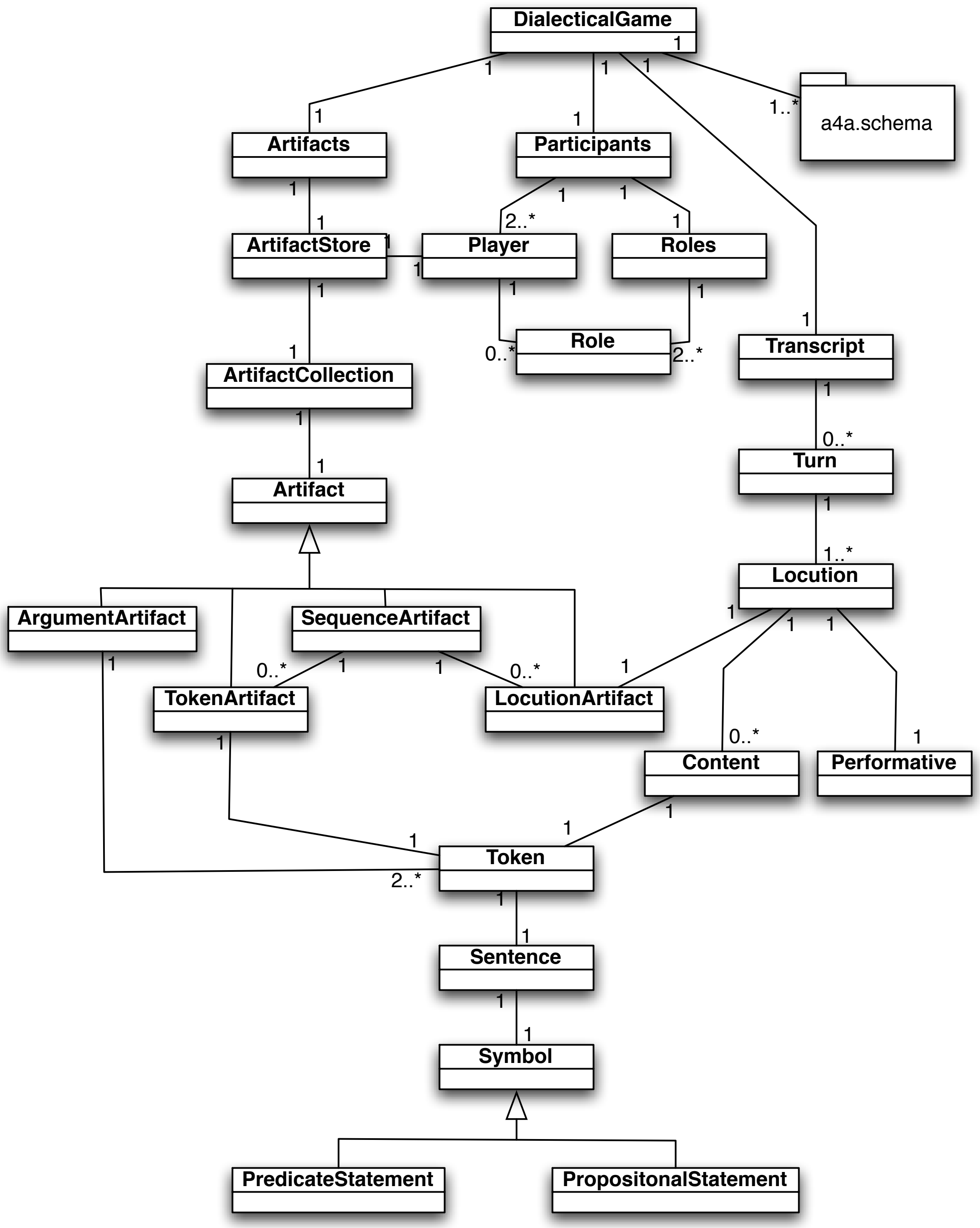

Figure 4.3: A4A Class Diagram

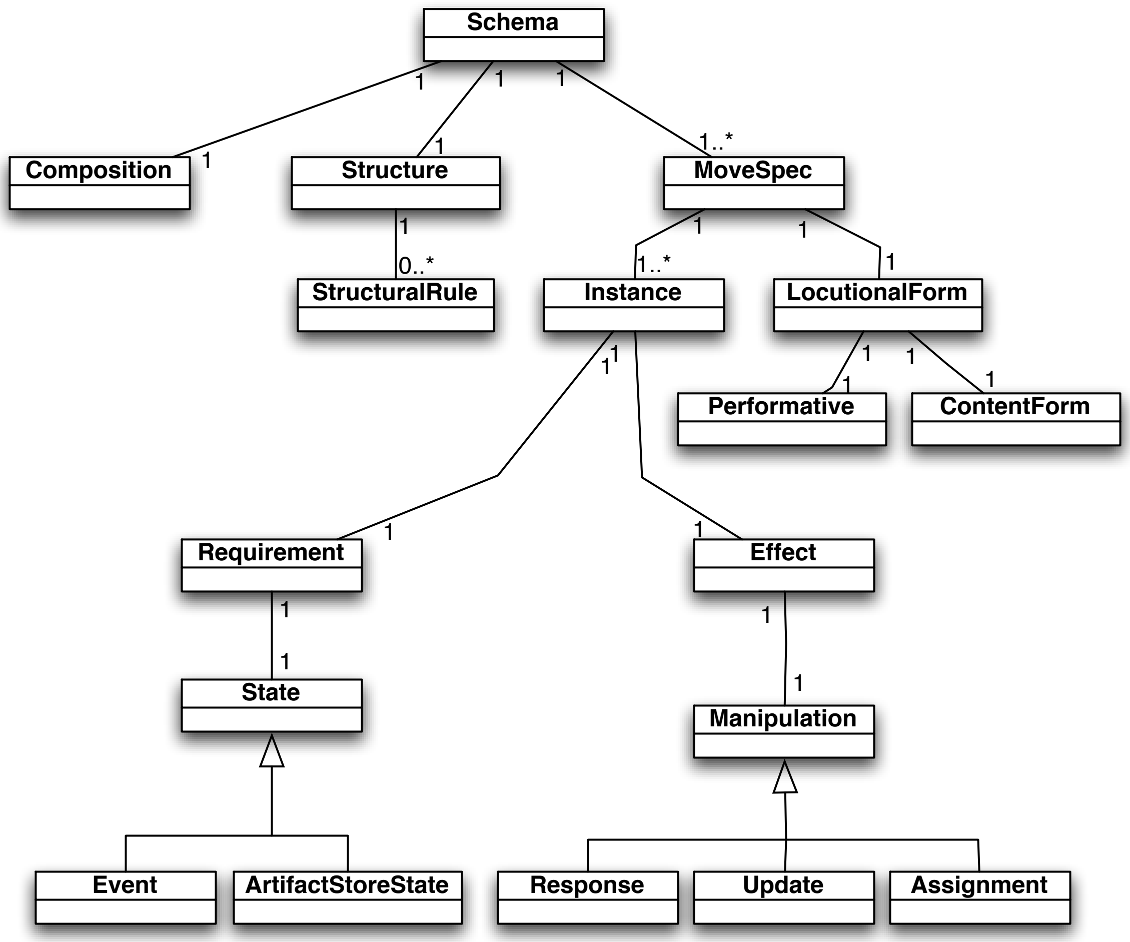

Figure 4.4: A4A Game Schema Class Diagram.

amongst others. The A4A does not provide message passing facilities, a reasoning process for determining the correct move to play from the legal moves, or a planning process to guide dialectical strategy, and relies upon the external application to determine its strategy, select the correct moves, and send them to its dialogue partners. What the A4A does do however is the following;

1. Specify the archetypal form of moves that can currently be legally played
2. Verify that fully instantiated moves are legal given the current state of the game
3. Update the game in light of sent and received utterances
4. Record the dialogue and all supplemental data such as artifact stores
5. Provide access to the records of current and past games
6. Provide access to the game schemas associated with current or previous games
7. Provide access to all of the game schemas available to player

The first three capabilities are facilitated by the A4A utterance interface which specifies that the *DialecticalGame* class incorporates methods to retrieve a list of legal moves at the current point, validate any fully instantiated moves that the external application has constructed, and update the state of game components in light of played moves. The game state is only updated if the associated played move is valid according to the current game schema. This validation process occurs internally to the A4A and any given move is either valid or invalid dependent solely upon the requirements of the moves and any associated structural rules that are met. If a move is invalid then the game state is not updated and a new move must be played instead. To deal with such situations a game schema might incorporate rules to deal with players who attempt to play invalid moves. The default behaviour however is to not update the game in light of invalid moves but to wait until a valid move is played. The fourth capability is necessary to keep track of the moves that have been played so far during a game. The remaining capabilities are required to access the data associated with any dialogues that the software has or is engaged in, and to access the rules of each dialectical game that the software has available. The data access capabilities provide access to the game schema and current game state to enable external strategist and planner components to determine which locution to make.

4.5.2 Turn Progression

The A4A supports a flexible role-based means of controlling turn progression. The meta-game specifies a range of roles that can be assigned to players and these are represented by role classes. One pair of such roles are the speaker and listener roles. Currently only one player can occupy the speaker role during any given turn although any number of participants may occupy the listener role. The only player who can legally make a move is the speaker. Turn progression is thus implemented within the A4A framework using one of two methods. In the first method which is applied to games with a strict turn structure, the speaker role is assigned to the next player in the participants list after each turn. In the second method which is applied to games that have a liberal turn structure, the speaker role is assigned according to the role assignments specified for each move. When a new utterance is received by the DialecticalGame class, the participants are checked to determine which one holds the speaker role. The utterance is thus legal with respect to its speaker if the player who made the utterance is the same player who is currently assigned the speaker role.

4.5.3 Artifact Stores

Artifact stores record the state of game components over time in line with the progression of the players turns. Between each turn the artifact stores contain the current state of those components that the game rules require to be tracked. During each turn the contents of a given store may be updated to add, remove, or rearrange those contents as required. This progression of artifact store states is illustrated in figure 4.5.

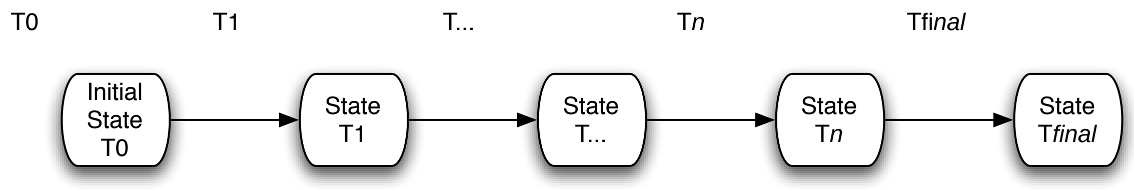

Figure 4.5: The progression of artifact store states.

To enable a store's contents to be efficiently tracked over the course of a dialogue each distinct store is structured internally as a sequence of collections of artifacts where the sequence is indexed by turn. To achieve this, before each turn, the state of the store in the last turn is cloned and the clone is manipulated during the current turn. The clone is then added to the sequence of stores ready for possible manipulation during the next turn. This approach is more efficient in terms of processing requirements when the contents of artifact stores at earlier points in the dialogue must be inspected. This is because the state of the store at every previous turn is recorded and ready for inspection. The alternative approach is to use a single store which represents the current state of that store but which must be processed to determine the state of the store at any specified earlier point in the dialogue by undoing each update made between the current turn and the required earlier turn. By cloning the stores after each turn a more computationally efficient implementation can be produced but this efficiency comes with the expense of increased memory usage. The internal structure of A4A artifact stores is illustrated in figure 4.6.

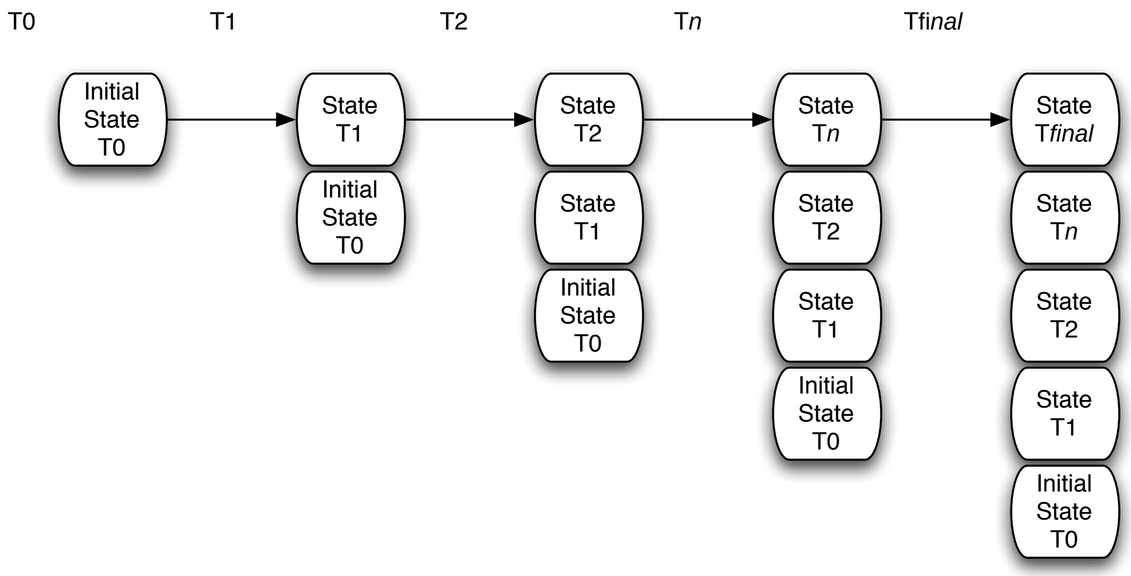

Figure 4.6: The internal structure of A4A artifact stores.

4.5.4 Interfacing to a Knowledge Base

Whilst the rules of a dialectical game can be used to establish the kinds of moves that a player can make at any given point, they do not make any provision for what the content portion of those moves should actually consist of. The content and the manner in which that content is expressed are dependent upon the aims of the application which makes use of the A4A. In a multiagent system knowledge concepts will most likely be stored in a knowledge base which stores domain relevant information. The concepts stored in this knowledge base may be expressed in various ways including as propositional statements, as predicate statements, as RDF, or as instantiated KIF messages. Yet from the perspective of the A4A the content portion of a locution consists of no more than statement variables which represent particular expressions of knowledge concepts. To make the A4A applicable to a wide range of situations, requiring the least amount of new code for each new situation, knowledge concepts are abstracted away from their knowledge base implementation into content tokens which are manipulated within the A4A framework. When a statement, of whichever form is uttered during a game a corresponding token is instantiated to represent the statement and it is the token that is manipulated by the game.

Many dialectical games require minimal, if any, interface to a specific knowledge base. The content of locutions is, from the point of view of a game schema, no more than a variable which is instantiated during the play of an actual game. Similarly from the perspective of a dialogue transcript, the content of played locutions are merely variables that are established, possibly in relation to previously established variables, during a game and whose validity can be established through application of the rules. For the most part dialectical games do not require to understand the content of locutions as anything more than a variable whose validity has been established according to the rules of the game. The A4A therefore does not specify the content of a token but that a token must stand in some position relative to existing statements. It is then the responsibility of the external knowledge base to produce content that satisfies the specified conditions.

This approach has both advantages and disadvantages, the advantage is that it does not subsume related components into the A4A architecture. To play a dialectical game requires a knowledge domain according to which the games tokens are instantiated however games can be specified and recorded without that knowledge domain needing to be a part of the game architecture. The disadvantage is that some aspects of the game are outside the A4A architecture. For example, the A4A must rely upon the external knowledge base to produce the required expressions, and to verify that they are valid expressions when utterances are received from other participants.

4.6 Summary

This chapter has introduced the Architecture for Argumentation (A4A), a framework for developing, expressing, representing, and implementing dialectical games. Development and representation of games is handled in the A4A using a meta-game, to represent those elements that are common to all dialectical games, and a game schema, to account both for the extensible and optional components, and to provide a unified layout for the specification of game rules. The game schema provides a clean and simple means to represent individual games and provides a useful starting point for making a comparison between games. This is because it is simple to see how the game is structured in terms of its constituent elements, and how the various rules regulate the legal moves available to the players. By taking a move from one game and a move from another game it is a straightforward matter to compare their locutional structures, sets of requirements, and sets of effects to see how the two moves are similar or differ. An even greater benefit of the game schema is that it provides a way to achieve a common implementation on which to run a wide range of games. Furthermore, by incorporating the A4A into a software application the actual game that is used by the software can be altered at runtime by loading a game specification file. This means that the specific game played can be selected (to some extent) by the end user or even the software, rather than by the developer.

Chapter 5

Dialectical Games & MAS

THE A4A can be utilised as a standalone command line application or as an API and framework for incorporating dialectical games into end-user applications. One application domain in which dialectical games are of increasing interest is that of Multiagent Systems (MAS) where they are finding utility as a means to regulate the dialectical aspects of inter-agent communication. This chapter investigates two aspects of arguing agents and dialectical games;

- Firstly, given a representation of a dialectical game, what other components are required to construct an agent that can use that game?
- Secondly, given the paucity of implemented arguing agent systems, how should the process of building such an system be approached?

Possible answers to these questions have been explored via two investigations into the use of dialectical games in the context of MAS;

1. MASADA - This project seeks to develop a component-oriented framework for building and deploying arguing agent systems.
2. The Information Exchange - This project (known as the "*i*-Xchange") sought to integrate disparate agent capabilities and their related components, each developed separately and using different development frameworks, tools and methodologies, into a cohesive agent architecture.

Both MASADA and the *i*-Xchange project incorporate the A4A and cover two canonical contexts in which arguing agents might be constructed. In the first approach, typified by MASADA, the developer has complete control over the entire agent including its knowledge and capabilities and can include those components necessary for dialectical argumentation without considering the needs of other components. In the second approach, typified by the *i*-Xchange project, the developer has only partial control of the components that are used to build an agent and the dialectical argumentation component is only a sub-component of a larger agent that includes other unrelated capabilities.

5.1 MASADA

The MultiAgent System And Dialectical game Architecture (MASADA[1]) is a modular, component-oriented architecture for building and deploying autonomous, intelligent, *arguing* agents. MASADA is comprised of Colloqui, a knowledge representation, reasoning and communication component, and Caucus, a component for representing agents. Together these components enable a society of agents to be created in which the agents are capable of engaging in argumentative dialogue in order to resolve any conflicts that are identified. Supporting tools are provided for visualising the agent society and associated data. The relationships between these components are illustrated in figure 5.1.

The development of MASADA has two aims. The first aim is to facilitate the set-up of a MAS in which agents act autonomously, and engage in rule regulated dialogues, to satisfy domain specific goals. The purpose of this is so that empirical evaluative data can be produced. This aim is explored in chapter 6. The second aim is to explore the process of constructing arguing agents. Although there has been much research into protocols for governing agent interaction which are based upon dialectical games, this has not been translated into standard approaches or methodologies for constructing arguing agents based upon those protocols. By implementing various dialectical games the range of capabilities that an arguing agent requires can be identified

To begin tackling these aims an initial problem domain was required that could act as a testbed for both evaluating games and constructing agents. Such a problem domain should satisfy the following criteria;

1. The problem domain should provide a social context for motivating argumentative dialogue that requires both individual, goal directed behaviour as well as multi-agent communication and interaction.

[1]Masada is the site of a ruined palace which was fortified by Herod the Great in the first century BC. It lies on the eastern edge of the Judean desert and overlooks the Dead Sea.

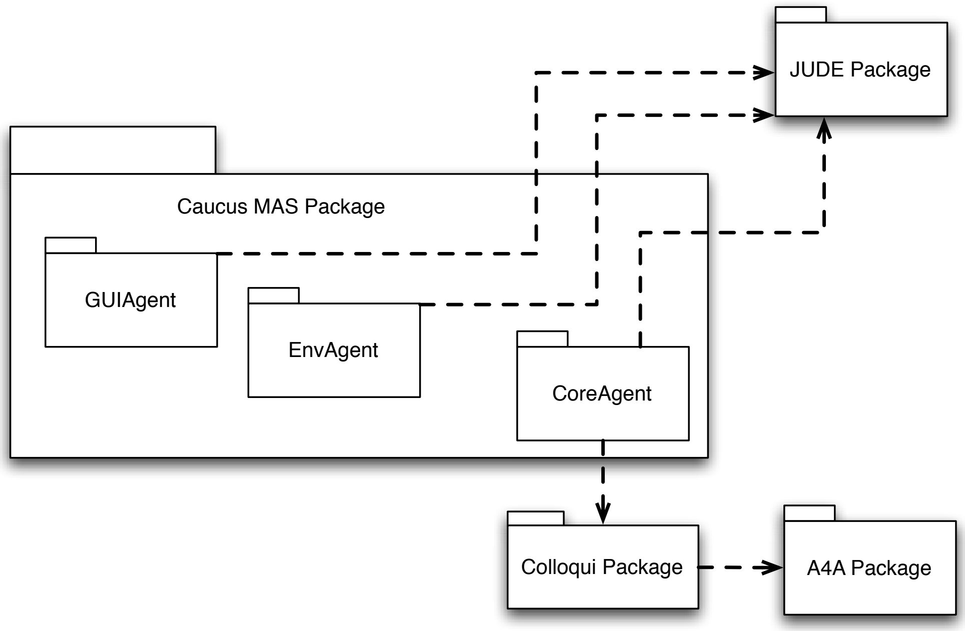

Figure 5.1: The Components of MASADA

2. The knowledge domain, as distilled from the problem domain, must provide a basis for argumentative discourse between agents such as conflict resolution over scarce resources. Dialogue artifacts such as the statements expressed by the agents during a dialogue should be rooted in the state of the MAS so that the basis for arguments expressed by the agents can be examined and verified through recourse to the state of the MAS at the time of expression.

3. The problem domain should provide sufficient scope for rich agent interactions and behaviour whilst being sufficiently straightforward that generated dialogues and arguments are simple to comprehend and analyse. If this balance is correctly achieved then dialogues reminiscent of actual real-world discourse should be created whilst avoiding confusion over whether an artifact is due to the dialectical game or an overly complex domain.

4. The domain should allow structured expansion either through the addition of new elements to the domain to create variant domains, or through the use of differing scenarios within the constraints of the base domain. The benefits of this approach are twofold; firstly it enables the basic assumptions in the original domain to be relaxed or altered whilst minimising the amount of new development,

and secondly it allows the scope of the system to be widened as new areas of dialogical interest are discovered. The intuition behind this requirement is that no single problem domain will enable a full analysis of all dialectical games in all the dialogical contexts in which they may be used. However, in the long term, the interpolation of results from evaluating dialectical games against a range of domain problems will deliver a comprehensive picture of the capabilities, performance, and suitability of a game both to and in a range of circumstances.

Such a problem domain could be considered a *drosophila* for computational dialectics, much as the fruit-fly is considered a *drosophila* for the biological sciences, and Chess is considered a metaphorical *drosophila* for A.I. The problem domain that has been adopted for use in MASADA is that of a graph colouring problem. The four-colour problem [17] is well known and asks whether any map can be coloured using only four colours such that no two neighbouring regions share the same colour. It was subsequently demonstrated that this is the case by Appel *et al* in [4, 5]. In graph-theoretic terms each region of a map can be considered to be a vertice in a graph and each border between regions can be considered an edge between those vertices. It may then be asked, given a connected planar graph, whether only four colours are required to assign each vertice a colour such that no two vertices that are connected by a single are assigned the same colour. This can be generalised to non-planar graphs to ask whether the graph can be coloured using only *n* colours with the same stipulations. The translation from map colouring to graph colouring is illustrated in figure 5.2.

It is unlikely that any single problem domain will prove to be sufficient for a comprehensive study of dialectical games given the range of contexts in which dialogue can occur. As such the graph colouring problem domain is the first of a suite of problems, in the style of the 'Thousands of Problems for Theorem Provers' (TPTP) library [124], that will be formulated for the investigation and evaluation of computational dialectics. A characterisation of a graph colouring problem was therefore developed and situated in a software component named Colloqui.

5.2 Colloqui

The Colloqui component encapsulates internal agent capabilities related to knowledge and reasoning about communicative acts. The responsibilities of the Colloqui component are fourfold:

1. to represent the agent's knowledge,

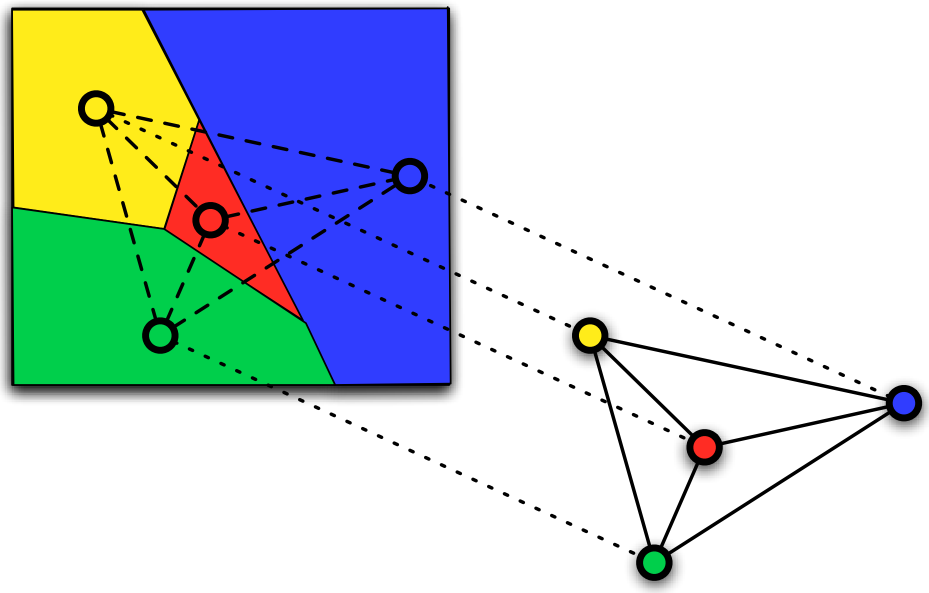

Figure 5.2: The translation between a map and a planar graph

2. to represent arguments constructed from the agent's knowledge,
3. to enable the agents to reason about their actions, and,
4. to enable the agents to communicate with each other to achieve their goals.

To achieve these ends Colloqui incorporates a knowledge representation scheme, an argument representation scheme, a reasoning process and an instance of the A4A as illustrated in figure 5.3. These elements are used to co-ordinate the process of:

1. handling incoming messages from other agents,
2. preparing and sending relevant responses,
3. keeping the knowledge base current in light of received messages.

It should be noted that this approach is similar to the MAS based argumentation framework due to Black [15] which investigates inquiry dialogues applied in a medical setting. Black creates a MAS framework which supports two types of inquiry dialogue, the first investigates the search for a justification for an argument from a set of arguments, and the second investigates the search for a set of facts that can act as justification for a given claim. Black's framework however is constrained solely to investigating inquiry dialogues whereas the current work is concerned with investigating arbitrary dialectical games.

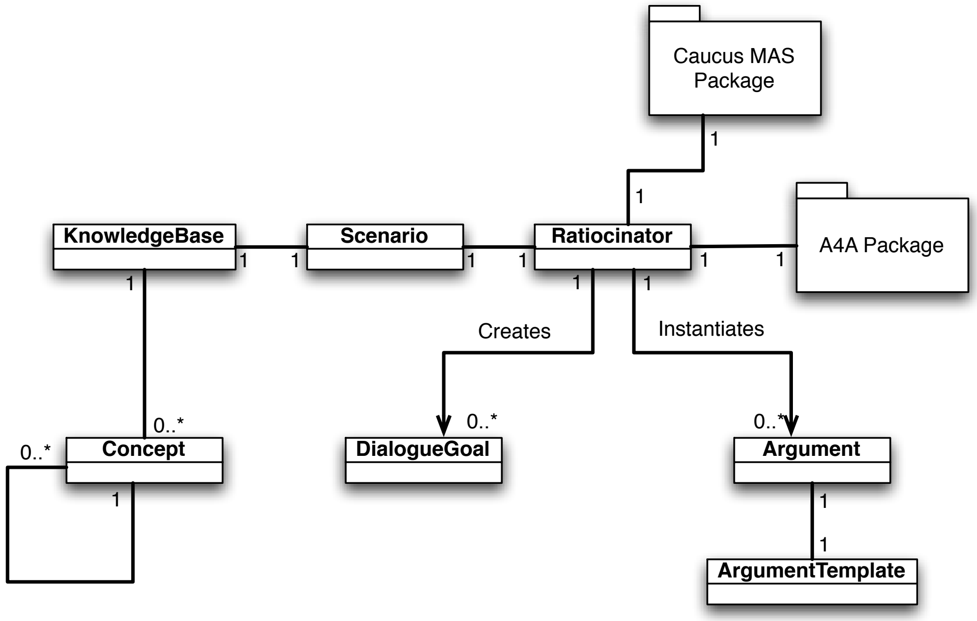

Figure 5.3: Colloqui Class Diagram

5.2.1 Knowledge Representation & Expression

Individual agent knowledge is represented in Colloqui using a version of the frame representation scheme introduced by Minsky in [85]. The specific implementation is based upon the object-oriented approach demonstrated by Brachman and Levesque in [16]. Elements in the problem domain, such as agents and colour states, are represented as individual frames of knowledge, termed *concepts*, which are in turn used to record the current state of the represented objects. It was considered that such a representation was appropriate because the problem domain is both well understood and well defined. The problem domain itself is based upon a characterisation of the graph-colouring problem. This domain is named GC_0. The subscript indicates the version of the domain that is being used and provides a basic naming structure for the implementation of additional domains based upon graph-colouring. GC_0 describes a range of basic parameters which are implemented in the Colloqui component. An agent's knowledge is loaded during initialisation and is specified using a scenario specification file that details specific values for the parameters of the domain and provide a convenient method to initialise both agent specific, and system wide, domain dependent parameters. The parameters available in GC_0 are;

Colours Each agent possesses a colour state attribute. Colour states are selected from a fixed pool. The number of available colour states is defined in the scenario.

Relationships Each agent maintains relationships with a known set of other agents. Graph-theoretically, if agents correspond to vertices then a relationship is defined as an edge joining two vertices, i.e. two agents have a relationship if the vertices to which they correspond are adjunct. Where two vertices are joined by exactly one edge those vertices are called *neighbours*. Neighbours are only ever connected directly by one edge. In GC_0 relationships are initiated during system start-up and are fixed throughout the lifetime of the MAS.

Agent Knowledge At start-up an agent knows only its own colour state and the set of its neighbouring agents. Agent knowledge is also uncertain and dynamic. Whilst an agent can only ever be in a single colour state at any given time that colour state is mutable, it may change with time as the agent determines through dialogue that another colour state is more suitable. As a result the agents must reason with uncertain and dynamic information in order to achieve their goals. An agents knowledge is uncertain because although an agent can perceive the current state of its neighbours it cannot perceive the state of other agents to which it is not related. The agent may however learn of the existence and states of other agents during dialogues with its neighbours, who might in turn have learnt of the unrelated agent through earlier dialogues. The state of the unrelated agent may have altered in the meantime and as a result an agents knowledge is always uncertain with respect to other unrelated agents in the MAS.

Conflicts Conflicts occur when neighbouring agents are in the same colour state.

Goals When agents recognise a conflict with one of their neighbours they instantiate a dialogue goal whose aim is to resolve the conflict.

Actions The only action that GC_0 agents are capable of performing is to change colour state. Agents only change state if they are persuaded as a result of a dialogue.

Conflict Resolution When a conflict occurs it is necessary to resolve the situation between the *conflicting agents* in accordance with the agents goals. A conflict between two agents is resolved when one of them changes state. A state change can however cause an agent to brought into a new conflict with another of its neighbours. This occurs if the new state of an agent is the same as the current state of one or more of its neighbours. In this situation there are said to be no conflict-free states. An agent has no conflict-free states if it will be brought into conflict with at least one of its neighbours no matter which new state it changes to. In the situation where there are no conflict-free states, resolving one conflict through a state change will serve to cause a new conflict to occur. This situation can be characterised as a propagation of the original conflict through the MAS.

Communication Agents can engage in dialogue with their neighbours. This dialogue is regulated by the available dialectical game protocols specified in the A4A. In GC_0 dialogue is the only means which agents can use to influence the actions of other agents and thereby bring about a conflict resolution.

Arguments Agents use their knowledge of relationships and conflicts to produce arguments. Arguments are expressions of support relations between concepts in the agents knowledge stores. There is a preference ordering over the arguments that an agent can produce which enables an agent to determine which arguments are acceptable. Provided that an agent has sufficient knowledge and that the current state of the system is such that an argument can be produced, then whenever a move requires production of some argument the agent should be able to furnish such an argument from its knowledge store.

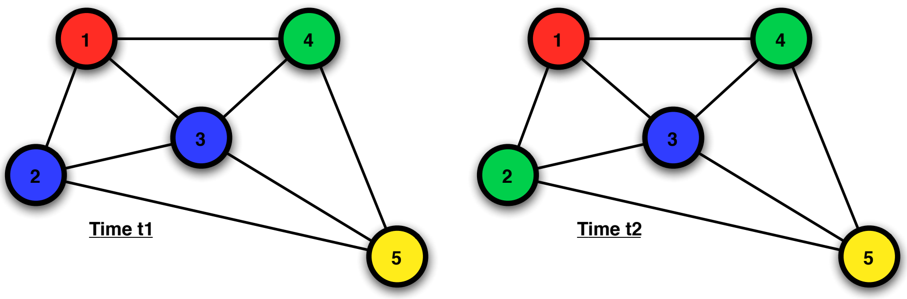

Figure 5.4: The same MAS-Graph at timesteps t1 and t2 illustrating a state change between the two timesteps.

Some of these parameters can be illustrated by considering the graphs in figure 5.4. For ease of comprehension agent states have been transcoded to the colours red, green, blue, and yellow, which form the pool of legal colours. With respect to agent 2, its current colour state is blue and the only colour available to agent 2 is green because any other colour change will lead to further conflicts. Agent 2 has three relationships such that its neighbours are agent 1, agent 3, and agent 5. At time-step t1 Agent 2 and Agent 3 are in conflict. If Agent 1 detects the conflict with its neighbour then it creates a goal to resolve that conflict. To resolve the conflict either agent 2 or agent 3 must effect a change state action. At t2 there are no longer any conflicts in the graph because agent 2 has performed a change state action to a conflict-free state. In GC_0, one constraint is that communication between agents is only valid if the communicating agents are neighbours hence agent 2 may only initiate dialogues with agents 1, 3, and 5. The remainder of this section explores how agents express statements about the state of the graph, given their curren knowledge. In the next section the process of argument representation and instantiation is expressed which builds upon the basic statement expression capabilities of the agents.

The knowledge base supports three basic interactions which enable agents to;

1. Update their knowledge base.
2. Retrieve the values stored as a part of a given knowledge concept.
3. Retrieve a predicate statement that expresses the current state of a concept.

Knowledge base updates enable the values associated with a given concept to be updated as new information is discovered. For example when an agent learns that one of its neighbours has changed colour state then the agent must update its knowledge base to record that fact. Similarly an agent may need to add a new agent frame to its knowledge base to record knowledge of its neighbours neighbours, and so on.

The values associated with a concept, for example the current colour state of an agent can be expressed in two ways. The first way is through the raw value associated with the concept. For example, an agent concept records the current state of the associated agent, e.g. that the colour state of agent 1 is blue, where blue is the value of the colour state. When an agent is performing reasoning about its actions and communicative acts, the agent's reasoning process needs access to this kind of information. The knowledge base thus facilitates raw access to this information to retrieve, for example, the actual value associated with a given agent's colour state. The other access mechanism is through the use of statements. The concepts and their associated values can be expressed in the form of statements for use in communication with other agents. Agents can thus express their knowledge in the form of either basic predicate statements which are constructed from a predicate and one or more terms, e.g. *predicate(term*$_1$*, term*$_2$*, term*$_n$*)*, or as compound predicate statements. Agents are also able to parse any received predicate statements and extract any information which can then be compared to existing knowledge concepts to determine agreement or used to update the knowledge base with new information. The set of statements that an agent can express is directly related both to the range and number of concepts stored in the knowledge base and the specific values associated with those concepts at any given instant. This means that if an agent can express four different statements for a given concept type then the maximum number of statements that the agent can construct is four multiplied by the number of instances of the concept. The actual number of statements that can be constructed at any given point may well be less than the maximum possible due to the actual state of each concept, for example if agent 1 knows that agent 2 is a neighbour of agent 3 but does not know agent 2's current colour state then agent 1 can instantiate and express the statement that agent 2 is a neighbour of agent 3, e.g. neighbour(agent(2), agent(3)), but cannot instantiate a statement expressing the colour state of agent 2 until it knows the colour state of agent 2. Colloqui does not therefore assume a closed world semantics with respect to knowledge. Both knowledge access mechanisms, value access and statement access, are implemented through stored procedures associated with each knowledge frame.

In GC$_0$ the range of available statements that an agent can possibly express fall into three groups. The first group of statements concern capabilities and actions that the agents can perform and are termed *practical statements*:

change_state(agent(X)) is an imperative statement which expresses that the agent identified by 'X' should change state. In GC$_0$ the only action that an agent can perform is to change its own colour state so this statement is used in the agents practical reasoning.

The next group are termed *primary statements* and consists of statements about the state of the environment, its member elements, and relationships between them:

conflict(agent(X), agent(Y)) which expresses that agent ‘X’ and agent ‘Y’ are in conflict.

colour_state(agent(X), state(Z)) which expresses that agent ‘X’ is in colour state ‘Z’.

neighbours(agent(X), agent(Y)) which expresses that agent ‘X’ and agent ‘Y’ are neighbours.

stability(agent(X), locale(Z), i) which expresses that agent ‘X’ has stability ‘i’ in locale ‘Z’. ‘Z’ is a number which indicates the size of the locale around an agent. A locale is defined as a sub-graph of the MAS consisting of all agents connected to the origin agent ‘X’ or their neighbours to the depth ‘Z’.

more_conflicts(agent(X), agent(Y)) which expresses that agent ‘X’ and agent ‘Y’ are in conflict.

num_conflicts(agent(X), j) which expresses that agent ‘X’ has ‘j’ conflicts.

less_stable(agent(X), agent(Y), locale(Z)) which expresses that agent ‘X’ is less stable than agent ‘Y’ in the locale of the same depth ‘Z’ around each agent ‘X’ and ‘Y’.

Agents can also construct statements that express relations between primary statements. These are *secondary statements* because they rely upon values that are established whilst constructing primary statements. However they are very useful when an agent is forming a valid argument:

greater_than(i, j) which expresses that ‘i’ is greater than ‘j’.

less_than(i, j) which expresses that ‘i’ is less than ‘j’.

Notice that there is a distinction between the basic and non-basic knowledge [1] held by an agent. Basic knowledge concerns agent states and agent relationships, concepts whose values are directly evident and can be verified through inspection of the MAS. Elements of non-basic knowledge are those concepts that are derived from knowledge of basic concepts, for example derivation of the existence of a conflict between two agents, or calculation of an agents stability.

Consider the graph illustrated in figure 5.5 in which agent 2 and agent 3 are in conflict. Both agents may express the statement change_state(agent(X)) respectively such that agent 2 may express the statement “change_state(agent(3))” for example. If the graph illustrated in figure 5.5 is considered as a representation of the current knowledge of agent 2 then the following list illustrates some of the statements that the agent is able to instantiate according to the forms outlined earlier:

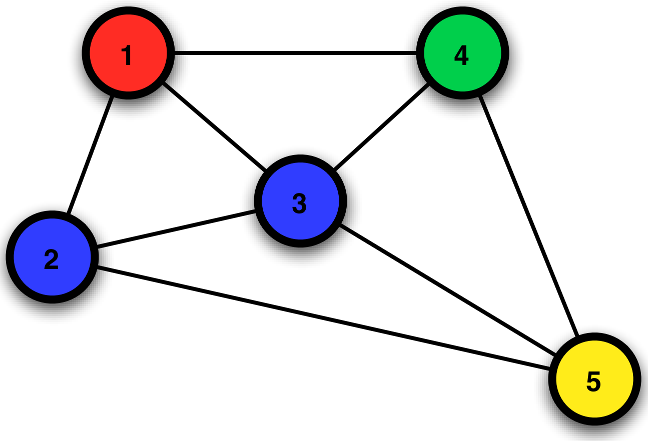

Figure 5.5: A conflict exists between agent 2 and agent 3.

conflict(agent(2), agent(3))

colour_state(agent(1), state(red))

neighbours(agent(2), agent(1))

stability(agent(2), locale(1), 1)

more_conflicts(agent(2), agent(1))

num_conflicts(agent(2), 1)

less_stable(agent(2), agent(1), locale(1))

Note that for each statement form there may be a number of instantiations of that statement dependent upon the state of the MAS. For example agent 2 could utter five different colour_state statements to account for the perceived colour state of each agent that agent 2 has knowledge of.

The frame-based knowledge-representation scheme used in Colloqui is sufficient to model the elements of the graph colouring domain and can be extended to include new elements present in enhanced graph-colouring domains.

5.2.2 Argument Representation & Instantiation

Certain moves within a dialectical game may require a player to utter a statement that supports or is otherwise related, in some argument theoretic fashion, to a previously uttered statement. For example, in the game DC, after a challenge of a statement q a player can utter some statement p in defense, and the argument $p \rightarrow q$ is entered into the uttering player's commitment store (amongst other effects). This requires that there is a mechanism for relating the statement p and the statement q, and the associated frames to which p and q refer, such that p supports q.

The knowledge representation portion of Colloqui does not directly include any argument theoretic relations as a part of the knowledge base but does support an overlaid argument representation scheme. This overlaid scheme uses *argument templates* to relate individual concepts stored in the knowledge base according to the elements specified in the template. This enables arguments to be specified externally to the knowledge base and for the set of available arguments to be altered at runtime. Argument templates are analogous to argumentation schemes, a popular method for describing the stereotypical form of real-world arguments. A typical argumentation scheme in the Walton tradition [132] identifies the premises and conclusions of an argument in terms of an abstract form that they might take in an instantiated argument. For example, take the following argumentation scheme for the *Argument From Sign* [137, pp. 113];

Specific Premise: A (a finding) is true in this situation.

General Premise: B is generally indicated as true when its sign, A, is true.

Conclusion: B is true in this situation.

To instantiate an argument according to this scheme, B can be taken to stand for "a bear has passed this way" and A can be taken to stand for "there is a bear print on the path". If A is true and there does happen to be a bear print on the path then it can generally be taken as true that B is true and a bear has passed that way, leaving its paw print on the path in the process. Notice that this argument is defeasible, signified by the use of the word "generally", and that given extra information an alternative conclusion to B could be drawn. If a person with a bear print stamp were to be placing fake bear prints on the path as a practical joke and this became known then the argument instantiated from the argument from sign could be defeated with the additional knowledge that the paw prints are fake. Such argumentation schemes have been applied successfully to argument analysis, for example in the Araucaria argument analysis software [104], and in the classification of natural arguments [56], and have been suggested for use in A.I. as a way to guide the argument generation process [107], as planning operators in domains such as natural language generation [106], and as a way to partition a knowledge base into concepts relevant to the task at hand [105].

An argument template can be considered as a semi-instantiated argumentation scheme for use in a particular knowledge domain. Templates guide the argument construction process for stereotypical arguments within the GC domains by providing parameterised templates for particular arguments that can be instantiated with information from an agents knowledge of the current state of the MAS. This allows flexibility from an agent deployment perspective where the number of and structure of arguments can be varied as required in the circumstances. The most important aspect is that it enables the arguments available to an agent to be altered at runtime so divorcing argument representation and instantiation to some degree from the underlying codebase. However, because the arguments that an agent can construct are intimately tied to the concepts that the agent knows and can express, and those knowledge concepts are dependent upon a particular domain and scenario, it follows that arguments and the templates that group their stereotypical forms are also closely tied to the domain and scenario. This approach, using argument templates, has been developed to enable an entirely Java based, extensible argumentative dialogue system to be constructed for use within Java-based applications without the added complication, from a deployment perspective, of using more suitable tools, such as Prolog, to carry out the agents argumentative reasoning. This does not preclude however a bridge to a Prolog-based reasoner as a future development of the system.

Argument templates are represented using XML files instantiated according to the Document Type Definition (DTD) found in appendix A.1 and have the following basic form;

```
<?xml version="1.0"?>
<templates>
   <template id="template_id" level="value">
      <conclusion>
         <pred>identifier string</pred>
         <term>identifier string</term>
      </conclusion>
      <premises>
         <premise>
            <pred>identifier string</pred>
            <term>identifier string</term>
         </premise>
      </premises>
      <rebuts>
         <arg>
            <pred>identifier string</pred>
            <term>identifier string</term>
         </arg>
      </rebuts>
   </template>
</templates>
```

Arguments are instantiated by the Colloqui knowledge base according to the forms specified in the argument templates. Each argument template contains a conclusion and a number of premises, all of which are represented by statements in an instantiated argument. The Colloqui knowledge base provides a mechanism for constructing the predicate statements required to instantiate an argument from an argument template. This mechanism takes an identifier string that specifies a predicate and an identifier string that specifies each term and returns a predicate object if such an object can be instantiated given the current status of the knowledge base. When instantiating an argument, the sources of the predicate and term identifier strings for each statement are the relevant conclusion and premise parts of the argument template that is being instantiated. If a term identifier identifies another predicate then that other predicate statement is substituted for the identifier to form a compound statement.

When an argument template is constructed it is specified in terms of predicate statements that are explicitly supported by the knowledge domain. For example, the GC_0 domain includes the following statement: *more_conflicts(agent(X),agent(Y))*. If an agents knowledge is such that it can identify that agent(X) has more conflicts than agent(Y) then it can create and utter the statement at that point in the dialogue. An example argument template that has as its conclusion the "more_conflicts" statement is the following;

Premise$_1$: num_conflicts(agent(X), *i*)

Premise$_2$: num_conflicts(agent(Y), *j*)

Premise$_3$: greater(*i*, *j*)

Conclusion: more_conflicts(agent(X), agent(Y))

In this template the more_conflicts statement is expressed as the conclusion, an instantiation of which would be, *more_conflicts(agent(0),agent(1))*, for two agents named '0' and '1'. When an argument is constructed the Colloqui reasoning process examines the knowledge base and attempts to retrieve each of the statements representing each premise for the supplied conclusion by inserting the relevant variables from the conclusion statement into the premises. For example given the conclusion, *more_conflicts(agent(0),agent(1))*, premise$_1$ can be instantiated by replacing the 'X' variable with the corresponding value for the X variable in the supplied conclusion, which means that for the example X=0. Those variables which can be instantiated directly through substitution in this way are represented by upper case Roman letters in the template and are the essential variables needed to identify a statement in the knowledge base. Variables which are not necessarily known before instantiation are represented by lower case Roman letters. For example the variable *i* represents the number of conflicts that the agent represented by X has but the value of *i* is not reified until the num_conflicts statement is retrieved. By passing the num_conflicts predicate identifier into the getStatement method of the knowledge base with the argument 'X' the following statement will be returned (if X=0 and *i*=5), *num_conflicts(agent(0), 5)* which could be expressed in everyday parlance as "agent$_0$ has 5 conflicts".

If, given the current state of an agent's knowledge base, all of the premises and the conclusion of an argument template can be consistently instantiated as statements then the argument can be validly and completely instantiated and used during a dialogue. If any part of the argument is wrongly instantiated, because an agent's knowledge is outdated for example, then the instantiated argument is invalid and thus subject to attack. If any part of the argument cannot be instantiated then the argument is weakened and deemed incomplete. Such an incomplete argument is subject to attack from the agent towards which it is directed.

All individual argument templates can be diagrammed using the same basic form of the linked argument because premises do not provide sufficient support for the conclusion on an individual basis but do so when working in conjunction. Whilst there is some disagreement amongst researchers as to the utility of the argument diagramming method, for example, the linked argument can, in many cases, be viewed as an instance of *modus ponens* reasoning involving minor premise, major premise, and conclusion, argument diagramming does provide an easily comprehensible visualisation of argument structure which has commonplace usage in argumentation theory, and which does not assume a background in logic. Figure 5.6 illustrates the typical form of such a linked argument.

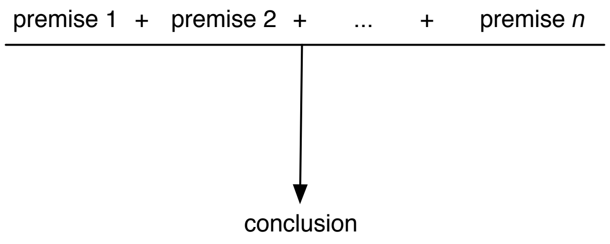

Figure 5.6: Argument diagram for a linked argument.

Argument templates can be chained together to create serial arguments whose premises incorporate the conclusions of other arguments. The general form of this kind of argument is shown in figure 5.7.

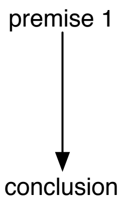

Figure 5.7: Argument diagram showing the form of a serial argument.

Many argument templates can be specified to provide independent support for a given conclusion. Such arguments, which consist of two or more premises providing independent support for a conclusion are known as convergent arguments whose typical form is diagrammed in figure 5.8.

When a set of templates includes convergent arguments, e.g. there are many arguments that support a given conclusion, a preference ordering is required to enable agents to determine how “strong” a given

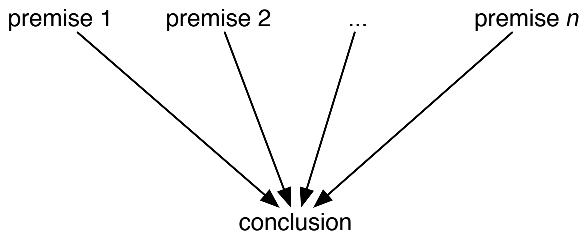

Figure 5.8: Argument diagram for a convergent argument.

argument is. This is is achieved through the level parameter of the template element which specifies a *preference level* for the associated argument. This approach is based upon the preference ordering approach due to Amgoud [2]. Arguments that have the same preference level are equally preferred whereas a higher level argument is considered preferable to a lower-level argument. Thus a rough-grained means by which to evaluate instantiated arguments is built into the argument representation. The following example argument template file fragment includes three argument templates;

```
<?xml version="1.0"?>
<templates>
   <template id="arg_0" level="0">
      <conclusion>
         <pred>change_state</pred>
         <term>agent(X)</term>
      </conclusion>
      <premises>
         <premise>
            <pred>conflict</pred>
            <term>agent(X)</term>
            <term>agent(Y)</term>
         </premise>
      </premises>
      <rebuts>
         <arg>
            <pred>not_change_state</pred>
            <term>agent(X)</term>
         </arg>
      </rebuts>
   </template>
   <template id="arg_1" level="2">
      <conclusion>
         <pred>change_state</pred>
         <term>agent(X)</term>
      </conclusion>
      <premises>
         <premise>
            <pred>less_stable</pred>
            <term>agent(X)</term>
            <term>agent(Y)</term>
            <term>locale(Z)</term>
         </premise>
      </premises>
      <rebuts>
         <arg>
            <pred>not_change_state</pred>
            <term>agent(X)</term>
         </arg>
      </rebuts>
   </template>
   <template id="arg_2" level="2">
      <conclusion>
         <pred>not_change_state</pred>
         <term>agent(X)</term>
      </conclusion>
      <premises>
         <premise>
            <pred>more_stable</pred>
            <term>agent(X)</term>
            <term>agent(Y)</term>
            <term>locale(Z)</term>
         </premise>
      </premises>
      <rebuts>
         <arg>
            <pred>change_state</pred>
            <term>agent(X)</term>
         </arg>
      </rebuts>
   </template>
</templates>
```

In the XML fragment there are three argument templates, *arg_0* which has a preference level of 0, and *arg_1* and *arg_2* which both have a preference level of 2. Both arg_0 and arg_1 support the conclusion that "change_state(agent(X))" whereas arg_2 expresses the argument that "not_change_state(agent(X))". Therefore arg_0 and arg_1 are convergent arguments which exist in a rebutting relationship with arg_2. In this case arg_0 is considered a "weaker" argument than arg_2 meaning that if arg_0 is presented in a dialogue then it can be defeated by the presentation of a rebutting argument at a higher preference level. A diagram illustrating this rebutting relationship can be seen in figure 5.9.

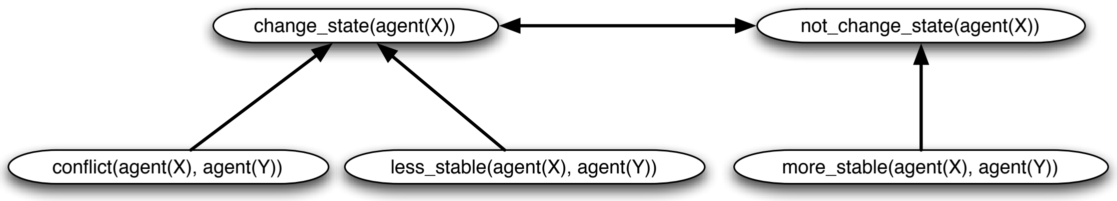
change_state(agent(X))
not_change_state(agent(X))
conflict(agent(X), agent(Y))
less_stable(agent(X), agent(Y))
more_stable(agent(X), agent(Y))

Figure 5.9: Argument diagram for the rebutting argument shown in the argument template file XML fragment.

This approach to the representation and instantiation of arguments according to the parameters specified in a template, is sufficient to provide the argument construction facilities required by the majority of extant games. However it is not sufficient to support the Toulmin Dialogue Game which bases many of its locutions, and the content of those locutions, upon the relationships between elements of an argument structured according to the Toulmin argumentation model. To support the Toulmin Dialogue Game requires that argument templates be extended to support the explicit representation of warrant, backing, qualifier, and rebuttal, in addition to the claim and grounds that are already accounted for by the conclusion and premises of the MASADA argument template schema. Without recourse to the extended elements of the Toulmin model a TDG dialogue can quickly reach an impasse where the locutors need to state, for example, the explicit warrant linking the grounds to a claim, but the required statement is not available because the argument representation aspect of Colloqui does not currently support this.

5.2.3 Reasoning

Whilst the A4A specifies to an agent the form of legal utterances through the Utterance archetype interface, it does not specify how each archetype can be instantiated as an utterable locution containing content related to the agents knowledge domain, and which can be subsequently communicated to another agent. Similarly the knowledge representation aspect of the Colloqui component does not do anything more than enable an agent to record certain knowledge concepts related to the agents domain, and the argument construction aspect of Colloqui only assembles arguments based upon the existence and state of knowledge based concepts. What is required is a process to tie all these elements together;

1. to determine initiatory goal conditions
2. to initiate dialogues
3. to instantiate utterances from archetype and content
4. to to send messages to other agents
5. to parse received messages
6. to update knowledge base on receipt of new information

It is the Colloqui reasoning process, known as the *ratiocinator* that ties all these elements together. The ratiocinator supports a mixture of epistemic reasoning, in which agents reason in relation to their knowledge and beliefs about their environment and other agents within that environment, and practical reasoning, in which agents reason about and with regards to their actions and capabilities. There is much recent work into combining these aspects of reasoning with argumentation, see for example Atkinson [6] which presents an argument scheme based account of practical reasoning for use in agent argumentation , and Prakken [99] which presents a combined formalisation for interleaving sceptical epistemic reasoning with credulous practical reasoning. Similarly in the case of agents built around the Colloqui component, because there are actions that the agent can perform, and beliefs that the agents possesses, agent must utilise both aspects to act effectively in the MAS. Unilateral action cannot attain a good outcome except through happenstance because any given change in an agents state could lead to further conflicts with other agents, a situation that could be avoided through careful negotiation, persuasion and deliberation. It should be noted also that whilst all agents in the current implementation have the same goal, to resolve conflicts, and there is no cost to an agent in changing state, the framework discussed here was designed to be extensible, to enable agents with different, even competing and conflicting goals, to engage in dialogue. Similarly, the framework is designed to support enhanced scenarios in which, for example, there is a cost to an agent whenever it change state. Such enhancements to the current base implementation would provide a basis for more advanced argumentative interactions.

Reasoning in Colloqui is based around three factors, character, tendency, and dialogue role. These factors give an agent very basic control over how it responds during a dialogue, in effect, how it plays the game. Two broad agent characters are supported, credulous agents that accept new information they receive and sceptical agents that require supporting reasons before they accept any new information. likewise two broad agent tendencies are supported, verbose agents that tend to respond during a dialogue if they have a response that they can make. Laconic agents will tend not to respond even when there are legal statements that they can make, although this is subject to their internal goals. If an agent has a goal that they wish to achieve such as a commitment that they want their opponent to incur and they have not yet attained that goal then the agent will respond so long as it can still muster arguments to support its position. These characters are not absolute though, it is not the case that a laconic agent will never respond however a laconic agent won't respond just because it can. The laconic/verbose tendency and the credulous/sceptical characters are further complicated by the role which the agent occupies in a given dialogue because this can affect how the agents enact their tendencies and characters. The initiator of a dialogue will have some goal that they are trying to attain. A verbose agent in the initiator role may initially attain its goal, for example acceptance of its initial thesis, however if it continues the dialogue for too long circumstances in the MAS may alter and the respondent may retract its commitment. This may occur because whilst two agents are engaged in dialogue with each other, the MAS continues around them and the circumstances under which the initial commitment was gained may alter. In this circumstance it is prudent for an initiating agent to continue the dialogue only as long as necessary. Characters, tendencies and roles all inform the most basic tactics that agents use when they engage in a dialectical games. As a result the way in which a game proceeds and the resulting dialogue will be affected. This process is illustrated in figure 5.10.

The goal handling process is illustrated in figure 5.11. It begins with checking the knowledge base for conflicts. If conflicts exist then a new dialogue goal is created for each conflict. For each dialogue goal a new A4A dialectical game is initiated and the set of archetypal initial utterances is retrieved. As discussed in the section on Colloqui knowledge representation the only goal supported by GC_0 is a

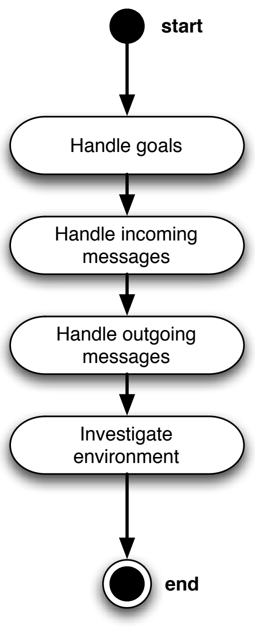

Figure 5.10: An overview of the Colloqui Ratiocinator Activities

"resolve conflict" goal. The only method for resolving a conflict is for one of the conflictees to change state so the resolve conflict goal is associated with the change state action. The change state action is associated with the "change_state(agent(X))" statement so this is supplied as the initial content of the first move in a dialogue. This method gives the agent an "entry point" into the knowledge base that relates a given dialogue goal to a relevant statement. An instantiated utterance is built according to a locution type identified in the set of initial utterance archetypes and the initial content statement. A message is then built which has as content the initial utterance. This message is then added to the outgoing message queue ready for delivery to the recipient agent.

Once goals have been created and handled to deal with any conflicts, the next task an agent performs is to handle incoming messages. This is the crux of an agents rational dialogical behaviour. The broad process is outlined in figure 5.12. Firstly, any received state update messages are parsed and the knowledge base updated. Next any dialogue messages must be handled. This requires the utterance in the message to be evaluated and the transcript to be updated. This involves the following process;

1. Retrieve the relevant move specification, for each locution expressed in the utterance, from the

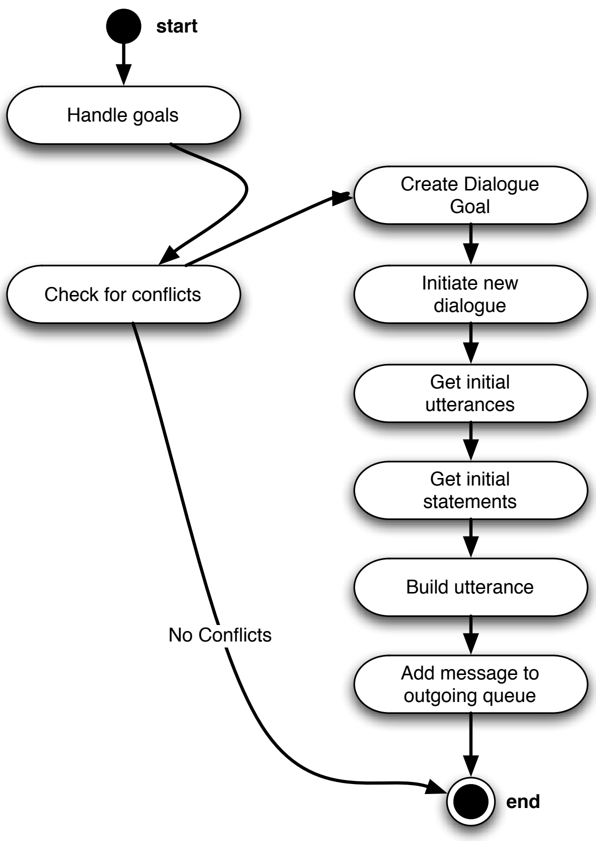

Figure 5.11: An overview of the Colloqui Ratiocinator goal handling process

current schema.

2. Applying any effects to the state of the game that the move specifies. Such updates might involve commitment store updates or player-role assignments.

3. If the move specifies mandatory responses then they are retrieved and the required content is assembled. Content can consist of new statements unrelated to existing statements within the dialogue, existing statements that have been previously uttered, or new content with an argument-theoretic relation to a previously established statement.

4. If there are no mandatory responses then the set of legal moves at that point is retrieved instead.

5. Once the set of move types and available content is retrieved an utterance can be constructed and inserted into a new message before being added to the outgoing queue.

Once incoming messages have been handled, the outgoing messages queue is dealt with. This is more straightforward than the incoming message queue and simply requires the agent to work through the outgoing message queue sending any messages in the queue to their relevant recipients. Once the outgoing message queue has been dealt with the agent creates and sends state request messages to the environment agent to find out about the current state of its environment. This enables the agent to determine whether there have been any changes to the environment during the last ratiocinator cycle.

One aspect that is currently not sufficiently well dealt with in the Colloqui component is the notion of utterance validation. Agents do not check whether the game moves expressed in an utterance conform to the rules of the game. The assumption that Colloqui relies upon is that all utterances are correctly formed by the uttering agents. This needs to be rectified in a later version so that agents receiving malformed or nonconformant utterances can reject the utterance and request a valid one, possibly also applying some sanction for uttering illegal moves. Two approaches to ensuring that the moves that are considered within a dialectical game are valid and legal are, firstly to utilise a concurrent meta-game that players can use to indicate problems within the issue dialogue. A second approach would be to design new dialectical games that include rules to facillitate recovery from rules breaking and malformed utterance episodes. The inclusion of such facilities is necessary in a real world MAS deployment but for the purposes of exploring dialectical games using dynamic knowledge the Colloqui component is sufficient.

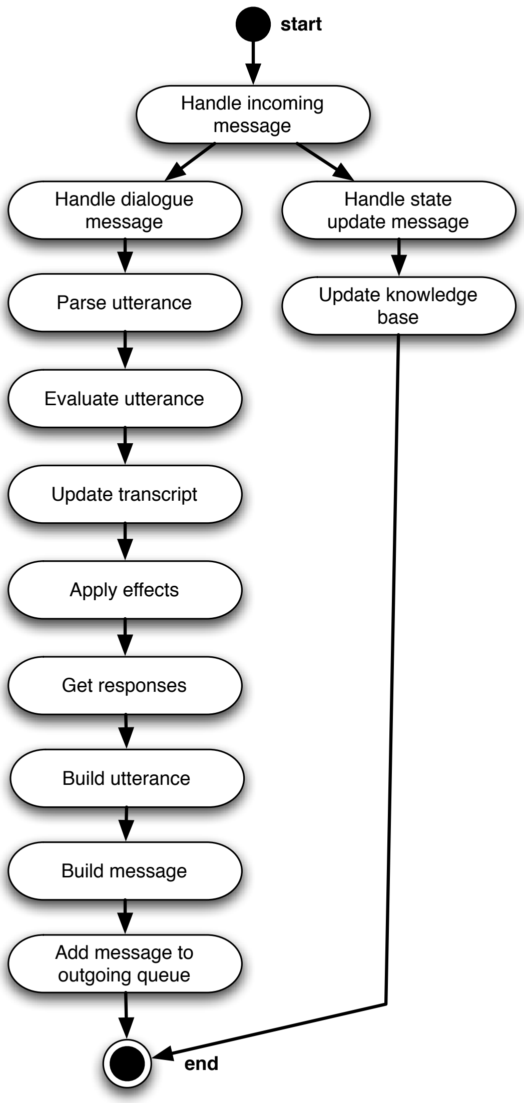

Figure 5.12: An overview of the Colloqui Ratiocinator incoming message handling process

5.3 Caucus

The agent level responsibilities of MASADA are encapsulated within the Caucus component. Caucus supports a heterogeneous society containing three types of agent, a core agent whose capabilities are specified in order to fulfil the *raison d'etre* of the MAS, an environment agent that maintains a representation of the state of the agent society and environment, and a graphical user interface (GUI) agent that presents MAS information to the end user.

The agent platform used to build the Caucus component is the Jackdaw University Development Environment (JUDE) developed by Calico Jack Ltd [67]. JUDE v1.0 is an entirely Java based lightweight, flexible, and industrial strength agent development platform that takes a modular approach to agent development. Individual JUDE agents consist of a core module, provided by the agent framework, which is extended via loadable modules to provide domain specific capabilities. The core module provides an agent with the ability to advertise itself in the MAS, to find other agents according to their advertisements, and to communicate with those agents that are found. To facilitate this the JUDE framework provides comprehensive agent discovery, agent lookup, and agent notification facilities. The capabilities of each type of Caucus agent are thus encapsulated into individual modules, dialogue capabilities into the core agent module, environment tracking into the environment agent module, and gui display capabilities into the GUI agent module. These modules are loaded into JUDE agents at runtime according to the requirements of a specified JUDE agent specification file. This approach enables new functionality to be loaded into or removed from an agent at runtime and also for existing functionality to be extended by substituting an enhanced module for an existing module. Figure 5.13 illustrates the modular approach used in JUDE.

Figure 5.14 shows the structure of a Caucus based MAS containing four core agents, a single environment agent, and a single GUI agent. Relationships between agents are defined by communication pathways such that if two agents can communicate then they are related. The type of communication that can occur between any two agents defines the type of relationship that exists between them.

5.3.1 Core Agent

Core agents facilitate the core functionality of a Caucus MAS through encapsulation of an instance of a Colloqui component. Figure 5.15 illustrates the class structure of a Caucus core agent. Each core agent

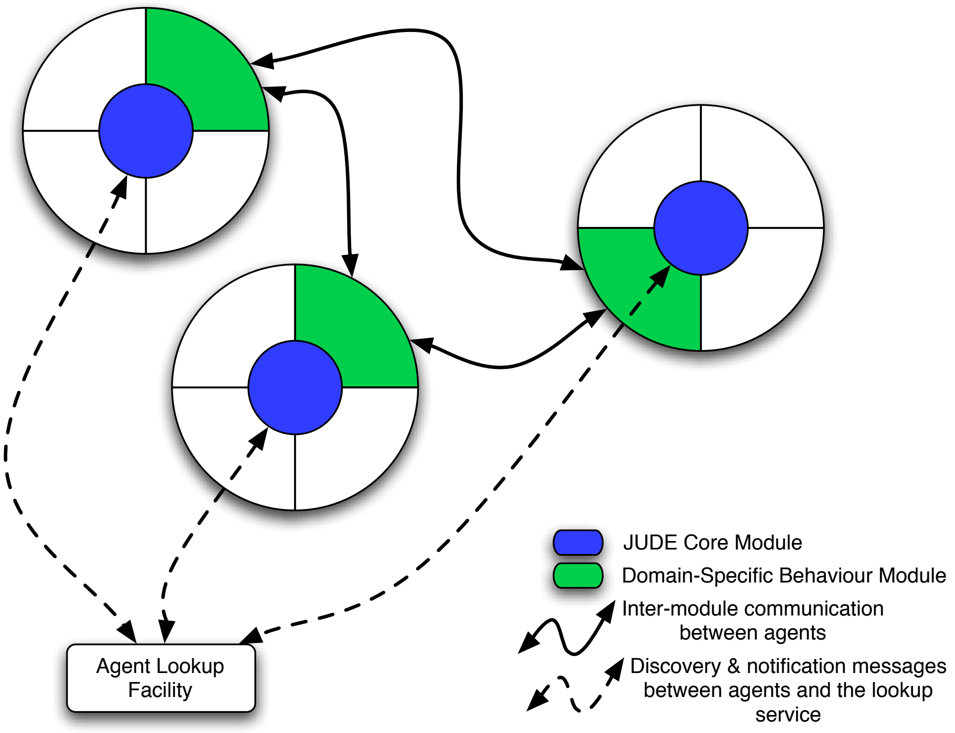
Agent Lookup Facility
JUDE Core Module
Domain-Specific Behaviour Module
Inter-module communication between agents
Discovery & notification messages between agents and the lookup service

Figure 5.13: The architecture of a simple JUDE MAS containing three agents and an agent lookup facility. Each agentcontains a JUDE core module and a domain specific behaviour module.

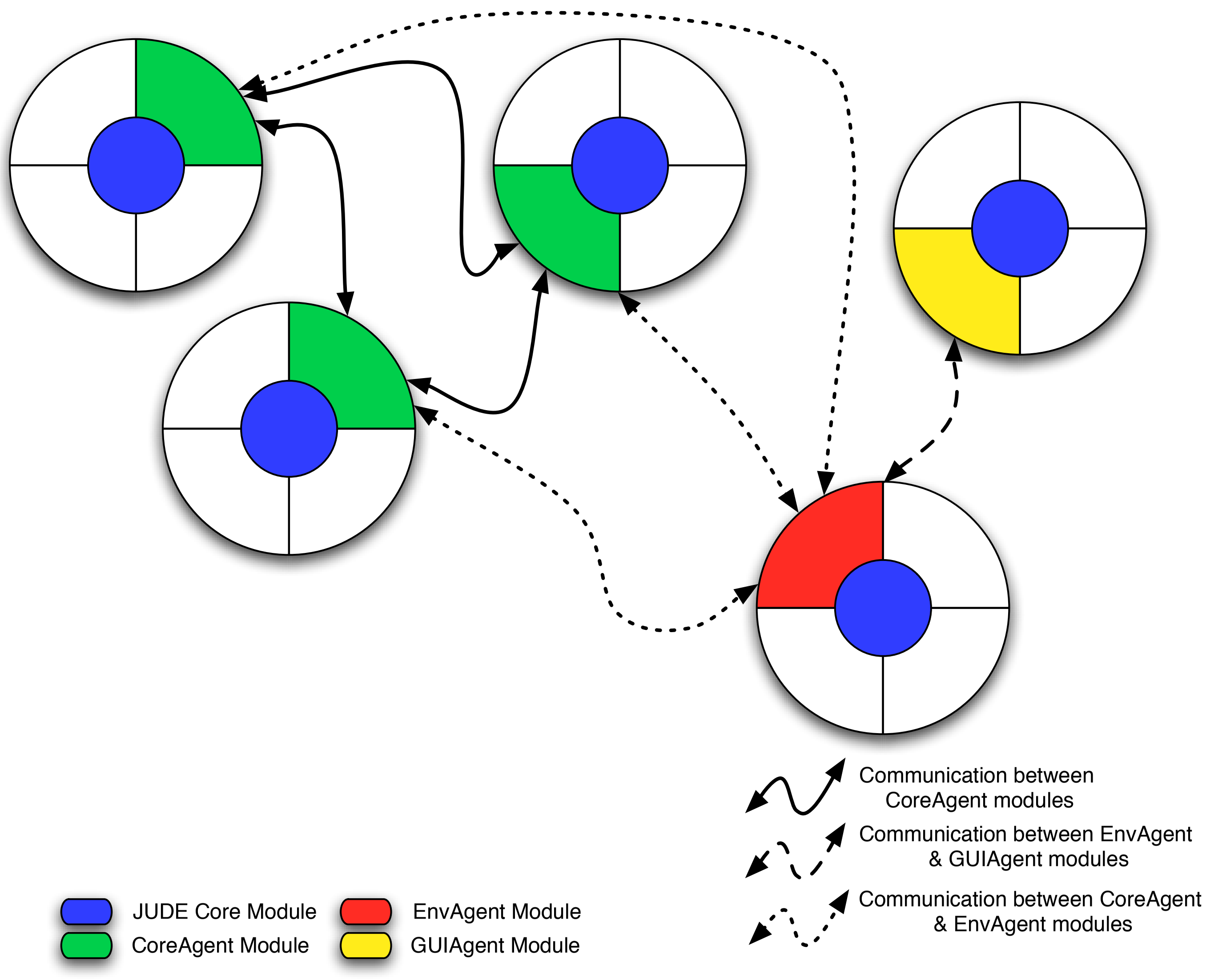

Figure 5.14: A simple Caucus MAS containing five JUDE agents, three of which have loaded CoreAgent modules, one of which has loaded an EnvAgent modules and one of which has loaded a GUIAgent module.

contains a single module, called the core module, which is used to facilitate inter-agent communication through message passing either to other core agents or to the environment agent. The core module maintains an incoming and an outgoing message queue, each of which can store message objects in a first-in, first-out structure. The responsibilities of the core module are related to message passing communications. The core module removes messages from the outgoing message queue and sends them to the agent specified in the message. Received messages are added to the incoming message queue ready for processing by the Colloqui component. Each core module runs in its own thread to ensure that it is responsive and able both to collate incoming messages and to send outgoing messages as required. On startup the core module creates a new thread in which the Colloqui reasoning process runs. The Colloqui process utilises the Comms interface to access the core modules message queues enabling the colloqui process to retrieve, parse and execute incoming messages, and to construct responsive messages for delivery to other agents.

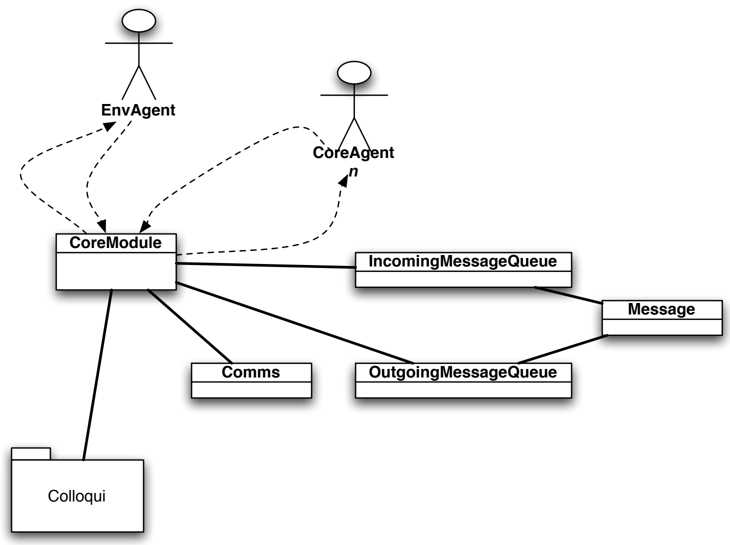

Figure 5.15: .

Messages are sent between core agents which hold the utterances of a given agent uttered during a particular turn of a dialogue. The following is an example message in which the payload is the statement *"init said: <question, (change_state(agent1)) >"* spoken by agent$_4$ and heard by agent$_1$ is as follows;

```
[MESSAGE ID: 0]
    |-> Dialogue ID: 7b56e:10df70b21ac:-7f88
    |-> From: agent4
    |-> To: agent1
    |-> At: Fri Sep 29 01:48:28 BST 2006
    |-> Content:
    |->init said:   <question, { change_state(agent1) }>
```

5.3.2 Environment Agent

Although individual core agents have the ability to perform actions, such as the execution of communicative acts or the alteration of their own internal state representation, their society is not situated within a directly perceivable environment. Core agents are not provided with senses to directly perceive state updates that occur in other agents or to identify which agents have relationships. These capabilities are provided in other ways. There are four basic ways to provide environmental sensing;

1. through sensors that measure physical changes in the real-world environment in which the (embodied) agent is situated,
2. through the use of an agent framework that provides an environmental simulation in which software sensors and actuators required by each agent can be modelled,
3. through the use of agent(s) designed to represent, record, and share environmental artifacts, and,
4. finally through direct communication between individual agents.

MASADA based agents are not situated within a physical environment so the first approach is not applicable, and the JUDE platform does not directly support environmental simulation so that approach is not applicable either. Both remaining approaches, the client-server based environment agent, and the peer-to-peer based inter-agent environment related communication are viable in MASADA, however the approach that is adopted here utilises an environment agent to represent the state of the world to the individual core agents rather than direct communication between those core agents. The rationale for this is that while the amount of communication required to update and retrieve environment data from a dedicated environment agent is similar to the communication load required to share pertinent environment data between individual agents, the dedicated approach offers additional benefits enabling information about the global state of the MAS to be collated and extracted for analysis in a more straightforward manner. It would have otherwise required processing to collate the same information if direct communication between core agents was used.

The Caucus environment agent maintains a representation of the core agents of the society. This representation includes data about each core agent including their name, relationships with other core agents, and the agents current state. This data is then used in three ways. Firstly, the core agents update the environment agent with their current status information and retrieve information about their neighbours. The amount of data that an agent can retrieve depends upon how deep the agents knowledge of the world is allowed to be. The default depth is to allow an agent to "*see*" one step away from themselves so that they can determine the state of their neighbours but not any agents that are further away. By varying the depth to which agents can see and the types of information that they can retrieve, agent societies can be investigated in which the amount of information available to the agents can be varied. The next way in which environment information is used is internally within the environment agent to enable global instrumenting and control over the entire agent society. For example, the information collated by the environment agent is used to determine when a solution to a graph-colouring problem has been found. The last way in which environment information is used is to supply information about the state of the agent society to the GUI agent.

The environment agent's representation of the MAS is updated and inspected by the use of communication between core agents and the environment agent. The environment agent supports three language primitives which are used to differentiate between two incoming message types and one outgoing message type. The incoming message types are a state update message type and a state request message type both of which originate from the core agent. State update messages are sent to the environment agent whenever a core agent has changed state and serves to update the environment agents representation of the society. The content of a state update message contains data about the new state of the agent who sent the message. An example state update message in which agent1 reports its current state, 0, to the environment agent is as follows;

```
[MESSAGE ID: 1]
|-> From: agent1
|-> To: env
|-> At: Tue Sep 19 23:23:56 BST 2006
|-> Content:
|-> 0
```

The state request message originates from an agent who wishes to determine the current state of another agent in the MAS. The following example state request message originates from agent1 and requests the state of another agent in the MAS named agent0;

```
[MESSAGE ID: 0]
|-> From: agent1
|-> To: env
|-> At: Tue Sep 19 23:13:54 BST 2006
|-> Content:
|-> agent0
```

The outgoing message supported by the environment agent is a state-info message type originating at the environment agent whose destination is a nominated core agent and whose content specifies the name of a core agent and the current recorded state of that agent.

5.3.3 GUI Agent

The GUI agent module, and by extension any agent that loads the GUI agent module, creates and displays a Java swing based graphical user interface to enable a user to view the current state of the MAS. This is illustrated in figure 5.16 which shows the GUI running under Mac OS 10.4.6. The GUI can also be used to view the transcripts of both ongoing and completed dialogues, to inspect the current scenario that was used to initialise the MAS, and to inspect the schema associated with the dialogues.

5.3.4 Summary of the MASADA Deployment

The development of MASADA demonstrated that even once a dialectical game is specified and represented within a software application, there are still many issues that need to be dealt with to achieve even basic argumentative dialogue capabilities if the software is to make use of an uncertain and dynamic knowledge base. Even once this basic capability is achieved it is unclear how to make good use of argumentative capabilities. Dialectical games such as those represented in the A4A only specify to an agent the form of legal utterances in a quite abstract fashion not what can actually be said in the form of fully instantiated utterances. Even once a set of potential utterances is constructed it is not clear which of those potential utterances should actually be uttered. Furthermore for any given potential utterance, there does not currently seem to be a way to determine whether it is the best utterance to use or even which criteria could suitably be applied to determining that “best” utterance. Selecting the best utterance is an issues of tactical play, the selection of the best move to make at the current juncture, and strategic play, the determination of the best sequence of moves to make in order to achieve some objective or satisfy some dialogical goal.

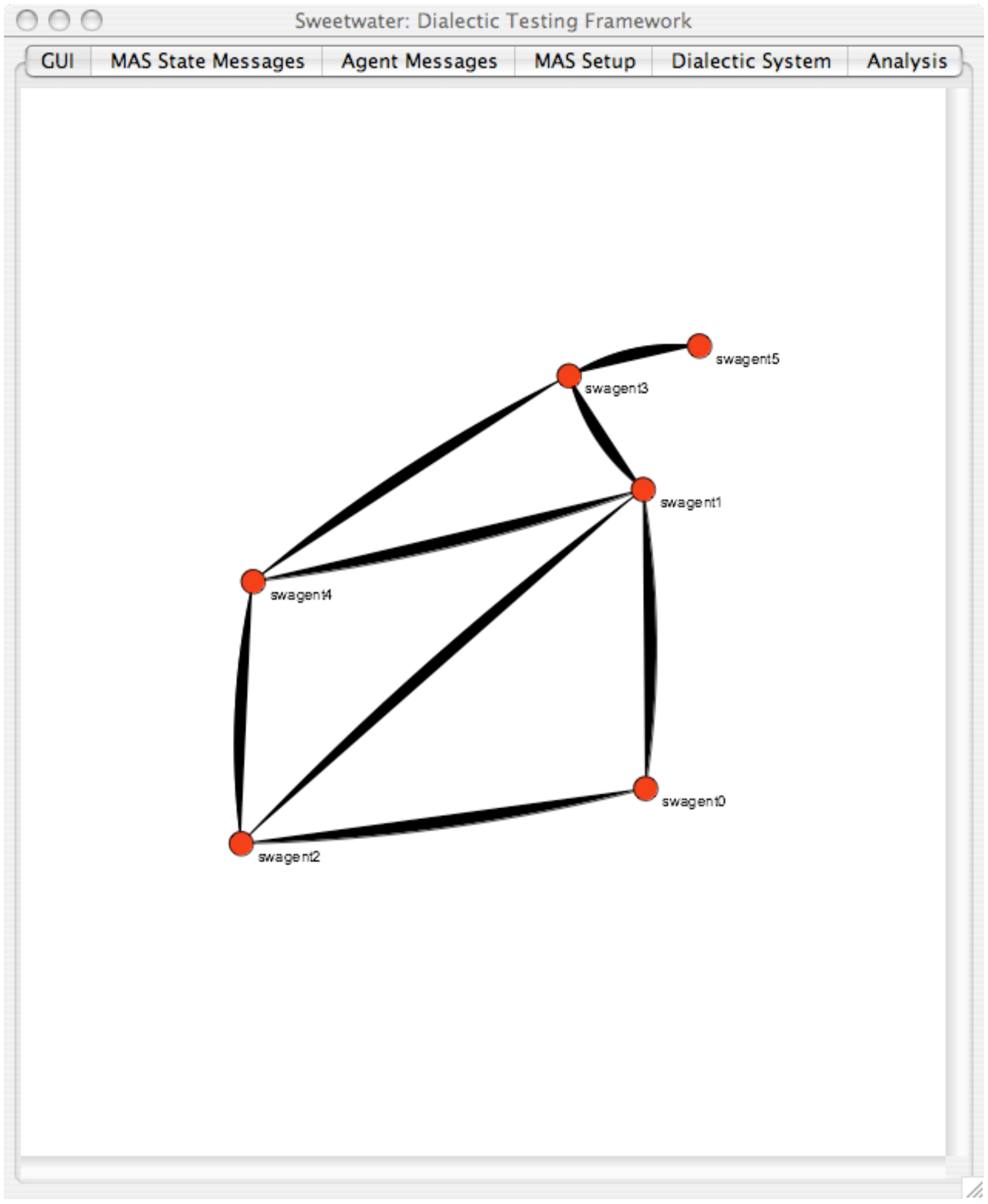

Figure 5.16: A screen-shot of the MAS display pane of the GUIAgent interface. Each agent is displayed as a vertice in the graph with relationships displayed as edges between those vertices. The colour of each vertice is achieved through a consistent mapping from the state associated with a given agent to a particular colour in the GUI display. The name of each agent is displayed alongside each vertex to enable particular nodes to be identified for further investigation.

5.4 The Information Exchange Project

In the EPSRC-funded "Information Exchange" research project[2], a MAS was created which utilised the A4A component to provide argumentative communication ability between *i*-Xchange agents. One aim of this project was to integrate elements of distributed planning, social-reasoning, and argumentation within a multiagent-based information management scenario. A second aim of the project was to explore the process of constructing a coherent agent architecture to underly agent planning, reasoning and arguing in social contexts. A brief overview of the *i*-Xchange architecture can be found in Kalofonos *et al.* [52].

An individual *i*-Xchange agent is based around a JUDE agent containing three modules in addition to its standard core module. Each of the three modules corresponds to an aspect of the research elements so there is a JUDE distributed-planning module, a JUDE social-reasoning module, and a JUDE argumentation module. One of each of these modules is loaded into a JUDE agent giving the agent the capability to create plans, reason about those plans in a social context, and to engage in dialogue with other *i*-Xchange agents. An interesting aspect of the *i*-Xchange system is that it uses a range of programming languages and agent platforms which are all integrated into the final agent. The JUDE planning module connects to an SWI prolog runtime [100] which runs the planner. The social-reasoning module connects over a communications bridge, called the JACK-JUDE bridge, to a JACK agent runtime [42] in which the social reasoning process executes. The JUDE social reasoning module acts as a message passing proxy to facilitate inter-module communication between the planning module and the dialogue module. Use of the JACK-JUDE bridge effectively enables JACK agents to be encapsulated within a JUDE agent module.

Figure 5.17 illustrates the various modules and their external language and runtime dependencies. In this case there are two agents loaded into the JUDE runtime, JUDE Agent 0 and JUDE Agent 1. Each of these agents loads a planning, a reasoning, and a dialogue module which in turn communicate with each other, in the case of the dialogue modules, with the JACK kernel in the case of the social reasoning module, and with the SWI Prolog runtime in the case of the planning module.

The aim of such an architecture is to draw together a number of disparate activities that are important in the context of agent argumentation. These include:

[2]The *i*-Xchange project was an EPSRC funded joint research project undertaken by the Universities of Aberdeen, Dundee, and Southampton under grant GR/SO3713

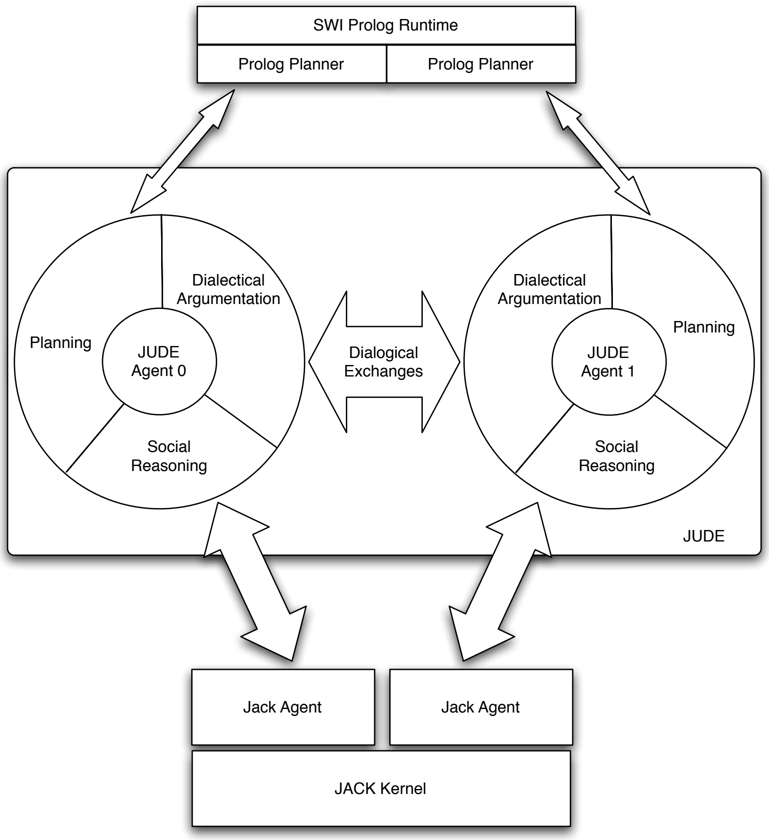

Figure 5.17: The *i*-Xchange Architecture

1. The ability to devise a plan of action. An agent may formulate plans that require agents other than itself to perform certain actions.

2. When a plan requires actions to be performed by an agent other than the originator of the plan, social reasoning is used to determine which agents should be selected to perform those actions. Such reasoning involves the determination of which agents are capable of performing the task, of those agents which are actually available to perform the task, and finally which agents can be relied upon to successfully perform the task.

3. Finally, once an agent has determined how to achieve its goal through planning, and which other agents are required to perform any actions involved in the plan, it is the responsibility of the dialogue module to communicate with the nominated agents and get them to agree to perform the required tasks.

These three elements form the basis of a coherent framework and software architecture for a MAS in which individual agents are able to perform a complete planning-reasoning-communicating-acting cycle as illustrated in figure 5.18. Notice that although communication is a type of action, because it is the central concern in this work it is handled separately to other forms of actions solely for the purpose of clarity between communicative and non-communicative action.

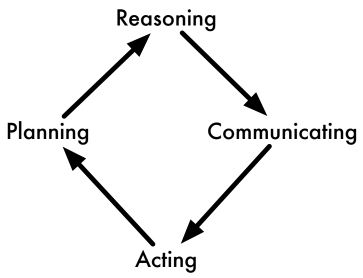

Figure 5.18: The cycle of capabilities within an *i*-Xchange agent.

Within any pair of *i*-Xchange agents there is a communication pathway, originating and terminating in a single planning module, which leads through each module of both agents. This pathway makes use of four inter-module communication interfaces and one inter-agent communication interface. The inter-module communication interfaces are as follows;

1. Service Interface - Used to communicate a new partial plan from the planner to the reasoner, and to communicate the acceptance or non-acceptance of such a partial plan by another agent back to the planner from the reasoner.

2. Dialogue Goal Interface - Used to communicate the goal of a new dialogue from the reasoner to the dialogue module, and for the results of any instantiated dialogue to be returned back from the dialogue module to the reasoner.

3. Proposal Interface - Used to communicate a received proposal from the dialogue module to the reasoner, and the acceptability of the proposal back from the reasoner to the dialogue module.

4. Evaluation Interface - Used to communicate a received partial plan from the reasoner to the planner for evaluation, and to communicate the results of such an evaluation back from the planner to the reasoner.

The communication pathway involving these interfaces is illustrated in figure 5.19. The pathway begins in the planning module of Agent 1 which is the initiating agent. A plan is decomposed by the planner into partial plans corresponding to third-party actions and each partial plans is communicated to the reasoning module in the form of a service request. For each service request, the social reasoning module nominates an agent who can perform the actions required and communicates both the service request and the id of the nominated agent to the dialogue module in the form of a dialogue goal. The dialogue module then initiates a new dialogue with the nominated agent, in this case Agent 2. During the dialogue a proposal is passed from Agent1 to Agent2 encapsulating the actions that Agent1 wises Agent2 to perform and this is passed to the reasoning module where any social issues are dealt with. The reasoning module then passes the partial plan to the planning module for evaluation. Agent2 evaluates the received partial plan and determines whether it is acceptable and the results of this evaluation are passed all the way back through each interface, eventually returning to the planning module in Agent1 where the path of communications originated. Notice that there is not necessarily a one to one correspondence between messages received at each interface. For example, for any single partial plan received by the social reasoning module a number of dialogues may have to be engaged in before an agent willing to accept the partial plan is found. Similarly for every dialogue goal received by the dialogue module there may be many messages that must be passed as a part of the dialogue in pursuit of the goal.

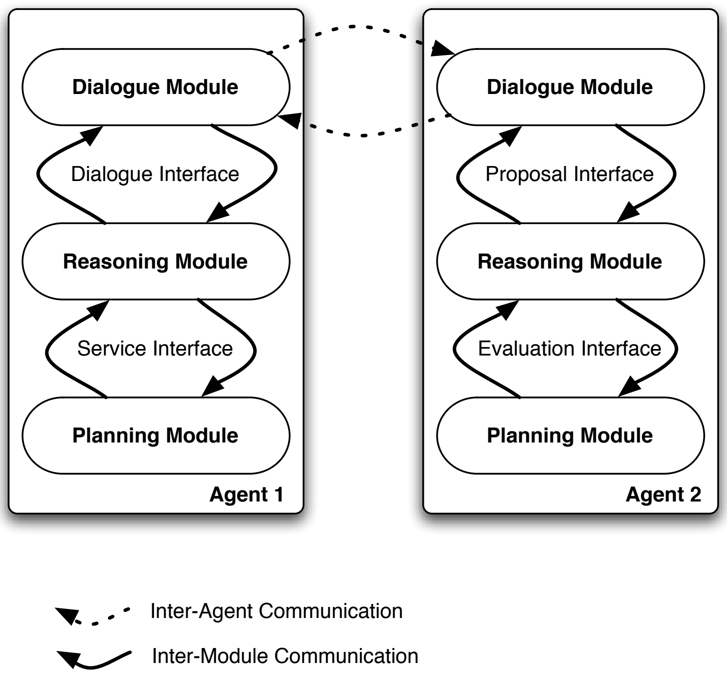

Figure 5.19: The path of communication between a pair of *i*-Xchange agents.

The internal details of the planning and social reasoning modules are explored in Kalofonos *et al.* [53, 54] for the planning module, and in Karunatilake *et al.* [55]. The dialogue module uses the A4A to manage dialogues through a dialectical game schema and to record the transcripts of dialogues.

The basic notion that the dialogue module assumes is that once a partial plan has been produced by the planner and an agent nominated who might agree to perform the actions associated with the partial plan then the nominated agent must be be interacted with to get the partial plan to be either accepted or rejected. The interactions that the dialogue module uses are communicative and these communicative acts are specified in an A4A dialectical game schema called $iXchange_0$. The following schema illustrates the basic dialectical game that *i*-Xchange agent engage in to determine whether another agent will accept a proffered partial plan;

Composition
Name iXchange$_0$
Turns < Strict, Single >
Participants { init, resp }
Stores
< CStore, Init, Set, Public >
< CStore, Resp, Set, Public >

Structure

< **Initiation** >
[-] Req: Turn = 0
[-] Eff: Response$^{init}_{Next}$ = <Initiate, (–)>

< **Termination** >
[-] Req: Transcript$_{Last}$ = <Affirm, (p)> ∨ <Deny, (–)> ∨ <Withdraw, (–)>
[-] Eff: Game$_{status}$ = complete

Interaction

< **Initiate,** {**–**} >
[1] Req: Turn = 1
[1] Eff: Response$^{Listener}_{Next}$ = < Acknowledge, (–)> ∨ <Withdraw, (–)>)

< **Withdraw,** {**–**} >
[1] Req: –
[1] Eff: Game$_{status}$ = complete

< **Acknowledge,** {**–**} >
[1] Req: Transcript$^{Listener}_{Last}$ = < Initiate, (-)>
[1] Eff: Response$^{Listener}_{Next}$ = < Enquire, (p)>

< **Enquire,** {**p**} >
[1] Req: –
[1] Eff: CStore^{Speaker} + {p}, CStore^{Listener} + {p}, Response$^{Listener}_{Next}$ = <Affirm, {p}> ∨ <Deny, {p}>)

< **Affirm,** {**p**} >
[1] Req: Transcript$^{Listener}_{Last}$ = < Enquire, {p} >
[1] Eff: CStore^{Speaker} + {p}

< **Deny,** {**p**} >
[1] Req: Transcript$^{Listener}_{Last}$ = < Enquire, {p} >
[1] Eff: CStore^{Speaker}+{¬p}, CStore^{Speaker} – {p}

The structure of this schema is linear enabling an agent to initiate a dialogue, and once initiated to gain acceptance or rejection with respect to the Statement “S”. In the basic *i*-Xchange scenario “S” is a partial plan that is passed between the agents. There are very clear termination rules. If any of three locution forms occur then the status of the dialogue is set to complete and neither interlocutor can utter any further locutions in the dialogue. The initiation rules of iXchange$_0$ specify that a dialogue must begin with the <*Initiate, (-)*> locution and the turn parameters specify that a single locution can be played per turn and that turns proceed in a strict alternation between the two players who are denoted “init” and “resp”. Once the initiate locution has been written by init, resp must reply with the <*Acknowledge, (-)*> locution. Once both of these locutions have been uttered and received init will proceed by uttering an enquire locution in which the content is the partial plan under question. If resp accepts the partial plan then the dialogue completes, according to the termination rules, with the partial plan in both the init and resp commitment stores. If resp declines to accept the partial plan then the dialogue completes with the denial of the partial plan being entered into resp’s commitment store. A dialectical structure diagram for the iXchange0 schema can be seen in figure 5.20.

A dialogue can have two different outcomes one in which the partial plan is accepted and one in which it is rejected. Each of these outcomes has a different effect with respect to the other modules in each agent;

partial plan $\in$ CStore^{init} $\wedge$ partial plan $\in$ CStore^{resp} :

- The initiator informs the social reasoning module that the dialogue goal was successfully achieved and the social reasoning module informs the planner that the service was successful.
- The respondent informs the social reasoning module that it is now committed to performing the actions encoded in the partial plan. The social reasoning module informs the planner that the partial plan has been accepted and the planner must now generate plans in light of this restriction.

partial plan $\in$ CStore^{init} $\wedge$ partial plan $\notin$ CStore^{resp} :

- The initiator informs the social reasoning module that it has been unsuccessful in attaining acceptance of the partial plan. If another agent is eligible to perform the actions in the partial plan then the social reasoning module selects a new agent to direct the enquiry towards and delivers a new dialogue goal to the dialogue module. If there are no more eligible agents to

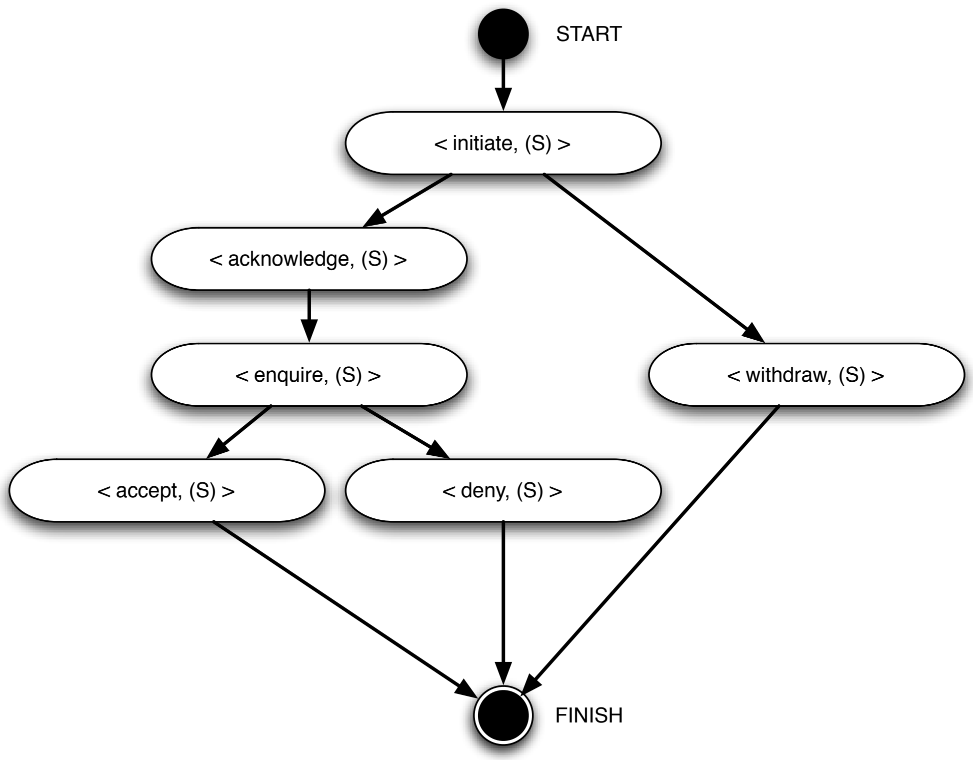

Figure 5.20: A dialectical structure diagram for the schema iXchange0.

communicate with and no agent has accepted the partial plan then the social reasoner informs the planner that no agent could be found to perform the actions.

- The respondent is not committed to the partial plan so informs the social reasoner that this is the case because not aquiescing to the request may have a social impact at some future point in the lifetime of the agent. The planner need not be informed as there is no change in circumstances that will impact the agents ability to generate plans.

The *i*-Xchange project was successful insofar as it proved to be the first, to the authors knowledge, implemented MAS to successfully integrate planning, social reasoning, and argumentation. A second novel aspect of this MAS is that the planner sub-system is fully integrated with the communications sub-system enabling non-acceptance of partial plans by other agents to be fed back to the planner which takes this into account "on-the-fly". A practical success of this project was in demonstrating that MAS can be constructed which are composed of multiple independent subsystems, created using heterogeneous agent platforms, architectures and engineering techniques. The conclusion can be drawn that despite the wide range of competing agent-oriented design and construction paradigms, the increasing number of very different agent platforms, and the disparate underlying formalisms and languages supporting domain specific activities, single, focused and coherent agent systems can still be bult.

5.5 Summary

This chapter has introduced two deployments of the A4A. The first was the encapsulation of the A4A into Colloqui, one component in a fledgeling framework for specifying, building, and deploying arguing agent systems named MASADA. A full arguing agent system that utilises the A4A and a graph colouring based knowledge domain can be built using the Colloqui and Caucus components. The central lesson learned in developing MASADA stems from the number and range of components that are required to construct a full agent. Whilst a lot of effort has been expended in argumentation and MAS research into protocols to guide particular types of interaction, there has been little research into how these protocols fit into a complete agent system. A MASADA agent, for example, requires at least a dialectical game protocol, a transcript and required artifact stores, a knowledge base, an argument construction capability, and a reasoning process to build even a basic arguing agent capable of acting in a dynamic environment. The second deployment of the A4A was in the *i*-Xchange agent architecture to govern inter-agent communication between agents that plan in a multiagent context and engage in social reasoning. This project dealt with a number of related issues to the development of MASADA namely how multiple disparate subsystems can be harnessed and assembled into a single cohesive argument capable agent.

Chapter 6

Dialectical Game Evaluation

MERELY implementing arguing agents that play dialectical games does not answer the questions of which game to use, or how to play the selected game. The adoption of software frameworks such as the A4A and MASADA reduce the effort required to implement new games but eventually a decision has to be made with respect to which game the agents will actually play in a given dialogue. Either the developer or the user, or even the agent itself, must identify a suitable game schema to use whenever a dialogue is entered into. How such game schemas are identified as "suitable" requires that three tasks be completed. The first task is to investigate each game schema in terms of a range of attributes of dialectical games. The second task requires identification of the range of contexts in which argumentative dialogue occurs. The third task requires a correspondence to be found between particular sets of attributes and particular dialogical contexts. Assuming a suitable game is selected for the required task there is still the question of how an agent determines what it should say given the range of legal moves specified by the game at any given point.

This chapter begins by examining empirical and non-empirical metrics and attributes and the ways in which these can be used to identify, group and organise various families of games. Evaluations on the basis of both empirical and non-empirical metrics are then carried out for a number of dialectical games. Finally, because graph-colouring has been used as a knowledge domain for evaluating dialectical games, the converse is explored, the use of dialectical games as a way to find solutions to graph-colouring problems.

For each dialectical game a range of attributes of that game can be identified and measured. Similarly desiderata for dialectical games designed to be deployed in particular dialogical situations can be identified. Whilst it is not expected that every possible attribute of a dialectical game could be identified, such attributes and desiderata as can be identified are useful for a number of reasons;

1. Measurements of particular attributes can be used to categorise disparate games into classes dependent upon the value of those measurements. For example, the group of games that specify a single move per turn includes H, DC, DD, DE, CB, CB(+), CBV, CBZ, DL, DL2, and DL3. However the group of games that allow multiple moves per turn currently includes only PPD_0.

2. If a correspondence is identified between a given class of games and a given class of dialogical context then the issue of selecting a game to use in that context is greatly simplified.

3. If existing games are unacceptable in a given situation but certain attributes are known to be applicable to that situation then those attributes can guide the process of designing new dialectical games.

Attributes of dialectical games must therefore be identified and measurements made. To determine values for these attributes requires two different approaches. This is because certain attributes cannot easily by measured purely from examination of the rules and thus require an alternative approach. For example the communicative overhead associated with a game, in terms of the average number of messages sent per dialogue, cannot easily be determined by inspection of the game because performance is a measurement that is related to a game's rules in use rather than an externally identifiable parameter. Two types of evaluation can thus be identified. *Non-empirical evaluation* is pursued by identifying those attributes of a dialectical game which can be investigated and measured solely through the inspection of the schema that specifies the game. *Empirical evaluation* is concerned with the investigation of those attributes which are best tackled via indirect means such as the examination of the resultant dialogues played out in accordance with the game rules. The two types of evaluations yield two types of result;

1. Results that can be interpreted in isolation , e.g. determination of whether a game is cumulative with respect to certain tokens, and,

2. Results whose interpretation is comparative. For example it can be useful to compare the efficiency of two games in relation to a given metric. Efficiency however is not an isolated measurement of some attribute but a comparison between two such measurements. For example, a measurement

of communicative efficiency requires the communicative performance of at least two games to be determined and then compared.

Measurements of empirical metrics for dialectical games are acquired by getting the agents to play the games in a dynamic environment and measurements for non-empirical metrics are made through inspection of the game rules.

6.1 Non-Empirical Evaluations

Evaluations that are non-empirical rely upon the idea that an examination of the rules and components of a game can inform the investigator about the game, how it might play, and what attributes it might possess. Such an investigation requires that a list of non-empirical metrics is produced and that the attributes of each are game are then identified in light of the metrics. This kind of investigation can be used to make coarse grained evaluations and can be used as a general guide for the development of games that have certain characteristics;

Dialectical Interactions Are the moves of the game open, closed, or partially closed? If every move has a specific set of mandatory responses then the dialectical interactions of the game are closed. If no moves have any mandatory responses then the dialectical interactions are open. If some moves specify mandatory responses then the dialectical interactions are partially closed. This attribute gives an idea of the number of responses that may be legal at any given time during a dialogue. A closed game has a minimal set of responses at any given point whereas in an open game, any move may be played for which the requirements can be met. The moves of a partially closed game can be represented by the ratio of open to closed moves, for example, there are five moves in the original formulation of Mackenzies' DC, of which two are open and three are closed, so DC can be described as a 2:3 partially closed game. The specification of particular dialectical interactions have an effect on the range of profiles of dialogue that can be produced using the game. A closed game is generally more restricted in terms of the profiles which it can be used to produce compared with a more open game on account of the restriction of legal responses.

Number of Moves The greater the number of moves that are available the more flexibility a game has, however a larger number of moves could lead to more verbose dialogues. The number of moves is only a rough guide however and comes with that caveat that the flexibility of a game and the size of resultant dialogues both depend upon the formulation of rules that set out legality for the move. A game may include many moves, but each move could be so tightly constrained that it is only legal in a limited set of circumstances so the large number of moves would not necessarily affect the size of the resultant dialogue. In combination with the status of the game in terms of its dialectical interactions, the number of moves attribute enables a game developer to gain a coarse grained view of the communicative performance of the game. Two other aspects of this attribute concern the number of requirements and the number of effects that are specified for each move. Although any given requirement or effect could be more computationally complex than the others, or possibly computationally intractable in a badly designed game, the number of such requirements and effects gives a coarse grained view of the computational complexity of any given move such that a move with a single requirement is less computationally complex than a move with a dozen requirements. It should be noted that in a framework such as the A4A the majority of requirements are specified in terms of the states of game components which can be verified in, at worst, linear time where such corresponds to the size of the component that has to be inspected.

Number of Players The number of players that can interact in a game has an impact upon the complexity of initiating a dialogue. To initiate a two player dialogue requires merely that the initiating agent send a message to its desired dialogue partner containing a legal initiatory locution. Conversely, initiating a dialogue with more than two players requires co-ordination to ensure that all players can be brought into the dialogue correctly.

Turn Structure The progression of turns in a game can be either strict or liberal. A strict system follows the traditional idea of each player having their turn which they are allowed to take in a specific order such that all players get approximately the same number of turns during a game, e.g. at termination the number of turns played by each player should be equal or within one turn of all the other players. In a liberal turn progression system, the player who takes their turn next is dependent upon the moves that have been played in the current turn. This kind of progression is only really applicable to games with three or more participants as in a two player game the only other player who can take their turn next is the second player. A second issue with regards to turn structure concerns the magnitude of turns. A game may allow only a single move to be played by each player per turn, or it may allow multiple moves per turn. In other games the number of moves allowed per turn may be dependent upon the other moves that are played in the turn such that the minimum number of moves is one.

Symmetry A symmetrical game ostensibly allows all players to play the same moves although the actual moves allowed at any given point is dependent upon history of the dialogue and satisfaction of each moves requirements. A non-symmetrical game assigns roles to different players where each role has different moves associated with it. By examining whether a game is symmetrical, in combination with the number of moves, and the requirements and effects of each move, a coarse determination of the computational load of the game can be made with respect to each player.

Cumulativeness A game may be cumulative or non-cumulative with respect to the tokens that it manipulates. These tokens may equate either to the sentences expressed as the content of each move or to the locutions themselves. Using cumulativeness as a way to group and differentiate games was explored by Wells in [139] in which the idea that certain application domains are more suited to cumulative games than non-cumulative games was explored. A game that is cumulative will, if it also includes rules to prohibit repetition of those accumulated moves, eventually stall when the players are no longer able to make an utterance because of the rules. This could be a useful way to limit the length of dialogues to some multiple of the size of the knowledge base. Although if appropriate strategies are used dialogues should not become excessively large it is comforting for a game developer if they can stipulate that dialogues will conclude at some finite point should the strategy fail.

Termination Rules Termination rules give specific states of the game components under which a dialogue will come to a conclusion. Termination rules may also include specification of how the dialogue terminates with respect to the participants, for example specifying a win-lose-draw assignment. Termination rules are important because they give an objective way to assess the success or failure of a dialogue with respect to its participants. For example a success rule may specify that a dialogue is successful with respect to the initiating party if the initial thesis, meaning the content asserted at the beginning of the dialogue, is accepted by the respondent.

Initiation Rules Initiation rules identify the requirements for a dialogue to be legally commenced. These might specify that only certain moves are legal during the initiation phase of the dialogue, or may be used to ensure that both players establish their position with respect to some initial thesis before the dialogue proper begins.

Progression Rules These enable a given schema to specify either a shift or embedding that can occur within a dialogue played according to the current schema. Such a progression could affect the potential computational or communicative load associated with any given game because it is no longer dependent just upon a single dialectical game but also all of the other games that could result from progressions, which could each possibly be played according to different sets of rules. Progressions rules therefore greatly complicate the issue of evaluating a given dialectical game.

The summary of an evaluation of a number of extant games using the static attributes can be seen in table 6.1. Whilst this table doesn't explain how each game performs in practise, it does enable a high-level comparison to be made of disparate games. For example, when selecting an existing game for implementation in a MAS, there may be a requirement to use a particular turn structure. If a game was to be implemented which allowed multiple locutions per move then only the PPD_0 and PPD_1 games are eligible. Similarly if the implementation called for the use only of games which include termination rules then the pool of existing rule sets is limited to the CB family of games, the PPD family of games, and TDG. Such a table is therefore a useful tool to allow existing games to be broadly characterised and grouped. To make fine distinctions between different rules on the basis of how they perform however requires a different tool which is the focus of the next section.

6.2 Empirical Evaluation

Contrary to the approach taken in the non-empirical evaluations, empirical evaluations are predicated upon the idea that the artifacts of a game, such as the transcript of player utterances, can be examined to learn about how that game performs.

Efficiency & Performance

Two types of efficiency are identified as particularly pertinent to dialectical games as used in MAS, these are communicative efficiency and computational efficiency. Communicative efficiency may be investigated in terms of a comparison between different games so that it can be said that under particular circumstances, one game is more efficient that the other, and it may also be investigated in terms of a comparison between the dialectical game approach to inter-agent communication versus other approaches.

Determining a measurement for the communicative efficiency of a system requires that the performance of that system be determined under a known set of circumstances. Such a performance measure-

Game	**Interactions**	**Moves**	**Players**	**Turns**	**Symmetrical**	**Cumulativity**	**Term**	**Init**	**Prog**
H	Partial 2:3	5	2	strict : single	yes	no	no	no	no
DC	Partial 2:3	5	2	strict : single	yes	locution:challenge	no	yes	no
DD	Partial 2:3	5	2	strict : single	yes	no	no	yes	no
DE	Partial 2:3	5	2	strict : single	yes	no	no	no	no
CB	Partial 2:2	4	2	strict : single	yes	Immediate consequence	yes	no	no
CB+	Partial 2:2	4	2	strict : single	yes	Immediate consequence	yes	no	no
CBV	Partial 2:2	4	2	strict : single	yes	Immediate consequence	yes	no	no
CBZ	Partial 3:2	5	2	strict : single	yes	Immediate consequence	yes	no	no
DL	Partial 4:4	8	2	strict : single	yes	no	no	no	no
DL2	Partial 5:4	9	2	strict : single	yes	no	no	no	no
DL3	Partial 4:5	9	2	strict : single	yes	no	no	no	no
PPD_0	Partial 4:5	9	2	strict : multiple	yes	no	yes	yes	no
PPD_1	Partial 4:5	9	2	strict : multiple	yes	no	yes	yes	yes
TDG	Closed	15	3	liberal : single	no	no	yes	no	no

Table 6.1: Non-Empirical Evaluation of Dialectical Games

ment may be in terms of a number of metrics including;

1. number of messages per dialogue,
2. size of individual messages,
3. number of individual locutions per message, and
4. number of individual locutions per dialogue.

Once such performance measurements have been made for multiple games then they can be compared and a performance ranking over the games determined. Notice that there is some antagonism between the performance metrics insofar as a game that allows many locutions per turn can, although not necessarily, yield dialogues of a shorter overall length in terms of the number of messages per dialogue because more information can be exchanged between the players in less moves. However this is at the expense of having to send larger messages in each turn. When developing a MAS this kind of tradeoff must be considered in light of the networks over which the MAS will be distributed. If those networks are public, so that the developers have little control over them, or if there are other constraints which limit the available bandwidth for communication between agents, then the ability to select a more communicatively efficient game is valuable.

Determination of a measure of the communicative efficiency of a game is an important metric to investigate from the perspective of real-world multiagent deployment because of the impact that a communication heavy protocol could have on the MAS in which it is deployed.

The computational performance of a game is important because it impacts on whether the game is suited to a particular application domain. A game that exhibits poor performance in a given situation is less computationally efficient than a game with better performance and should therefore be pursued when games are being evaluated with the aim of deployment in a MAS. A dialectical game can affect an agents computational performance if the number of states that must be checked to verify legality is large or if evaluating those states is computationally complex or heavy. The metric associated with the computational performance attribute that can be measured is the CPU load during message building tasks.

Measuring the computational performance of a given agent in a MAS is more difficult than measuring the communicative performance especially when the multi-threaded nature of most MAS frameworks is taken into account because it is difficult to coordinate the computational load at any given time against any individual agent in the MAS. As a result it can be difficult to isolate any single agent to determine which part of the agents cycle is CPU intensive. Two approaches can be identified, the first approach involves evaluation of the MAS as a whole such that for a particular period of time the average CPU load is determined and this value compared against the same set-up using another game. Unfortunately this approach does not differentiate between individual agents or specific activities that the agents are performing at any given point. For example arguing agents spend time reasoning, updating the knowledge base, and investigating their environment, in addition to engaging in dialogical activities. However, similar to the evaluations of communicative performance, if the same initial set-up is reused for two game and only the games themselves differ then any differences in output could be ascribed to the difference in game. The other approach is to isolate a single agent onto a different machine to the other agents in the MAS and to make the required measurements on the isolated agent. This approach enables computational load associated with specific activities to be measured and evaluated. This kind of approach is specifically supported by the Caucus MAS approach, firstly because it uses the JUDE platform which makes it simple to run agents within the same MAS but on physically separate machines, and secondly because the Colloqui component runs in its own thread so that it is only performing a single activity at any given time.

By carrying out a range of investigations, and making measurements for a number of different metrics in a number contexts of interaction, a comprehensive picture of how each game performs can be built.

6.2.1 Application Of MASADA Components To Game Evaluation

The Colloqui and Caucus components of MASADA have been developed to support the empirical investigation and evaluation of dialectical games in the context of MAS. These investigations concern three important issues;

1. What capabilities are required of an agent for it to be able to engage in purposeful, dialectical game regulated, communicative interactions with other similarly skilled agents?
2. How do these capabilities translate into software?

3. Given a society of agents who are capable of such communication, how do the resultant dialogues compare in terms of their empirical metrics?

The first two issues, concerning capabilities and software, have been investigated to some extent in the development of the A4A, Colloqui, and Caucus components. Although they appear similar, there are real questions to be answered concerning the boundaries between components, and which agent capabilities should be subsumed into any single component. If a single component is responsible for too many capabilities then it is no longer a straightforward matter to replace the original component with a new component to provide enhanced capabilities. There is also an incremental aspect to providing capabilities to an agent. For example, once the A4A had been constructed to support representing and playing dialectical games, it was necessary to construct software to record knowledge about a given domain and enable that knowledge to be expressed as content within a dialectical game move. This yielded the part of the Colloqui component concerned with knowledge representation which enables an agent to express statements that encode a representation of the agents environment in such a form that the statements can be exchanged with other agents in the MAS. Given the A4A component and the knowledge representation portion of the Colloqui component, an agent can instantiate fully formed utterances that can be legally expressed as a part of the current dialectical game. However neither the A4A nor the Knowledge Representation portions of Colloqui tell the agent which move to select at any given point in the dialogue. This is the domain of the Colloqui reasoning process which tells the agent how it should engage in dialogue. Given the types of locution available, specified by the A4A, and the range of statements that could be used to as content for each locution, which locutions should be uttered?

For example, at a given point in a dialogue, an agent might have two legal locution archetypes available, <*Retract,* {*p*}> or <*Support,* {*p, q*}>. The agent may thus retract its commitment to p or it can alternatively utter q in support of p. However this is as far as a dialectical game goes in formulating what an agent can legally say. The dialectical game doesn't suggest which of the alternatives should be taken, neither does it specify what q should be uttered in support of p.

This kind of move selection requires some level of strategic reasoning, whether lower level strategy which specifies the heuristics necessary to exhibit general trends of behaviour, or higher level strategy where end-to-end reasoning is required to enable construction of a set of moves, utterance of which lead the agent through the dialogue from initial goal to satisfied goal. The relationship between these levels of responsibility is illustrated in figure 6.1 which shows how the set of permitted moves, which are either mandatory or merely legal and are defined by the game schema, is a superset of the relevant moves as selected by the reasoning process. In turn the set of good moves are a subset of the set of relevant moves and the optimal (if any) moves is a subset of that.

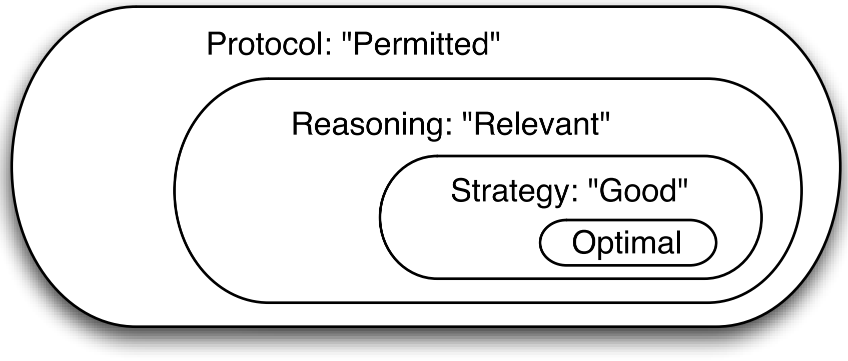

Figure 6.1: The hierarchy of encapsulation between legal utterance and optimal utterance.

By identifying the capabilities required to construct a fully formed agent capable of arguing with other similar agents, and by implementing each capability within a distinct software component, robust arguing agents can be constructed, built from separate modular components with well-defined interfaces. This process compliments a wider goal within multiagent research, that of conceptualising, designing and building agents that argue. This is still a major challenge and implemented systems which integrate such disparate elements as planning, reasoning, and argumentation are scarce. One implemented system is the *i*-Xchange MAS discussed in section 5.4 which integrates social-reasoning, multiagent planning and argumentation. By engaging in the process of designing and creating arguing agent systems, the requirements of such agents can be established and investigated, robust components to support each capability can be constructed, and new tools to support the construction and analysis of arguing agent systems can be developed.

The third issue that MASADA is applied to is the issue of dialogical metrics. The primary metric which MASADA has been used to investigate is that of communicative performance. Because dialectical games govern the way in which agent dialogues evolve, the performance of a game can directly affect the communications overhead of a MAS. On a coarse scale, if a protocol requires many large, verbose messages to be exchanged then that protocol performs less well than a protocol in which a smaller number of more succinct messages are exchanged. So two communicative performance metrics can be identified, the average number of messages exchanged per dialogue, and the average size of each message. A further game attribute that is intimately concerned with the notion of communicative performance is that of termination. For the average number of messages to be determined requires that the dialogues actually terminate within some finite number of moves. If a dialogue cannot be guaranteed to terminate then that game is likely unsuitable for many real-world applications where efficient use of available computational and communicative resources is very important. The ability to calculate such metrics is important, not only because it enables a practical means to differentiate between the many otherwise similar protocols, but also because it suggest a path towards arguing agent societies in which the dialectical games are fine-tuned to their application domain. Measurement of metrics such as communicative performance are also a useful way for system developers to decide which game to adopt when new agent systems are being developed.

A MAS constructed using the Colloqui and Caucus components can be applied to the problem of determining measurements for those empirical metrics associated with dialectical games by generating a number of dialogues which subsequently form the basis for a comparison between the games. The core of this approach relies upon the notion that if a MAS is set up in which the initial parameters are defined and isolated, and the only changes that are made alter the schema controlling the production of dialogue, then any differences in those resultant dialogues are due to the changes in the schema. The remainder of this section details the set-up of such a Caucus MAS for dialectical game evaluation, the set-up of a particular scenario for evaluating the communicative performance of a number of games, the results of such an evaluation, and some lessons learned and conclusions drawn from implementing such a MAS.

To run an instance of a Caucus MAS with the aim of producing empirical data to support comparative dialectical game evaluation requires that a range of startup parameters for each component of the MAS are defined. These parameters are used to initialise the agents as they are created so that they are ready to take part in dialogue once they join the MAS. This is achieved by specifying parameters through each components preference file and by supplying required supplementary files such as a scenario specification and the required dialectical game schemas.

For the Caucus component this requires specifying the number of agents to load, through the JUDE preference file, and the destination for agent log files, through the Caucus preference file, *caucus.xml*. The Colloqui preference file, *colloqui.xml*, is used to specify the location of a scenario specification file to load and to nominate the A4A schema that is to be used during the evaluations. The A4A component requires that the destination for dialogue transcripts be specified, this is done through the A4A preference file, *a4a.xml*.

An overview of the input and output files for the Caucus MAS used in evaluation is shown in figure 6.2.

For the purposes of evaluation the following JUDE preference file "*20.xml*" is used;

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE agentloading SYSTEM "agentconf.dtd">
<agentloading>
   <agents defaultgroup="colloqui">
      <agent name="agent" number="20">
         <module jar="file:///colloqui/dist/jars/CaucusCoreAgent.jar"/>
      </agent>
      <agent name="env" number="1">
         <module jar="file:///colloqui/dist/jars/CaucusEnvAgent.jar"/>
      </agent>
      <agent name="gui" number="1">
         <module jar="file:///colloqui/dist/jars/CaucusGuiAgent.jar"/>
      </agent>
   </agents>
</agentloading>
```

In this preference file a lookup group named "Colloqui" is specified which the agents use to find each other. Three different groups of agents are then created to account for the core agents, environment agents, and gui agents and the JUDE module to load for each agent is specified. The core agent group contains twenty agents each of whose name is prepended by "agent" and each of which load the CaucusCoreAgent module. The env group contains a single agent whose name is prepended by "env" and which loads the CaucusEnvAgent module. Similarly the gui group contains a single agent whose name is prepended by "gui" and which loads the CaucusGuiAgent module. This enables three types of agents to be loaded, each of which has different capabilities, roles, and responsibilities within the MAS.

During the load process, as each agent is initialised according to the parameters specified in the JUDE preference file, core agents parse the following Colloqui preference file to determine which scenario to load;

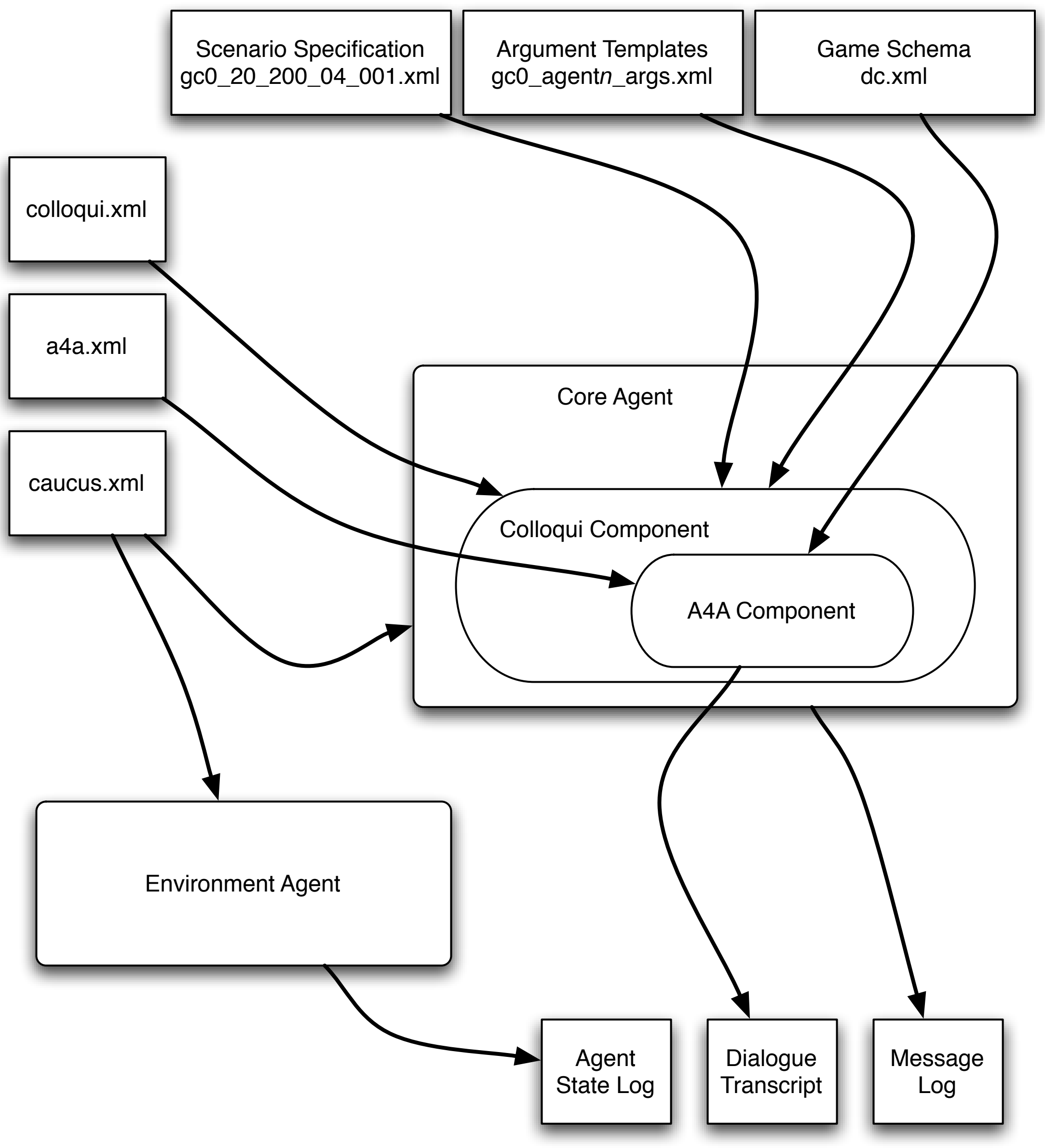

Figure 6.2: Colloqui I/O Overview: This Caucus MAS contains an environment agent and a number of core agents. Six separate XML files are used to initialise the various components that form the agents. These including a preference file for each of the A4A, Caucus, & Colloqui components, a scenario specification file, an argument template for each agent and a game schema. Three types of output file exist, a log of agent state updates received by the environment agent, a log of messages received by each agent, and a transcript of each dialogue engaged in by the agents.

```
<?xml version="1.0"?>
<colloqui>
   <domain name="gc" instance="0">
      <scenario name="20:200:04">
         <initfile>
            <name>gc0_20_200_04_001.xml</name>
            <path>/Users/simon/code/projects/caucus/config/scenario/gc0/</path>
         </initfile>
      </scenario>
   </domain>
</colloqui>
```

This specifies that the agents should load instance '0' of the gc knowledge domain and the particular scenario within the GC_0 domain is named GC0_20_200_04. Once an agent has parsed the Colloqui preference file, it parses the nominated scenario specification file to determine its initial knowledge and state. The scenario used in the evaluations is named GC0_20_200_04 and specifies 200 relationships between the twenty core agents such that each individual agent has a relationship with ten other agents. All agents are initialised to state '0' so that conflicts exist in the agent society immediately following startup. Because the actual scenario file used is both long and repetitive, a representative fragment is presented here in which the first and last agent specifications are included and the intervening eighteen are replaced by "...";

```
<?xml version="1.0"?>
<knowledge>
   <agent>
      <name>agent0</name>
      <colour>0</colour>
      <neighbours>
         <neighbour_of>agent1</neighbour_of>
         <neighbour_of>agent3</neighbour_of>
         <neighbour_of>agent5</neighbour_of>
         <neighbour_of>agent7</neighbour_of>
         <neighbour_of>agent9</neighbour_of>
         <neighbour_of>agent11</neighbour_of>
         <neighbour_of>agent13</neighbour_of>
         <neighbour_of>agent15</neighbour_of>
         <neighbour_of>agent17</neighbour_of>
         <neighbour_of>agent19</neighbour_of>
      </neighbours>
      <argument_templates>
         <path>/Users/simon/code/projects/colloqui/config/args/templates/</path>
         <template_file>gc0_agent_args.xml</template_file>
      </argument_templates>
   </agent>

...

   <agent>
```

```
      <name>agent19</name>
      <colour>0</colour>
      <neighbours>
         <neighbour_of>agent0</neighbour_of>
         <neighbour_of>agent2</neighbour_of>
         <neighbour_of>agent4</neighbour_of>
         <neighbour_of>agent6</neighbour_of>
         <neighbour_of>agent8</neighbour_of>
         <neighbour_of>agent10</neighbour_of>
         <neighbour_of>agent12</neighbour_of>
         <neighbour_of>agent14</neighbour_of>
         <neighbour_of>agent16</neighbour_of>
         <neighbour_of>agent18</neighbour_of>
      </neighbours>
      <argument_templates>
         <path>/Users/simon/code/projects/colloqui/config/args/templates/</path>
         <template_file>gc0_agent_args.xml</template_file>
      </argument_templates>
   </agent>
   <states num="4" />
</knowledge>
```

In this case each agent is initialised with the knowledge that it has relationship with ten other agents, that it's own initial state is '0' and that the number of available states is '4'. Agents only parse the part of the scenario that refers to them individually so although each agent can 'see' the state of its neighbours it does not initially know about the relationships between the other agents. The <argument_template> element is used to specify the location for each agents argument template file. In this case, because all agents have access to the same range of arguments the same argument template file is supplied to each agent.

Argument templates are used to relate statements into stereotypical arguments associated with the knowledge domain. Argument templates are specified for each agent within an argument template XML file as detailed in section 5.2. This allows the available arguments to differ between agents although all agents use the same set of templates in the following evaluations. In the graph colouring domain, dialogues are instantiated with the aim of achieving dialogue goals. The only implemented dialogue goal is the resolution of conflicts. Conflicts can only be resolved through an agent changing state, and agents will only change state if they are committed to so doing at the end of a dialogue. If an agent is credulous with respect to its practical reasoning then it will accept a request for a state change, however if an agent is sceptical it will require justification, in the form of a supporting argument, for why it should change state. To support this a a number of legal arguments are specified for use in the evaluations to enable an agent to justify its position with regards to a state change request. In the evaluations the only action capability available to core agents is to change their own state. This is expressed by agents in the form of the *change_state(agent(x))* predicate, which states that agent$_x$ should change state. All arguments provide either direct support for the change_state statement or support its premises in some way.

The following template can be used to provide a basic argument for why a change of state should be effected;

premise$_1$: $conflict(agent(X), agent(Y))$
conclusion: $change\_state(agent(X))$

In turn an agent can support the conclusion that it is in conflict with another agent using the following template which provides an argumentative basis for the notion of a conflict built from the recognition of that two neighbouring agents are in the same state;

premise$_1$: $colour\_state(agent(X), state(Z))$
premise$_2$: $colour\_state(agent(Y), state(Z))$
premise$_3$: $neighbours(agent(X), agent(Y))$
conclusion: $conflict(agent(X), agent(Y))$

This argument is quite weak and the notion that an agent should change its state merely because it is in conflict would only persuade a particularly altruistic agent. A slightly stronger argument can be constructed based upon the idea that there are a limited number of colour states. If an agent has no available colour state that it can change to but the neighbour whom it is in conflict with does have available colour states then the agent with the available colour states should change colour. The rationale for this is that an agent without colour states will only swap one conflict for another so the number of conflicts that exist locally will not change, however if the agent who does have available colour states elects to change colour then that will reduce the local number of conflicts by at least one, and possibly more if the agent is involved in multiple conflicts involving the same colour state.

premise$_1$: $available\_states(agent(X))$
premise$_2$: $no\_available\_states(agent(Y))$
premise$_3$: $conflict(agent(X), agent(Y))$
conclusion: $change\_state(agent(X))$

The last argument template introduced the notion of arguments based upon the number of conflicts in a given locale reducing the number of available colour states for the agents concerned. The next template builds upon this notion but takes a different approach using the idea that the agent which has more conflicts should change state. This argument can be instantiated in two forms and the simpler form is presented first;

premise$_1$**:** $more\_conflicts(agent(X), agent(Y))$
premise$_2$**:** $conflict(agent(X), agent(Y))$
conclusion: $change\_state(agent(X))$

Upon a challenge of the premise of the "more conflicts" argument template can be supported, in a similar manner to the defense of the *Conflict* → *Change_State* argument, with the following template;

premise$_1$**:** $num\_conflicts(agent(X), i)$
premise$_2$**:** $num\_conflicts(agent(Y), j)$
premise$_3$**:** $greater(i, j)$
conclusion: $more\_conflicts(agent(X), agent(Y))$

The more complex form of the "more conflicts" argument template involves a count of the number of conflicts in a specific area, called the locale, around the conflicting agents. The determination of the locale of the agent is made by counting the number of conflicts within a particular radius of each of the conflicting agents. This argument is based upon the notion of stability and that a more stable locale has less conflicts than a less stable locale, hence a more stable locale is more desirable and the agent in the less stable locale should change colour state in order to remove a conlflict from its locale and thus improve its stability.

premise$_1$**:** $less\_stable(agent(X), agent(Y), locale(Z))$
conclusion: $change\_state(agent(X))$

The stability argument can be supported if needs be with the following template;

premise$_1$: $stability(agent(X), locale(Z), i)$
premise$_2$: $stability(agent(Y), locale(Z), j)$
premise$_3$: $greater(i, j)$
conclusion: $less\_stable(agent(X), agent(Y), locale(Z))$

Because there are three separate argument templates that each provide independent support for the change_state conclusion, a preference ordering is specified amongst them as follows;

$$conflict << available\_states << more\_conflicts << less\_stable$$

This ordering means that, for example, a complete and valid reification of the available states argument is a stronger argument than a complete and valid reification of the conflict argument.

Analyses of the argument templates can be found in figures 6.3 and 6.4 which present the arguments in the diagramming style used by Araucaria [104].

6.2.2 An Investigation of Communicative Performance

An investigation of the relative performance of a number of dialectical games was performed using a Caucus MAS and the Colloqui component. The MAS was initialised according to three different scenarios which are characterised by the ratio of number of agents to number of relationships. The three scenarios used were GC0_20_150_04, GC0_20_200_04, and GC0_20_250_04 which have vertice to edge ratios of 20:75, 20:100, and 20:125 respectively. Each scenario incorporates twenty agents with seventy five, one hundred, and one hundred and twenty five relationships between the agents respectively. The agents used were specified with the following characteristics. Initiating agents are laconic and will immediately withdraw from the dialogue in one of two cases, once either its opponent has incurred commitment sufficient to satisfy the initiating agents dialogue goal or if the initiating agent has uttered all available arguments in support of its position and yet the respondent has still not accepted the initiating agents goal. Responding agents are verbose and will respond to the initiating agents messages as long as they have a response. If an agent cannot respond then it withdraws from the dialogue. After the initial, goal-related utterance with content, <change_state, agent(X)>, both agents select their utterances by random from the pool of available utterances given the legal or mandatory locutions, and the statements that can be expressed given the agents current knowledge. Once initialised the MAS was allowed to run

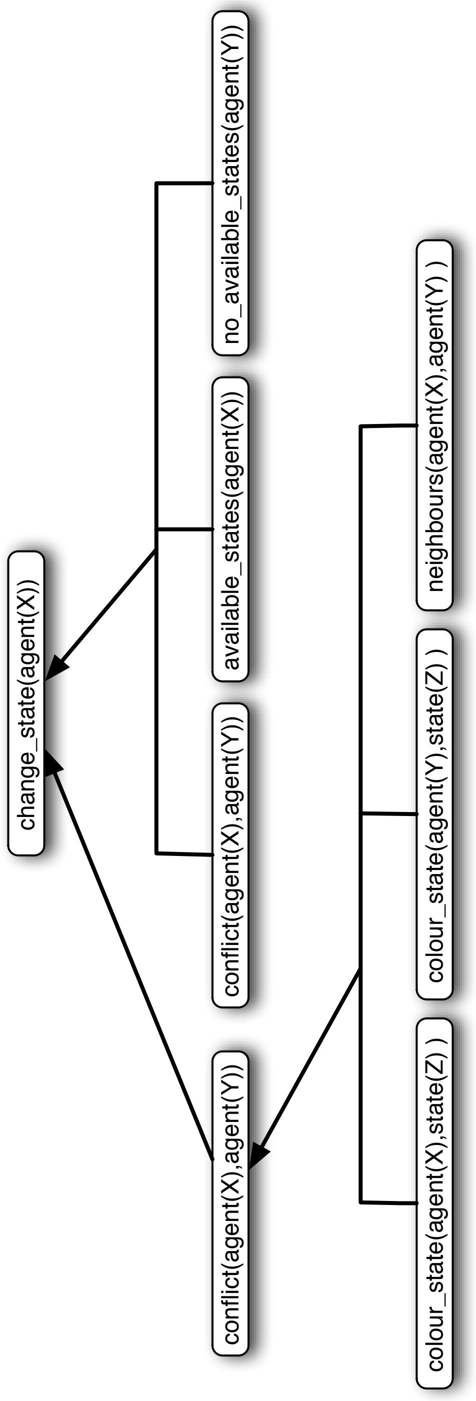
change_state(agent(X))
no_available_states(agent(Y))
available_states(agent(X))
conflict(agent(X),agent(Y))
conflict(agent(X),agent(Y))
neighbours(agent(X),agent(Y))
colour_state(agent(Y),state(Z))
colour_state(agent(X),state(Z))

Figure 6.3: Argument diagram for the “conflict” and the “available states” arguments.

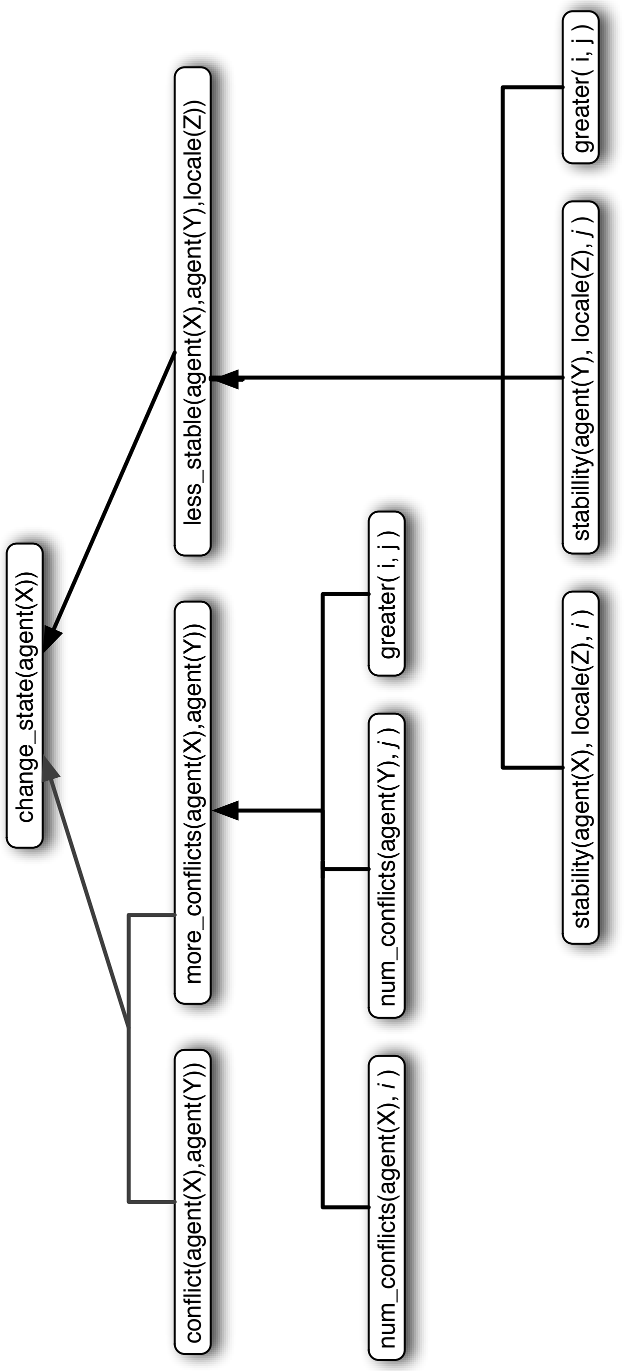
change_state(agent(X))
conflict(agent(X),agent(Y))
more_conflicts(agent(X),agent(Y))
less_stable(agent(X),agent(Y),locale(Z))
num_conflicts(agent(X), *i*)
num_conflicts(agent(Y), *j*)
greater(i, j)
stability(agent(X), locale(Z), *i*)
stability(agent(Y), locale(Z), *j*)
greater(i, j)

Figure 6.4: Argument diagram for the “more conflicts” and the “less stable” arguments.

until one thousand dialogues had completed. This was repeated ten times for each of the games CB, DC, DL3, H, and PPD $_0$ to yield raw data with which to analyse the communicative performance of each game. The whole process was then repeated for each of the three scenarios.

All games are played according to their original rules however only CB and PPD_0 specify termination rules. As a result dialogues played using H, DC and DL3 continue indefinitely. This is the worst case scenario in terms of communicative performance in a MAS context because if a dialogue has no way to be terminated then it may continue indefinitely and retain valuable system resources that could be otherwise deployed. Where agent communication is concerned, it is actually less important a question of whether a dialogue will complete eventually but whether dialogues will complete within a finite time, measured in terms of the number of turns that have occurred during the dialogue. If the average length of a dialogue exceeds a given measure then, dependent upon the domain in which the agents act, the dialogues can be deemed too long to be of practical use, and hence the dialectical game which gave rise to them can be deemed unsuitable for that application. This means that H, DC and DL3 cannot be evaluated without some means to identify a termination state and to inform all concerned agents that such is the case.

To avoid this situation, the approach taken in the evaluations is to use a meta-game which runs concurrently to each instance of any dialectical game that does not specify termination conditions. The meta-game contains only the <withdraw, (–)> locution which causes the current dialogue to terminate. Because neither H, nor DC, nor DL3 specify rules that require a player to continue in a dialogue, it is assumed that using the withdraw locution is not an infraction of the rules of any of those games, and that the addition of such a locution is actually in fact necessary to make the games tractable in the context of agent communication. The contents of the player's respective commitment stores at the termination point are then used to update the agent's knowledge base, in light of new or updated information that the agents have learnt during the dialogue, and to determine whether a change state action must be performed as a result of commitments incurred.

Once the issue of dialogue termination has been tackled, the notion of communicative performance is less clear cut. This is because the issue of communicative performance becomes entangled, to some degree, with the capabilities of the agents who are engaged in the dialogue. Although a baseline measure of efficiency can be made, this measurement can be greatly improved by supplying enhanced reasoning and strategic capabilities to the agents concerned.

Game	**20:75**	**20:100**	**20:125**
CB	233	247	257
DC	143	147	162
DL3	222	240	266
H	147	153	159
PPD$_0$	468	484	518

Table 6.2: Results of an investigation of the communicative performance of a range of dialectical games. For each of the games CB, DC, DL3, H, and PPD$_0$ three evaluations of the communicative performance were carried out under three different scenarios. A MAS was allowed to run until one thousand dialogues had completed. The number of messages per dialogue was recorded. Ten iterations were performed and the average number of messages per dialogue per iteration was calculated. This process was repeated for three different Caucus scenarios which are indicated by the ratios in each column heading.

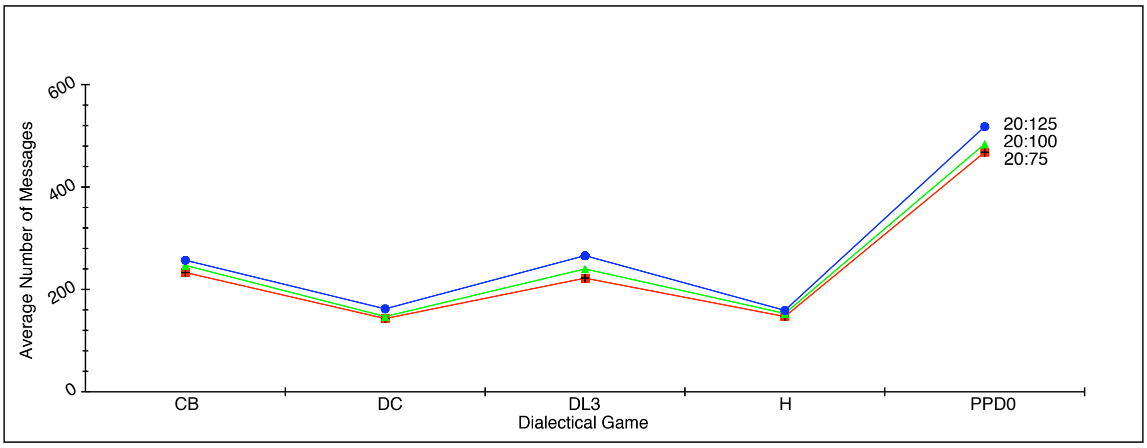

Figure 6.5: Graph showing the average number of messages per scenario for each dialectical game

The results of the communicative performance evaluations can be seen in table 6.2. One aspect that is immediately noticeable is that the average dialogue size is very large with the shortest dialogues occurring in the 20:75 scenario of DC with an average dialogue length of 143. All other iterations, in every game investigated and for every scenario, yielded dialogues with a longer average length. This is a direct result of a combination of two factors, firstly the use of random utterance selection as opposed to the use of an efficient strategy for selecting relevant utterances, and secondly the size of the agents knowledge base which is related to the number of agents, the number of states, the number of relationships, the num-

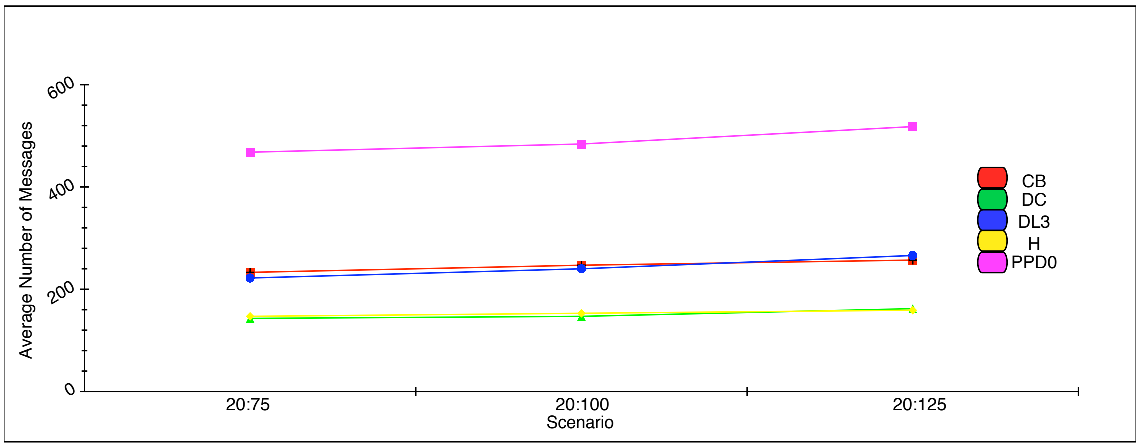

Figure 6.6: Graph showing the average number of messages per dialectical game for each scenario

ber of expressions that can be stated about the agents knowledge, and the number of arguments that can be constructed using those expressions. It is assumed that the use of an efficient strategy would greatly reduce dialogue length although large dialogues could still result if particularly sceptical agents have access to a large knowledge base. Nevertheless, the results do show a difference in the way that each game performs. The games can be ranked in the following order in terms of their relative communicative performance, from most efficient to least efficient DC $>$ H $>$ CB $>$ DL3 $>$ PPD_0.

Because the games have been evaluated using the same initial set-ups, averaged over a number of iterations, it is assumed that any differences in the results stem from differences in the underlying dialectical games. The structure of each game, how it is specified in terms of its constituent components, affects how the game performs. The underlying reasons therefore for any measured differences should be identifiable by examination of the games specification in light of the evaluative results.

Broadly, the better performing games, such as DC and H, have a smaller range of locutions whilst the games that performed worse, such as DL3 and PPD_0, have a larger range of locutions. Noticeably the game with the smallest range of locutions, CB, did not perform best, and actually performed approximately as well as DL3 despite having twice as many locutions as CB. Also, whilst PPD_0 and DL3 have a similar number of locutions they actually perform very differently. Clearly it is not therefore solely the number of locutions that is a factor in the performance of a given game. The best performing games both include locutions that cause a player to be able to add commitments into their opponents commitment store. Whereas a player must usually attempt to cause their opponent to act to incur commitment such locutions enable a player to be able to reverse this process, requiring an agent to be active to discharge any unwanted commitments incurred due to their opponents utterances.

It would appear from this investigation that games applicable to MAS contexts would consist of a smaller range of locutions. Such games would include a number of locutions that incur commitment in their opponents commitment stores thus causing them to need to be active to discharge unwanted commitments. Finally, games developed for application in agent contexts require termination conditions to enable the participating agents to withdraw from the dialogue when required.

6.3 Solving Graph-Colouring Problems

Graph-colouring was selected as a knowledge domain in which to situate arguing agents because of the inherent scope for conflict and thus conflict resolution via dialogue. The scenarios used in the evaluations do not have four-colourings so no solutions are found during those evaluations. The rationale for selecting these scenarios was so that there would always be at least one conflict in the MAS. Whenever that conflict is resolved it therefore leads to new conflicts and for each conflict a dialogue occurs. This ensures that the setup will provide sufficient dialogues for evaluation. Because conflicts provide motivation for the agents to engage in dialogue this was necessary as the generation of dialogues, and hence empirical data, was of prime concern from the point of view of investigating the properties of dialectical games. However this does not mean that finding solutions to graph-colouring problems is not of interest. To the contrary there are two aspects of graph colouring that are both pertinent and interesting here. The first aspect concerns the search for new techniques to apply to distributed graph-colouring, and the second aspect concerns the idea that there are certain classes of problems that can be tackled using argumentation based techniques.

In the game evaluations, the concern was primarily the investigation of the properties of dialectical games. In this context graph-colouring provided a *knowledge domain* and motivation for agent action and interaction. The focus was upon the dialectical games and not upon the domain that facilitated playing the games. However, the domain itself is of interest especially in the context of application, rather than evaluation, of dialectical games, i.e. distributed graph colouring as a *problem domain* rather than merely as a facilitating knowledge domain. Distributed graph-colouring is of increasing importance because results and techniques found therein can be applied to a range of practical issues. For example, determining the initial structure for an *ad hoc* multi-hop wireless radio network, or assigning communication channels to autonomous nodes in distributed wireless LAN networks (known colloquially as Mesh networks). In *ad hoc* multi-hop radio networks the nodes may act autonomously communicating via radio. If any two nodes are not within communication distance of each other then they may communicate via a third intermediate node, if such a node exists. The organisation of communication between the nodes is organised by the nodes themselves but the lack of an *a priori* infrastructure has an effect upon the construction and deployment of the network. Without a top-down structure, the nodes must self-organise initially to determine which communication pathways are valid between any pair of nodes. Furthermore, if the network is required to be robust and efficient, then the network must determine the most efficient communication pathways amongst the merely valid pathways. Additionally, as the network ages, some nodes may become non-operational or function at reduced power, and new nodes may be added to replace or supplement existing nodes requiring the existing recognised communication pathways to be re-evaluated. An example of such an *ad hoc* radio network can be found in a network of Bluetooth computer peripherals, for which even with a small number of devices, initialisation can be very slow [87].

These issues are commonly investigated in the context of graph colouring, for example, conflict in a radio network may occur when allocating time slots in time-division multiple access (TDMA) based networks, thus if colours are associated with time slots then a colouring can be equated with a conflict-free delegation of time slots [87]. In Mesh networks the most pertinent issue is with regards to aggregation of communication channels. A single node can increase its available bandwidth by aggregating many channels together but this reduces the available channels to other nodes and can affect the connectivity of the network. Channel assignment algorithms are used to get the best balance between overall connectivity for all nodes in the network and aggregated bandwidth for individual pairs of nodes. Such algorithms must avoid collisions in channel assignment so that no more than one pairs of nodes is assigned the same channel at the same time. There is a natural corollary between the resolution of conflicts in a graph-colouring domain situated MAS and, for example, the resolution of channel clashes between wireless nodes or the initial structuring of a multi-hop radio network. The traditional approach to solving such problems assumes extensive, if not perfect, knowledge of the system and centralised control sufficient to use a traditional graph-colouring algorithm. For real world problems, where there is often not perfect knowledge of a system, or the system is too dynamic to allow an end-to-end solution to be found rapidly enough, other approaches must be explored. The application of argumentation in a MAS context enables a distributed approach to be taken in which local solutions are found. An advantage of this is that the state of the system can be explained to interested stakeholders, after the fact, through recourse to the arguments that lead to the situation.

An investigation into distributed graph-colouring as a problem domain for dialectical games is a part of a wider endeavour, to determine whether there are particular groups of dialectical games that are applicable to particular groups of problems. Supplemental to such an endeavour is the need to identify a sufficient knowledge base to enable the agents to resolve their conflicts. For example, a dialectical game alone is insufficient to resolve conflicts. This is because an agent also requires knowledge about the domain. This knowledge can then be expressed as the content of locutions in a dialogue whose goal is the resolution of the conflict. This presupposes that the agent is also capable of selecting the correct locutions to utter at each turn in the dialogue. Although, as was demonstrated in the communicative performance evaluations, agents can resolve a conflict through dialogue by randomly selecting statements to instantiate from their knowledge bases, the resultant dialogues were far from optimal. It is necessary therefore to identify strategies that agents can pursue to resolve conflicts, and the tactics required to facilitate these strategies. Such strategies should seek to balance the desire for optimality in a dialogue with the need for success in terms of the agents individual dialogue goals.

There are two mains benefits of applying dialectical game playing and arguing agents to distributed graph colouring. The first is that agents can find *intelligent* solutions to the conflicts that arise. For example, using knowledge about the history of the MAS, or the results of previous conflict resolution dialogues to inform the current dialogue. These could be used to avoid cyclical repetition of states in the MAS. If a pair of agents have previously been in conflict and the circumstances surrounding the current conflict are similar to those from the initial conflict then a different solution to the conflict might make more sense than repeating the same resolution. The second benefit is that conflicts and their resolution can be analysed after the fact. For example, an agent could justify to its owner the solution that was found in terms of the arguments that were used to pursuade one of the conflicting agents to change state. This is useful in domains where oversight is required over the MAS and its agents.

Two issues can be identified with regards to distributed graph-colouring, the first is the issue of *identification* and the second is the issue of *recognition*. Identification is the ability to find solutions in a distributed rather than a centralised fashion which is important in light of the modern trend towards distributed systems as identified by Wooldridge [146]. In a MAS in the graph-colouring domain, the agents self-organise to find an n-state solution, where n is the number of states available to the agents such that, for example, a four-colour solution equates to a 4-state solution. Identification involves the techniques used to find such a solution.

The need for an additional recognition step is an artifact created by the distribution of the graph colouring problem. In a traditional, centralised graph-colouring technique, all of the data about the system is held in one place and solutions can be easily verified through exhaustive examination of the data. It is usually the case that checking for a solution is a actually a step in the algorithm used to halt the process once a solution is found. When the search for an n-state solution is distributed there is no longer necessarily a centralised collection of data about the state of the system and thus there is no longer a centralised way to verify that a solution has been found. Thus there can arise a situation in wcich a potential solution has been found but there has been no top-down validation that this is the case, the solution thus remains a *potential solution*. When a potential solution has been found it may be necessary, in a distributed system, to check that such a solution is in fact valid and there are a number of ways to do this which have varying degrees of exactitude.

Before considering methods for recognition of potential solutions, two other issues are of relevance, firstly, whether it is actually necessary given the context in which the system operates to verify that a colouring has been found, and secondly, how can such colourings be found in a distributed context. Tackling the first issue first, many real-world solutions requires only good-enough performance rather than optimal performance, e.g. The overhead of finding and verifying optimality has a larger impact on the system than running the system with a low, but manageble, number of conflicts which are tackled as they occur. For example, when assigning channels to distributed wireless nodes, an optimal assignment would ensure that when the wireless network is at maximum capacity there would not be any clashes, e.g. Nodes assigned the same channel that were trying to transmit data at the same time. However, under normal, and possibly even abnormal, loads the network could continue to operate even in the presence of a number of clashes. In these cases, so long as the individual nodes are capable of resolving any clashes as they occur, it is not necessary to verify that there are no clashes in the system. It is in this mode that the Caucus MAS used in the evaluation of communicative performance is run. The aim of the evaluations was not to resolve all conflicts in the MAS but to produce dialogues. As a result it was not necessary to investigate at each step as to whether a colouring had been found. The important idea within the Caucus evaluation framework is that there is the existance of conflict, a motivation for conflict resolution, and a means to achieve such a resolution. Rather than determine outright that there are no conflicts in the system, the agents in the MAS resolve any conflicts as they occur.

For a system aimed at real world deployment dynamic conflict resolution is an important benefit because it means that the system can cope to some degree with the addition of new nodes and edges, as well as changing circumstances. The addition of a new node into a graph may cause a previously colourable graph to no longer be so colourable. In the context of the Caucus MAS such a system would always contain conflicts. The minimum number of such conflicts, the *minimum conflict level*, can be determined for a given MAS such that the number of conflicts in the system will never be below that number. The conflicts left in the system once the minimum conflict level is reached are the *residual conflicts* in the system. Notice that although there is a mechanism to facilitate resolution of conflicts, each time a residual conflict is resolved it does not remove the conflict from the system but merely serves to *propagate* the conflict from its current position in the MAS to another position. For example if two agents are in conflict and are members of a MAS in which there is at least one residual conflict, this means that there are no conflict-free states available to the conflicting agents, e.g. no matter which state either of the conflicting agents change to they will be brought into conflict with at least one of their other neighbours.

The resolution of conflicts in any MAS based upon the graph-colouring domain can be conceived in terms of such conflict propagation. Conflict propagation is also a way to categorise the conflicts that arise, for example, conflicts can be either *resolvable* or *residual*. Whereas residual conflicts cannot be resolved because of the structure of the MAS and the constraints on the system, resolvable conflicts can be removed from the MAS. This occurs in two ways, *immediately resolvable conflicts* can be removed immediately from the MAS when an agent changes to a conflict-free state. If however the conflicting agents have no conflict-free states then the conflict is propogated to a new position in the MAS and the conflict is a *delayed resolvable conflict*. The delayed resolvable conflict is eventually resolved once the conflict has propogated to a location of the MAS where there are conflict-free states.

Conflict resolution in a graph-colouring domain situated MAS equates directly to finding an *n*-state colouring for the vertices of a graph. Once there are no conflicts left in the MAS then a colouring of the graph has been identified. In the Caucus MAS used for the game evaluations, each conflict results in a single dialogue between the conflicting agents. Dependent upon the dialectical game, the knowledge base, the available argument templates, and the character, strategy and tactics of each agent, each dialogue that occurs may be an extended exchange of messages between the communicating agents until a termination point is reached. At the termination point, each dialogue results in either success or failure from the perspective of the initiating agent. A successful dialogue is one in which the initiating agent has caused the responding agent to accept the initiating agents goal, and an unsuccessful dialogue is one in which the initiating agent has not caused the responding agent to accept the initiating agents goal.

Of prime importance when putting together a MAS to find distributed solutions to graph-colouring problems is to ensure that the arguments available are actually sufficient to find a solution. It is quite straightforward to find a minimal argument and such an argument is akin to a brute force approach to finding a solution, requiring only that the initiating agent proposes that the respondent change state because they are in conflict. If one of the conflicting agents changes state each time a conflict is found then eventually a solution will be found if such exists. This is analogous to a basic algorithm which searches through every pair of neighbours for a conflict and resolves the conflict by changing the state of one of the conflicting agents. This however may require a great number of state changes to be performed. If there is a cost associated with each state change then a more advanced approach that minimises the number of state changes is appropriate. These more advanced approaches can be implemented by making a wider range of argument templates available to the agents. This can demonstrated by comparing two of the argument templates used by the Caucus MAS. The first argument states that an agent should change state purely because it is in conflict. The second argument states that an agent should change state because it is less stable than another agent. Stability is calculated by examining the number of conflicts in a given area, known as the *locale*, around each conflicting agent. If the stability of both agents is equal for the given locale then the radius of the locale can be increased until the agents find that one of them is more stable than the other. The second argument requires more computational power and more knowledge but has benefits over the first argument. The rationale for the argument from agent stability is that it is more difficult to resolve all of the conflicts in an area of high conflict density compared with an area of low conflict density because there is more chance in an area of high conflict density that there will not be any conflict-free states. An agent who is less stable is in an area of higher conflict density and should therefore change state to reduce the conflict density in that area. This kind of argument, rather than relying merely on the existance of a conflict between two agents, relies upon a wider context of density of conflicts in a given area of the MAS and seeks to reduce that density of conflicts, albeit only through the resolution of one conflict.

There are two basic approaches to the recognition of a solution. The first is strict and the second is lax, additionally both strict and lax recognitions can be pursued through centralised or decentralised means. Strict, centralised recognition involves the system-wide collection of agent state data and the exhaustive search of that data to determine whether or not a solution has been found. In the Caucus MAS this search is conducted by the environment agent which continuously searches for conflicts within its internal representation of the state of the agent society. The env agent keeps an internal count of messages received and time is measured in discrete quanta where one timestep equates to one message being received. At each time step the environment agent checks its representation of the environment for the presence of conflicts. If conflicts are discovered then a solution has not been found. However if there are no conflicts in the society then the problem contained in the initial scenario has been solved.

A lax, distributed method for detecting graph-colouring solutions is through the agents themselves. If an agent has no conflicts then it does not initiate any dialogues. If all agents are conflict free then the number of dialogue messages in the MAS will fall to zero. However, because the JUDE platform does not provide any instrumenting for the agent society the global number of messages in the MAS at any given point is difficult to ascertain. Similarly, if there are no dialogues occuring in the MAS then there will be no state update messages received by the environment agent as a result of those dialogues completing. When there are no state update messages received then the Environment agent can infer that a solution has been found although it must still do an exhaustive check to determine this for certain. This is because the lack of state update messages for any period of time may be due to a long running dialogue rather than the lack of conflicts in the MAS. These two approaches are termed lax because they do not determine for certain that a solution has been found but are very suggestive that that is the case.

The MASADA components have been applied to an exploration of distributed graph-colouring. A society of Caucus core agents was used to identify a solution to a number of graph colouring problem and the Caucus environment agent was used to provide strict centralised recognition of any identified solutions. Recogntion was performed by examining the state of the MAS after each state update message was received by the environment agent. For each agent the number of conflicts was calculated. A solution was found once the number of conflicts had fallen to zero. The initial setup of a Caucus MAS for the purpose of exploring graph colouring is similar to that for the evaluations of communicative performance, however scenarios have been selected that possess four-colourings. The agents load their initial knowledge and identify any conflicts that exist with their neighbours. Once a conflict has been identified an agent then initiates a dialogue with their conflicting neighbour where the goal of the dialogue is to resolve the conflict. At the end of each dialogue the conflict is resolved by one or the other conflicting agents committing to change state and subsequently transmitting a state update message to the environment agent. The environment agent keeps track of all agents in the MAS, their relationships with other agents, and the colour states of all agents. This information is used to determine whether a solution has been found.

Three scenarios were explored using a Caucus MAS containing core agents and an environment agent. These scenarios were hand generated for the current investigation. In the longer term, an exhaustive investigation must be carried out which will require automated graph generation but hand generation suffices for the number of investigations carried out here. The first scenario specified ten agents with forty relationships between those agents which equates to a graph containing ten vertices and twenty edges. This scenario was named 10:20 to indicate the ratio of vertices to edges. The initial state of this graph is illustrated in figure 6.7 and shows all agents having the same initial colour state which guarantees that there will be conflicts in the system. The number of messages per dialogue and the number of state changes that occured in the MAS were recorded. Once a colouring had been found, such as that illustrated in figure 6.8, the MAS was stopped. At this point there were no longer any conflicts in the system so the agents were not motivated to initiate any new dialogues. Figures 6.9 and 6.10 show the initial setup and one possible colouring for the largest scenario investigated which had a vertice to edge ratio of 20:45.

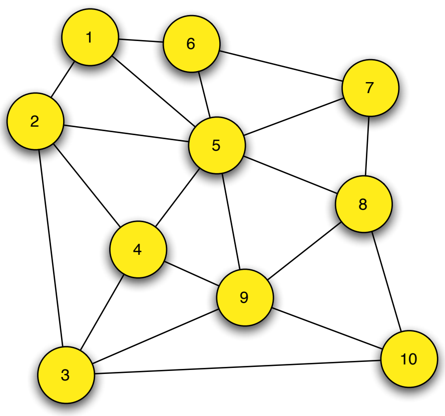

Figure 6.7: The initial setup of the 10:20 scenario

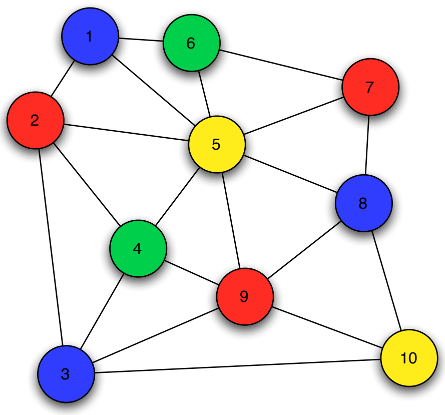

Figure 6.8: A colouring of the 10:20 scenario

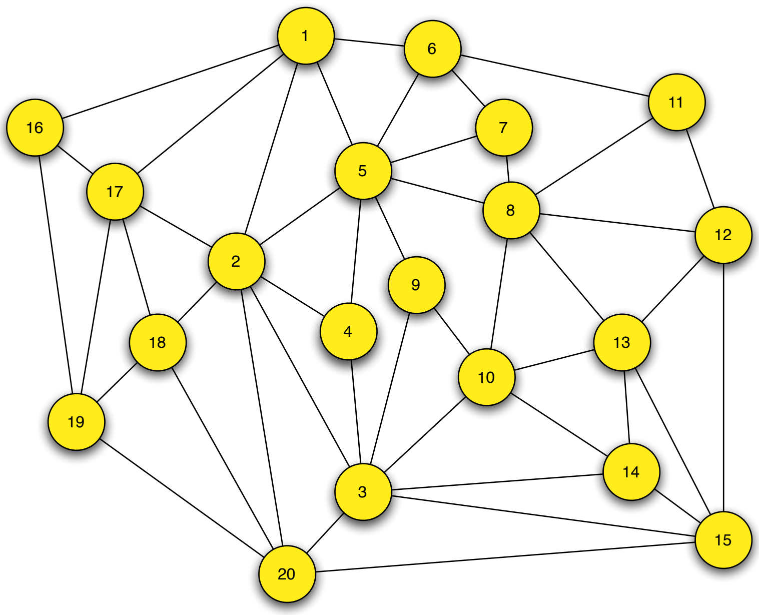

Figure 6.9: The initial setup of the 20:45 scenario

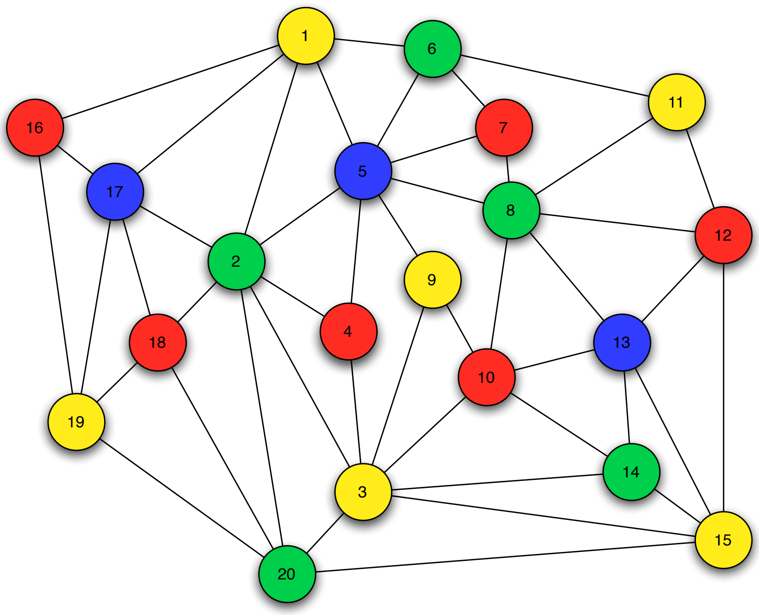

Figure 6.10: A colouring of the 20:45 scenario

Each scenario was explored using agents who engaged in dialogue to resolve each conflict that was found. There is a one to one correlation between each conflict, each pair of neighbouring agents who are in the same colour state, and each dialogue that resolves the conflict. For each exploration of a given scenario a single type of dialectical game was used to find a colouring of the graph specified in the scenario. So, for example, for scenario 10:20, a colouring was found using each of H, DC, CBZ, DL3, and PPD_0 to govern the dialogues in which each conflict was resolved. This process was iterated ten times for each dialectical game under each scenario and the average number of state changes per scenario per dialectical game was calculated. Table 6.3 collects the results of these explorations and displays the average number of state changes for each dialectical game under each of the three scenarios.

Game	**10:10**	**10:20**	**15:35**	**20:20**	**20:45**
CB	93	97	144	128	225
DC	79	98	172	139	182
DL3	86	123	163	108	238
H	75	114	166	116	213
PPD_0	76	95	178	109	209

Table 6.3: Average number of state changes per colouring for a range of dialectical games under three different scenarios

These results are demonstrated more clearly in the graph in figure 6.11 which shows how the average number of state changes per colouring are fairly constant with respect to each scenario. This was to be expected because the only changes between investigations are either the dialectical game or the scenario. Additionally, the range of stereotypcial arguments that agents can construct is kept constant throughout all of the investigations so there is no additional influence in the MAS to affect how the agents select their new state at the resolution of each dialogue. This results in the games behaving similarly to each other. Each iteration of a scenario starts with the same amount of conflicts, because all agents are set initially to the same colour, from that point onwards, each individual conflict motivates a new dialogue to resolve the conflict, and the outcome of the dialogue is exactly one state change. So, although individual dialogues can vary greatly in average length, dependent upon which game is played, the effect upon the number of state changes should be minimal.

There is trend such that as the number of vertices and edges in the graph increase so the average number of state changes required to find a colouring increases. This is expected because as the number of vertices and edges increases so the number of possible incomplete colourings, graphs in which not all conflicts have been resolved, increases. It appears also that the density of edges in the graph has more effect on the resultant number of state changes than the density of vertices. This is because, although more vertices in a graph affect the number of possible arrangements of states, the number of edges increases the potential for conflict, for example two agents who are not neighbours cannot, given the current setup, be brought into conflict but an edge between those two agent introduces a new conflict vector. It can be concluded that the number of vertices does not greatly affect the resultant number of state changes because merely adding another vertice to a graph does not mean that there is the potential for more conflicts, however the addition of new edges will lead to the potential for new conflicts. There is not a great deal of difference between the average number of state changes required to find a colouring for the 10:20 scenario compared with the 20:20 scenario, despite the latter having twice as many vertices. This supports the previous conclusion that edge density has a greater effect than vertex density.

There is some variation in the average number of state changes per colouring, for example there is quite a large dip in the results for DC in scenario 20:45 but these are believed to be statistical artifacts. As a result the dialectical games overall appear to perform fairly similarly in terms of finding a colouring. It would be interesting, in future to investigations to determine how better strategies could be used to reduce the number of state changes required to find a colouring. There is presently no intelligent colour state selection in the MAS. When an agent commits to change state it selects a conflict-free state, if such is available, or randomly selects a valid colour state otherwise. An agent could however use its knowledge of its own past colour states and the states of its neighbours agents, and wider locale to make an intelligent selection of its next colour state. A similar approach could use the same setup to explore agents coalitions in which agents work together to find a solution rather than working independently as is the case at present.

The issue of agents intelligently selecting colour states so as to avoid conflict, or to minimise the number of state changes, moves the discussion in the direction of strategy which is an issue that this thesis has barely touched upon. How should an agent play any of the games in order to achieve it's goals or, put more simply, what would be a good strategy for the agents to use? There has been very little work into strategic play of dialectical games and the current climate tends towards identification of heuristics for agents to follow rather than full blown strategies. Oren *et al* have suggested, for example, that agents should tend towards laconicism with the heuristic that "loose lips sink ships" [92]. This kind of heuristic for agents is at odds with the kind of message directed towards human locutors however. Gilbert suggests that nobody ever loses an argument unless they are particularly attached to a specific position [38]. The idea being that at the end of an argument you can always be a winner if you leave the dialogue with more knowledge than you had at the beginning. The focus of this thesis, at least in terms of strategic play, has been to lay the groundwork for an examination of the kinds of strategies that agents can use. Until now there has been no common representation of dialectical games rules, and hence no framework within which to represent possible strategies. Any work which could be performed would be either tied to a particular game, because of the non-common representation, or at a much higher level than strategic play, and more akin to heuristics. This is evidenced by Yuan who developed a set of advanced strategies for playing Mackenzie's DC but which were tied specifically to the locutions available in DC [149]. The A4A provides the first step away from the development of such strategies which rely upon particular locutions and towards a method for representing strategy which relies instead upon the requirements and effects associated with each locution, regardless of the label associated with it. Such an approach is, however, one direction included in the list of further work to base upon that carried out in this thesis.

In summation, although each game performs similarly with respect to the number of state changes required to find a colouring fo each scenario, the average number of messages exchanged per dialogue is similar for each game to the trends demonstrated in the communicative performance evaluations for the same games. Because each conflict motivates a single dialogue, and each dialogue results in a state change for one of the conflicting agents regardless of the protocol that is used to govern the dialogue, it would appear that the games are equipotent with respect to termination status.

The most important conclusion that can be drawn however is that these results demonstrate that dialectical games need not be considered merely as protocols for controlling the legal interactions of parties during a dialogue but that they can also be used as a part of a technique for solving certain distributed problems.

6.4 Summary

This chapter has investigated dialectical game evaluation and began with a discussion of the ways in which such evaluations can be undertaken. Two distinct approaches were presented, a non-empirical approach based upon an analysis of a game's components and rules, and an empirical approach based upon the dialogues that are constructed when a game is played.

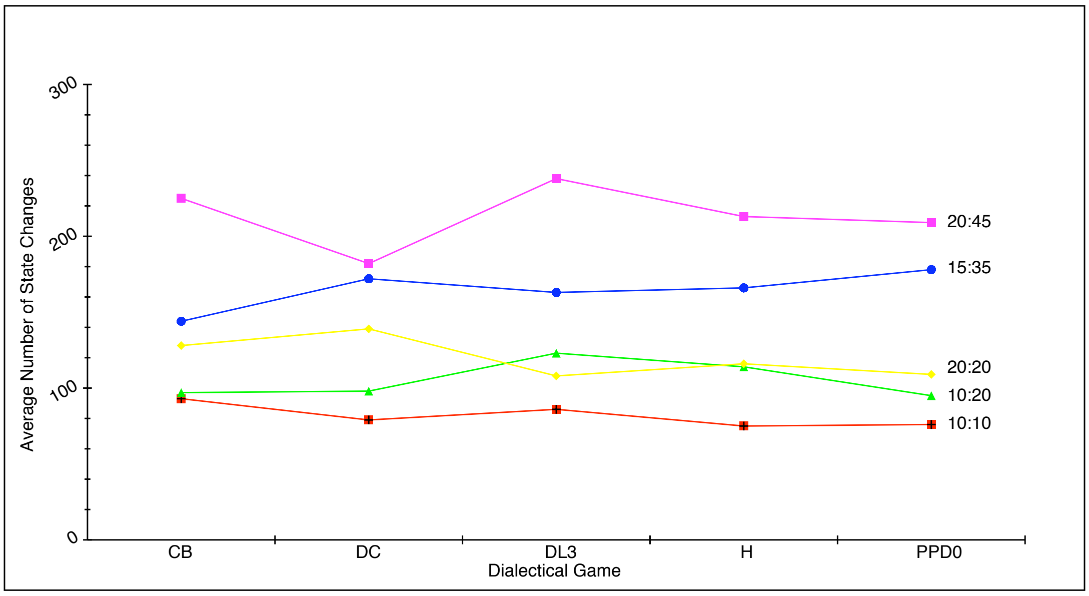

Figure 6.11: Graph showing the average number of state changes per scenario for each dialectical game

Both the MAS implementation process and the actual utilisation of a MAS in the later evaluations are useful exercises that provide insight into issues in constructing arguing agents. The process of implementing a MAS whose agents use dialectical games to regulate their argumententative interactions proved as useful an exercise as the later evaluation which made use of this implementation. The process of implementation and running dialectical games in a MAS has shed light on a number of areas and issues that have either been neglected or have not been tackled because basic elements required to investigate the issues were not yet mature enough to support a proper investigation. The most glaring issues that become apparent are the need for strategic considerations and the need for termination conditions as a necessary part of an inter-agent communication protocol. The related concepts of strategy, tactics, heuristics, and dialogical planning are important because although the rules of a dialectical game enable an agent to determine the kinds of things that it can legally say, the game does not help the agent to select what it should say.

It became apparent during the investigations that merely representing dialectical games in an agent is not sufficient to enable the agent to play the games well. Dialogues constructed by agents using naive strategies were far from optimal and suggest the need to perform an in depth investigation of the strategies and tactics that an agent can pursue in order to achieve its dialogical goals. Without a sufficient capability for reasoning and strategy there can be a huge impact upon the communicative efficiency of any dialectical game that is used. This was shown in the evaluation in which the agents had no strategic capability. The resulting dialogues were long and inefficient. Because the same games can produce very short dialogues if played in a different way, it follows that it is the way that the agents play a game that can have a large effect on both the outcome and the characteristics of a dialogue, although this effect is reduced by the actual formulation of rules for the game which could be formulated to produce, for example, dialogues of constant length by formulating a dialectically-closed game that prohibits all cyclical dialectical structures. Furthermore in order to meet its own requirements and dialogical goals an agent needs to be able to plan ahead, both to be able to avoid its opponent from trapping it into commitments which undermine its own aims and also in order to maneuver its opponent into commitments consistent with the agents goals. Agents do not hold shared dialogue goals so it is possible that either participant is unaware of the reason why they are engaged in the dialogue and is merely responding to received utterances in line with the requirements of the protocol, and so may be maneuvered into a set of commitments leading to an obligation to accept a commitment which it would otherwise not do and which is commensurate with the goal of the initiating participant.

Finally, this chapter has demonstrated that dialectical games can have a role in the search for solutions to distributed problems such as graph-colouring. Because graph-colouring problems can be considered as analogues of many real-world problems it is important to explore new techniques that could lead to improvements in the search for solutions. Whereas dialectical games have previously been proposed as a means to regulate communication between software agents, it would appear that these games could have a much wider range of possible uses and applications which need to be further explored.

Chapter 7

Conclusions

ROS: I mean, what exactly do you do?
PLAYER: We keep to our usual stuff, more or less, only inside out. We do on stage things that are supposed to happen off. Which is a kind of integrity, if you look on every exit as being an entrance somewhere else.
– Tom Stoppard, *Rosencrantz and Guildenstern are Dead*

THIS chapter begins by summarising the contributions made to the study of dialectical games and MAS by the research presented in this thesis. The summary is followed by some possible future research directions based upon questions that have arisen during the production of this thesis and its associated software.

7.1 Contributions

Chapter 3 began by motivating the need for a unification of the disparate approaches to representing dialectical games. Two goals were identified in the context of the increasing popularity of these games, mainly as protocols for structuring agent interaction, but also as tools for coordinating and tackling various distributed problems. The first was the need for a consistent means of representation, and the second was the ability to harness the various features of different games so that they could be recombined to form new games and thus explore the dialectical game space. Consistent representation is important because it can form the basis for implementational frameworks to reduce the costs of developing software systems that use dialectical games and thus reduce the possible risks and complexity associated of adopting dialectical games. To construct such a consistent representation required analysis of the existing games to determine the kinds of components and rules that those games use, to identify the range of representational styles that are used, and to see how ostensibly similar components vary between different game developers. One aspect that was problematic was that the definition of dialectical games introduced by Hamblin was very loose, and this has been adhered to only roughly by subsequent researchers. This has meant that it is hard to determine limits for what does and what does not constitute a dialectical game as opposed merely to a protocol for interaction. A dialectical game may therefore be played by two or more players, uttering one or more locutions per turn and thereby possibly incurring commitments, or accumulating some other such dialogue artifacts, along the way. Dialectical games also specify rules which restrict the locutions that are available to the players at every point in a dialogue. It is this aspect, the specification of *dialectical interactions* which make a dialectical game dialectical. An underlying theme of chapters 3 and 4 is that dialectical game research should move away from individual games towards an examination of a space of possible games. Such a space is delineated by, for example, the number of players, the range of dialogue artifacts, the range of locutions, and the specified turn structure. By selecting particular ranges for these parameters a specific game, a distinct point in the game space, can be identified. The major contribution of chapter 3 therefore is to identify the range of components that are found in the extant dialectical games and to determine the ways that these components can be manipulated. Although many researchers have taken existing games from the literature, such as Mackenzie's DC, and made modifications as necessary, occasionally there have been novel additions to the games. Dialectical game research has thus progressed by a process of small increments but there has been, until now, no attempt to draw the disparate features of the extant games together into a cohesive whole. This is an important contribution that underlies the development of common implementation frameworks and facilitates the movement of dialectical research from an *ad hoc* activity towards an engineering discipline.

Supplementary to the idea a space of dialectical games is the idea of dialogue progressions. If it is a simple matter to specify new points within the game space then it should also be a simple matter to move from a game played according to one set of rules to a game played according to another set of rules. This is based upon an observation by Walton and Krabbe [138] that specific dialectical games should equate to specific dialogue types and contexts of dialogical interaction. Furthermore, although a dialogue may begin as one type it may shift to another type entirely before completion, or admit multiple embedding of other types of sub-dialogue. These kind of progressions between dialectical games are an important aspect of a framework for representing dialectical games in the context of a game space and provide additional flexibility that could not be reliably achieved using a single monolithic dialectical games that tried to account for all contexts of dialogue. Chapter 4 thus introduces a unified layout, called the game schema, for representing specific dialectical games, and a more generalised meta-game for defining the game space. Together these items form the theoretical portion of the Architecture for Argumentation (A4A). As well as the theoretical aspect there is the implemented A4A computational framework which enables dialectical games to be designed, rapidly developed, implemented and deployed within other software. This is novel within dialectical game research and parallels the wider movement in MAS towards frameworks for implementation which reduce the cost and complexity of development. The range of capabilities and components of both the game schema and meta-game are based directly upon the components and manipulations identified in the analysis from chapter 3. A great number of games can be specified in the A4A which also includes a practical means of implementing progressions from dialogue carried out according to the rules of one game schema to a dialogue played according to the rules of another game schema within a single overarching dialogue. Whilst there has been much theory devoted to the types of shifts between and embeddings of sub-dialogues that could occur during a dialogue, the author is not aware of any theoretical and practical systems for representing and implementing dialectical games with dialogue progressions. The representational use of the A4A is illustrated by the introduction of two new dialectical games that were used to explore the fallacy of bargaining in argumentative dialogue. These games include PP_0, a game for persuasion dialogues, and NP_0, a game for negotiation dialogues. The games are then used to explore the idea that even though one agent possibly cannot successfully persuade its opponent having no sufficiently persuasive argument, the agents can still work towards an acceptable solution to their disagreement. This is achieved by beginning a persuasion dialogue and, once the agents have exhausted their available persuasive arguments or at the very least discharged any burden of proof associated with their positions, they then progress to a negotiation dialogue which they use to try and reach a mutually acceptable solution. This kind of progression, from persuasion to negotiation has not previously been explored in the context of dialectical games although it is characterised, under certain conditions, by the fallacy of bargaining. The future of dialectical games lies in developing games that characterise the interaction of agents in specific contexts of dialogue. Flexible arguing agents require the ability to progress between different games as their context of interaction develops and changes, an ability that that A4A provides. The A4A therefore represents the first implemented framework for development of multiple disparate dialectical games that uses a unified schema for layout and that facilitates progressions between individual games. Given a software framework such as the A4A, the process of developing arguing agents can be explored. The most pressing issue a developer initially faces is which game to implement. An important aspect of the A4A is that the implementation provides a core runtime into which a specific game schema is loaded. This means that new games can be formulated and implemented without needing to develop and debug new source code. This does not completely avoid the issue of determining a games suitability to a given context but it does mean that the risk of selecting an unsuitable or inappropriate game is greatly reduced because whichever games is implemented initially can be simply and rapidly changed at a later point.

Chapter 5 looked at the process of building arguing agents that use dialectical games. Two projects were introduced, MASADA and the *i*-Xchange, which focussed upon two complementary areas of dialectical games and MAS. MASADA, the MAS and Dialectical Game Architecture, focussed upon the required components that an agent needs to enable it to play dialectical games. Whilst the A4A facilitates the representation of a game's rules and the recording of information associated with playing a game, it doesn't incorporate any kind of representation of what the game is about, i.e. the knowledge domain, of how the game should be played, be it from the point of view of an agent reasoning about how to select a merely relevant utterance or an agent attempting to select the optimal utterance to play in order to achieve some strategy. Two components were developed for MASADA, named Colloqui and Caucus. Colloqui is a component that encapsulates an instance of the A4A to represent dialectical games, but also incorporates elements of knowledge representation and reasoning. The knowledge representation scheme is based upon an implementation of Minsky's frame-based knowledge representation scheme [85] with stored procedures and is used to enable an individual agent to record knowledge about its environment, to express that knowledge in the form of statements about the environment, and to reason about which expression to use as the content of a move within a dialectical game. A knowledge domain was constructed based upon the activity of graph colouring and this was used to instantiate the agents knowledge base. Because dialectical games include aspects of argumentation a template-based system was developed which was used to specify a range of concepts stored in the knowledge base and premises and conclusions of a stereotypical argument from the graph-colouring knowledge domain. This approach enables given frames of knowledge to be interrelated to form arguments. This means that the agents knowledge and the arguments that can be constructed given the knowledge base are specified and represented separately which is an aid to the investigation of dialectical games allowing a fine control over both aspects. The Caucus component specifies modules containing domain specific behaviours for various types of agent. The agent framework that underlies MASADA is JUDE industrial agent platform [67] which uses modules to supply particular capabilities to the individual agents in a MAS. It is in the Caucus component that these modules are specified. As well as the arguing agents that use the Colloqui component, there are environment agents that enable the arguing agents to sense various aspects of their environment as well as a GUI agent to provide a user interface to the MAS. The Caucus component implements modules for each of these three types of agent. Whilst there has been much theoretical research into dialectical games and their use in MAS there has been a dearth of implemented MAS software in which the agents can play dialectical games.

The most important issues concerning implementing arguing agents, once a dialectical game has been implemented, are that knowledge has to be structured and represented sufficiently for the agents to be able to reason about their situation and also to be able to communicate that situation to other agents during a dialogue. Without an adequate reasoning process, agents are unable to select relevant utterances from the set of merely legal utterances, and without strategic capabilities there is little chance of agents engaging in optimal dialogues merely because they can play dialectical games. The development of MASADA added to the small number of existing MAS that use dialectical games, allowed the exploration of wider implementational issues such as identifying the minimal range of additional components that an agent requires in order to be capable of playing dialectical games, and shed light on otherwise neglected issues of how to play dialectical games that are pertinent once an agent containing the necessary minimal components is constructed.

The *i*-Xchange project sought to investigate the issues relevant to building agents that incorporate a range of independent capabilities into a cohesive framework. This contrasts with the MASADA development which was focussed solely on facilitating dialectical game play and so the components of MASADA agents are coherent and developed in pursuit of a single goal. The *i*-Xchange however was more concerned with the integration of disparate components and the construction of coherent agent architectures to underlie the development of agents with disparate and varied capabilities. The *i*-Xchange agents are novel because no other implemented MAS has constructed such a coherent architecture to underpin different, and to some degree, competing capabilities. An additional aspect that made the *i*-Xchange agents particularly interesting was the incorporation of agents from two separate agent platforms, Jack and JUDE, into a single coherent MAS. As more agents are developed and used in real-world situations there will be an increasing need to construct agents that have a range of capabilities and that are built using a range of frameworks and platforms. The research performed in the *i*-Xchange will provide a basis for the development of such future systems.

A framework such as the A4A reduces the effort required to implement a dialectical game. Once the A4A is embedded into an application, the additional effort required to change the dialectical game requires only a new file to be loaded. This means that the decision about which game the agents will use does not need to be made at the beginning of design but can be postponed until later in the development cycle once more information about the agent's operating environment has been identified. However this does not mitigate the need to evaluate dialectical games but merely postpones it. Chapter 6 investigated how games that can be specified using the A4A can be evaluated using a Caucus MAS built from the MASADA components. Two aspects of evaluation were identified, non-empirical evaluation which can be carried out simply by examination of the specification of a game's rules and components, empirical evaluation which is based upon the idea that to determine how a game performs in a given context requires that the game be played in that context and the results, the dialogues, be examined. Both empirical and non-empirical evaluations were carried out on a number of extant games. These were used to identify how and why certain games perform better in certain circumstance than other games. Empirical evaluations were carried out to determine the communicative performance of H, DC, CBZ, DL3, and PPD_0 when these games were played using the graph-colouring knowledge domain. Although a single knowledge domain cannot be used to conclusively state that a given game performs better, or worse, than another game in all circumstances it can be used to state that there is a difference in performance between some games in the circumstances tested. A more comprehensive evaluation will require that a range of knowledge domains are implemented and the MASADA components will form the basis of such an implementation. The issue of evaluation has not been explored before in terms of a practical implemented system for comparing different games and there has been no comparison of how a existing dialectical games perform. In addition to the use of MASADA to investigate empirical evaluations and comparisons of dialectical games the same components were used to investigate the use of dialectical games as a method for solving distributed graph-colouring problems. This is important because there are many parallels between problems in distributed graph-colouring and issues in real-world domains such as *ad hoc* radio networks. Dialectical games were thus explored as an alternative approach to solving these problems and as an alternative to using the games purely for communicative purposes. A number of important points were identified: firstly that the particular dialectical game used to find a colouring does not necessarily have a great effect upon the number of state changes that lead to the colouring because there is a one to one correlation between each state change and each dialogue carried out to resolve a conflict. However the size of the dialogue that results in the state change can vary greatly between different games. Another important point was that there is a difference between finding a solution and verifying that a solution has indeed been found. This difference is quite pronounced in distributed graph colouring using MAS because there is not necessarily a system wide collection of information about the state of the system that can be investigated. Although the environment agent of the Caucus MAS was used to verify the existence of a solution, a number of alternative approaches were also identified. Firstly, it might not actually be necessary to remove all conflicts from the MAS, instead relying upon the agents to deal with each conflict as it arises. Secondly, there are more indirect ways of ascertaining whether a colouring might have been found. For example, by monitoring the number of messages that are being passed or the number of state changes. Only when the number of state changes, or number of messages being passed falls to zero is it likely that a colouring has been found. By using such implicit techniques the need to verify whether a colouring has been found after each update can be removed and such an explicit verification step is then only required on occasions where the implicit techniques suggest that a colouring might have been found. This has two important benefits, firstly that in distributed systems without centralised monitoring it may still be possible to determine that a solution may have been found as opposed to having no idea. and secondly that the number of times that a verification step needs to be performed can be greatly reduced thereby producing savings in computational resources.

7.2 Future Directions

The research undertaken for this thesis has produced a number of novel features and findings which will contribute to the fields of dialectical game research, argumentation theory, multiagent systems, and distributed graph colouring. However, for every issue that has been investigated in this thesis many more issues have been brought to light. These issues will form the basis of both refinements and extensions to the A4A and MASADA software and associated problems. Some of these issues include;

Control/Meta-Level Dialogues Colloqui makes use of a rudimentary meta-level dialectical game that includes only the single $<$ *withdraw, (–)* $>$ locution. Whilst this is sufficient for the evaluative scenarios examined in chapter 6 it is unlikely to be sufficient to support disparate agents in a heterogeneous society who require to specify which schema to load to regulate their pending dialogue. A second situation in which a meta-level dialogue is useful is to enable agents to invite other agents to take part in a multi party dialogue or to aid agents in recruiting a referee agent to arbitrate during a dialogue. The last situation in which meta-level and control dialogues are important is in dealing with rule breaking and the need to either resume a dialogue from an earlier point, discarding the unacceptable utterances, or to continue with sanctions applied to the rule-breaking agent. However, even with sanctions applied during the game in which they were incurred, it may be necessary to incur a societal impact for persistent rule-breaking. This assumes a social model, which doesn't currently exist in MASADA, in which agents have some reputation for honesty and square dealing in their dialectical game playing. The notion that a dialectical game is used to resolve conflicts and that some of the decisions made and conclusions reached during a dialogue have an effect on the wider agent society is important. Again, similar to the issue with rule breaking, sanctions are required that could be applied to any agent who fails to honour commitments incurred during a dialogue.

Error Recovery The extant games assume a robust and reliable communication network in which messages are neither lost nor garbled. If this does happen then agents need to be able to recover to a suitable earlier point in the dialogue and resend any lost updates. A facility to enable this is not currently included in the MASADA components and would prove a useful function when the components are used in real-world agent systems.

Dialectical Reasoning A lot of work has been conducted into researching the kinds of rules that can be used to regulate dialectical exchanges between a dialogue's participants. However these rules only regulate the kinds of things that can be said in terms of the performative verbs that can be allied with statement variables. In Colloqui a knowledge base instantiates statements which express the agents knowledge of its environment however the reasoning process that determines which expressions are relevant and thus eligible for selection to construct an utterance are both rudimentary and coarse-grained.

Heuristics, Tactics, Strategies, & Dialogue Planning Similar to the situation regarding dialectical reasoning, very little work has been conducted into how a party should engage in a dialogue once it has determined the set of utterances that it can say and the subset of those utterances that are legal to say. As was demonstrated in chapter 6, the way in which a game is played has nearly as large an effect on the resulting dialogue as the formulation of rules that govern the legality of each move. One approach to this was pursued by Yuan [148] for example investigates dialogical heuristics in the context of the game, '*DE*'. Recently, Oren *et al.* have investigated heuristics for engaging in argumentative dialogue [92, 93]. However with the advent of more generic dialectical game frameworks, such as that embodied by the A4A, the agents concerned are not confined to a single set of locutions. This means that the approach taken by Yuan is not applicable to the A4A because the heuristics are specific to the locutions of DE and are not generalised to other games with different locution sets. A possible non-game specific approach involves formulating heuristics and strategies in terms of the requirements and effects that are used to specify locutions in the A4A.

Explore Applications to Distributed Problem Solving Whilst graph-colouring was used to provide a knowledge domain in which to situate agents for the purposes of evaluating dialectical games, dialectical game regulated dialogues could equally be applied in the other direction as a method to distribute the search for solutions within graph-colouring and related distributed problems. Rudimentary notions of distributed solution identification were discussed in section 6.3 however more solid ideas need to be formulated and explored. This is especially important as problems that might traditionally have been tackled from a centralised perspective are increasingly reappraised in a distributed light.

Colloqui: Enhanced Scenarios The graph-colouring knowledge domain could be extended with additional parameters to enable construction of more sophisticated scenarios. A limit of the current domain is that all agents are acting within a single mode, the need to resolve conflicts based upon the colour state of individual agents. Agents engaged in real world dialogues often have conflicting preferences of their own which inform how they engage in argumentative dialogue. Such a multimodal approach in which agents must build their own position in light of conflicting preferences could make use of parameters such as:

1. Mutative Number of States - The GC_0 domain uses only four colour states however this number could be varied to simulate the abundance or scarcity of available resources.
2. State Change Penalties - Agents are currently able to change state without incurring any penalty for those changes. However state changes could be considered a finite resource which incurs a cost for consumption. Agents would then need to use strategies which minimise costs in their joint searches for solutions.
3. Search For Particular Colourings - The GC_0 domain is concerned primarily with generating dialogues, and has as a secondary aim the distributed solution of graph colouring problems. In the study of graph colouring theory there are many other possible colourings to look for including edge colouring, list colouring, and harmonious colourings to name but three. Some areas of application of graph colouring are timetabling, scheduling and resource allocation. Scenarios based upon the search for particular colourings would suggest new applications of arguing agent systems to those areas.

4. Communication Channels - Agents are only able to communicate directly with their neighbours but if agents were allowed to engage in dialogues with other members of the MAS then scenarios could be explored in which the agents form coalitions and work together to solve problems.

Colloqui: Runtime Domain Selection The current Colloqui implementation is limited to problems within the graph-colouring domain. Whilst such problems are a useful initial step toward making comparative dialectical game evaluations, it is likely that a game that performs well in the graph-colouring domain will not perform equally well in all knowledge domains. To mitigate this criticism evaluations should be performed in a range of knowledge domains.

Colloqui: Runtime Reasoning Selection Whilst Colloqui has been used to explore some rudimentary reasoning processes the current reasoning process implementation is dependent upon the graph-colouring domain. If runtime domain selection is enabled then the reasoning process needs to be altered to make it abstract with regards to the specifics of the knowledge domain.

In addition to the specific enhancement and directions outlined above, the components of the A4A suggest a space of possible games that it would be interesting to explore. In addition there are a wide range of possible GC_0 based scenarios, and new knowledge domains that could be developed to gain a deeper and more comprehensive understanding of dialectical games, both as general model of dialogical interaction and as applied specifically to multiagent communication. It is necessary to explore a range of domains for the same basic reason that multiple problems are used to test theorem provers [124], because a single problem cannot be used as the sole basis for evaluation. Without performing a range of tests under a range of conditions it cannot be said categorically that a given game generally performs differently.

7.3 Aims Revisited

At the beginning of this thesis a number of questions were asked. Answers to these questions are presented here:

1. What components do dialectical games use?

Dialectical games make use of a range of constituent components that were explored in detail in chapter 3. To summarise they include:

Number of moves per turn May be either a single move or multiple moves per turn.

Turn progression Turns may be organised in a strict manner or a liberal manner.

Locution: Performative Performatives broadly identify the kinds of locution that a game supports

Locution: Content Token: Arrangement The content of a locution can contain single or multiple content tokens

Locution: Content Token: Status Logical and argument theoretic relations between tokens

Boards: Dialogue transcripts record the sequence of turns. Other boards store sets of tokens and are either individually owned or shared, have publicly visible contents or private contents, and are either static or dynamic.

Tokens: Commitment to content tokens, locutions, immediate consequence conditionals, justification sequences.

Move Legality: Formulated in terms of prior moves, commitment store contents, form of content tokens, and player roles.

Locutional Effects: Manipulations Move commitments into and out of commitment stores and assign roles to players.

Locutional Effects: Responses Specify a set of mandatory responses that can be played subsequent to a given locution.

2. Can a unified representation of extant games and their constituent components be constructed?

   Such a unified representation can be constructed and was detailed in chapter 4 where the A4A, a framework for representing, exploring, implementing, and deploying dialectical game, was introduced.

3. Can a software framework be developed to support the rapid development and deployment of dialectical games, and to reduce the effort required to do so?

   The answer to this question was explored in chapter 4 in detail. A software framework to reduce implementational effort can and has been constructed in the form of the A4A.

4. What additional capabilities do agents need in order to play dialectical games?

   Chapter 5 explored the additional requirements of arguing agents once they have a dialectical game representation. The minimal requirements are for domain-specific knowledge representation and reasoning.

5. How do these additional capabilities affect the development of intelligent arguing agents when an agent:

   (a) is developed as a single cohesive software application?

   (b) has multiple developers who are incorporating competing capabilities?

   The MASADA and *i*-Xchange projects respectively explored these questions. MASADA demonstrated that arguing agents can be built as a single cohesive software application using components to support dialectical game playing, knowledge representation and reasoning. The *i*-Xchange project demonstrated that cohesive agent systems can be built that cohesively integrate disparate capabilities such as planning, social reasoning and dialectical game play.

6. Given the plethora of extant games and the huge space of possible games, how can games be evaluated in terms of their suitability to a given context?

   Chapter 6 demonstrated that games can be evaluated through the complimentary approaches of non-empirical evaluation and empirical evaluation.

7. Given a society of agents that possess dialectical game playing capabilities, can the agents being used to actually tackle certain problems as opposed to be used solely as a means to structure communicative acts? Chapter 6 demonstrated that arguing agents can be applied to distributed graph-colouring problems and that dialectical games have a wider range of possible roles to play in MAS than merely regulating communicative interaction.

In summary, the A4A provides a framework for designing, developing, implementing, and deploying dialectical games. MASADA has shown how agents that are constructed to engage in argumentation require domain specific knowledge and reasoning capabilities in addition to a representation of a dialectical game. The *i*-Xchange project demonstrated how intelligent agents can be constructed that incorporate multiple capabilities and multiple agent frameworks whilst maintaining an integrated cohesive design for the individual agents. The value of evaluation was demonstrated and the need both to examine the rules of dialectical games as well as the dialogues that are constructed according to them. As well as evaluating dialectical games within the graph-colouring knowledge domain, distributed graph-colouring was investigated as a target problem which dialectical games can usefully tackle. Overall it has been shown that dialectical games have an important role in the construction of effective and robust arguing agents and that dialectical games are not solely a way to regulate dialogical interactions but are an important tool that should be included in the arsenal of researchers who are investigating distributed problems.

Appendix A

Document Type Definitions

DUE to the adoption of XML to represent both input and output files to the MASADA and A4A components a number of Document Type Definition (DTD) files are required to ensure that correctly formed XML files are created.

A.1 Argument Template DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT templates (template+,pref_order*)>
<!ELEMENT template (conclusion+,premises+)>
<!ELEMENT conclusion (pred+,term*)>
<!ELEMENT premises (premise*)>
<!ELEMENT premise (pred+,term*)>
<!ELEMENT pred (#PCDATA)>
<!ELEMENT term (#PCDATA)>
<!ELEMENT rebuts (arg*)>
<!ELEMENT arg (pred+,term*)>
<!ATTLIST   arg
            id          CDATA #IMPLIED
            level       CDATA #IMPLIED
>
```

A.2 Colloqui Prefence DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT colloqui (domain?)>
<!ELEMENT domain (scenario?>
<!ELEMENT scenario (initfile?)>
```

```
<!ELEMENT initfile (name?,path?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ATTLIST   domain
            name        CDATA #IMPLIED
            instance    CDATA #IMPLIED
>
<!ATTLIST   scenario
            name        CDATA #IMPLIED
>
```

Bibliography

[1] R. J. Ackermann. *Belief and Knowledge*. Anchor Books, 1972.

[2] L. Amgoud and S. Parsons. Agent dialogues with conflicting preferences. In *ATAL 2001*, pages 190–205, 2004.

[3] I Angelelli. The techniques of disputation in the history of logic. *The Journal of Philosophy*, 67(20):800–815, 1970.

[4] K. Appel and W. Haken. Every planar map is four colorable. part 1. *Math*, 21:429–490, 1977.

[5] K. Appel, W. Haken, and J. Koch. Every planar map is four colorable. part 2. *Math*, 21:491–567, 1977.

[6] K. Atkinson. *What Should We Do? Computational Representation of Persuasive Argument in Practical Reasoning*. PhD thesis, University of Liverpool, 2005.

[7] K. Atkinson, T. Bench-Capon, and P. McBurney. Parmenides: Facilitating democratic debate. In *Lecture Notes in Computer Science*, pages 313–316. Springer Berlin / Heidelberg, 2004.

[8] J. L Austin. *How To Do Things With Words*. Oxford University Press, 1962.

[9] E. M. Avedon and B. Sutton-Smith. *The Study of Games*. John Wiley and Sons, Inc., first edition, 1971.

[10] M. Baker. The roles of models in artificial intelligence and education research: A prospective view. *International Journal of Artificial Intelligence in Education*, 11:122–143, 2000.

[11] J. A. Barker. The fallacy of begging the question. *Dialogue*, XV:241–255, 1976.

[12] E. M. Barth and E. C. W. Krabbe. *From Axiom To Dialogue*. Walter de Gruyter & Co., 1982.

[13] T. J. M. Bench-Capon. Specification and implementation of toulmin dialogue game. In *Proceedings of JURIX 98*, pages 5–20, 1998.

[14] T. J. M. Bench-Capon, T. Geldard, and P. H Leng. A method for the computational modelling of dialectical argument with dialogue games. *Artificial Intelligence and Law*, 8:223–254, 2000.

[15] E. Black. *A generative framework for argumentation-based inquiry dialogues*. PhD thesis, Department of Computer Science, University College London, 2006.

[16] R. J. Brachman and H. J. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, 2004.

[17] A. Cayley. Open problem. *Proceedings of the London Mathematical Society*, 9:148, 1878.

[18] C. I. Chesnevar, A. G. Maguitman, and R. P. Loui. Logical models of argument. *ACM Computing Surveys*, 32(4):337–383, dec 2000.

[19] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990.

[20] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. The MIT Press, 1990.

[21] P. R. Cohen and R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[22] L. Cummings. Formal dialectic in fallacy inquiry. *Argumentation*, 17, 2003.

[23] N. Dahlback. Towards a dialogue taxonomy. *Lecture Notes in Coputer Science*, 1236:29–40, 1996.

[24] F. P. M. Dignum, B. Dunin-Keplicz, and R. Verbrugge. Agent theory for team formation by dialogue. *Seventh International Workshop on Agent Theories, Architectures and Languages (ATAL 2000)*, pages 141–156, 2000.

[25] F. P. M. Dignum, B. Dunin-Keplicz, and R. Verbrugge. Agent theory for team formation by dialogue. *Intelligent Agents VII, Agent Theories, Architectures and Languages*, pages 150–166, 2001.

[26] F. P. M. Dignum, B. Dunin-Keplicz, and R. Verbrugge. Creating collective intention through dialogue. *Logic Journal of the IGPL*, 9(2):305–319, 2001.

[27] F. P. M. Dignum and G. A. W. Vreeswijk. Towards a testbed for multi-party dialogues. In *Workshop On Agent Communication Languages*, pages 212–230, 2003.

[28] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and *n*-person games. *Artificial Intelligence*, 77:321–357, 1995.

[29] C. Dutilh Novaes. Medieval logic as logical games of consistency maintenance. *Synthese*, 145(3):371–395, 2005.

[30] F. H. van Eemeren, R. Grootendorst, and F. Snoeck Henkemans. *Fundamentals Of Argumentation Theory*. Lawrence Erlbaum Associates, 1996.

[31] R. Elio and F. J. Pelletier. Human benchmarks on ai's benchmark problems. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 406–411, 1993.

[32] M. Elvang-Goransson, J. Fox, and P. Krause. Dialectic reasoning with inconsistent information. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 114–121, 1993.

[33] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, D. McKay, R. Fritzson, S. Shapiro, R. Pelavin, and J. McGuire. Specification of the KQML Agent-Communication Language – plus example agent policies and architectures, 1993.

[34] Foundation for Intelligent Physicial Agents. Fipa communicative act library specification available from http://www.fipa.org, 2001.

[35] Java Agent DEvelopment Framework. http://jade.tilab.com/, 2006.

[36] S. Franchi. Chess, games, and flies. *Essays in Philosophy*, 6(1), 1995.

[37] M. R. Genesereth and R. E. Fikes. Knowledge interchange format, vesion 3.0 reference manual. Technical Report logic-92-1, Computer Science Department, Stanford University, 1992.

[38] Michael A. Gilbert. *How To Win An Argument*. John Wiley & Sons, second edition, 1996.

[39] R. A. Girle. Dialogue and entrenchment. In *Proceedings Of The 6th Florida Artificial Intelligence Research Symposium*, pages 185–189, 1993.

[40] R. A. Girle. Knowledge organized and disorganized. *Proceedings of the 7th Florida Artificial Intelligence Research Symposium*, pages 198–203, 1994.

[41] R. A. Girle. Commands in dialogue logic. *Practical Reasoning: International Conference on Formal and Applied Practical Reasoning, Springer Lecture Notes in AI*, 1085:246–260, 1996.

[42] Agent Oriented Software Group. http://www.agent-software.com, 2006.

[43] S. M. Grunvogel. Formal models and game design. *Game Studies*, 5(1), 2004.

[44] C. L. Hamblin. *Fallacies*. Methuen and Co. Ltd, 1970.

[45] J. Haugeland. Semantic engines: An introduction to mind design. In J. Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, pages 1–34. MIT Press, Cambridge, MA, 1981.

[46] J. Hintikka. Language games for quantifiers. In *Studies in Logical Theory*, number 2 in American Philosophical Quarterly Monograph Series, pages 46–76. Blackwell, 1968.

[47] D. Hitchcock, P. McBurney, and S. Parsons. A framework for deliberation dialogues. In *Proceedings of the Fourth Biennial Conference of the Ontario Society for the Study of Argumentation*, 2001.

[48] Carnegie Melon University Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.

[49] H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *Proceedings of the 7th International Conference on Artificial Intelligence and Law*, 1999.

[50] M. W. Johnson, P. McBurney, and S. Parsons. When are two protocols the same? In *Communication In Multiagent Systems*, volume 2650 of *Lecture Notes In Computer Science*. Springer, 2003.

[51] R. M Johnson. *A Logic Book*. Thomson Wadsworth, 2007.

[52] D. Kalofonos, N. Karunatillake, N. R. Jennings, T. J. Norman, C. Reed, and S. Wells. Building agents that plan and argue in a social context. In P. E. Dunne and T. J. M Bench-Capon, editors, *Computational Models of Argument*, pages 15–26. IOS Press, 2006.

[53] D. Kalofonos and T. J. Norman. Exploiting abstraction for multi-agent planning. In *Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, pages 88 – 93, Cork, 2004.

[54] D. Kalofonos and T. J. Norman. An investigation into team-based planning. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 5590 – 5595, The Hague, 2004.

[55] N. C. Karunatillake and N. R. Jennings. Is it worth arguing? In *Argumentation in Multi-Agent Systems*, LNAI 3366, pages 234–250, NY, USA, 2004.

[56] J. Katzav and C. Reed. On argumentation schemes and the natural classification of arguments. *Argumentation*, 18(2):239–2259, 2004.

[57] B. W. Kernighan and D. M. Ritchie. *The C Programming Language (2nd Edition)*. Prentice Hall, 1988.

[58] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

[59] E. C. W. Krabbe. Profiles of dialogue. In J. Gerbrandy, M. Marx, M. de Rijke, and Y. Venema, editors, *JFAK - Essays Dedicated to Johan van Benthem on the occasion of his 50th birthday*. Amsterdam University Press, 1999.

[60] E.C.W. Krabbe. Meeting in the house of callias: Rhetoric and dialectic. *Argumentation*, 14(3):205–217, 2000.

[61] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: A logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69, 1998.

[62] Y. K Labrou and T. Finin. History, State of the Art and Challenges for Agent Communication Languages. *Informatik/Informatique*, 2000.

[63] Y. K Labrou, T. Finin, and Y. Peng. Agent Communication Languages: The Current Landscape. *Intelligent Agents*, 14(2), 1999.

[64] V. Lifschitz. 25 benchmark problems in nonmonotonic reasoning. In *Nonmonotonic Reasoning*, pages 202–219. Springer, Berlin, 1989.

[65] A. R. Lodder. *DiaLaw*. Law And Philosophy Library. Kluwer Academic Publishers, 1999.

[66] A. R. Lodder and A. Herczog. Dialaw: A dialogical framework for modeling legal reasoning. In *International Conference on Artificial Intelligence and Law*, pages 146–155, 1995.

[67] Calico Jack Ltd. http://www.calicojack.co.uk, 2005.

[68] R. D. Luce and H. Raiffa. *Games and decisions: Introduction and critical survey*. Wiley, 1965.

[69] J. D. Mackenzie. How to stop talking to tortoises. *Notre Dame Journal Of Formal Logic*, XX(4), 1979.

[70] J. D. Mackenzie. Question begging in non-cumulative systems. *Journal Of Philosophical Logic*, 8:117–133, 1979.

[71] N Maudet and F. Evrard. A generic framework for dialogue game implementation. In *Proceedings of the Second Workshop on Formal Semantics and Pragmatics of Dialog*, 1998.

[72] P. McBurney, D. Hitchcock, and S. Parsons. The eightfold way of deliberation dialogue. *International Journal of Intelligent Systems*, 2002.

[73] P. McBurney and S. Parsons. Agent ludens: Games for agent dialogues. In *Game-Theoretic and Decision-Theoretic Agents (GTDT 2001): Proceedings of the 2001 AAAI Spring Symposium*, 2001.

[74] P. McBurney and S. Parsons. Chance discovery using dialectical argumentation. In *New Frontiers in Artificial Intelligence: Joint JSAI 2001 Workshop Post Proceedings, Lecture Notes in Artificial Intelligence vol. 2253*, pages 414–424. Springer-Verlag, 2001.

[75] P McBurney and S Parsons. Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence*, 32(1–4):125–169, 2001.

[76] P. McBurney and S. Parsons. Dialogue games in multi-agent systems. *Informal Logic*, 22(3):257–274, 2002.

[77] P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11(3):315–334, 2002.

[78] P. McBurney and S. Parsons. The posit spaces protocol for multi-agent negotiation. *Advances in Agent Communication: International Workshop on Agent Communication Languages, ACL 2003*, 2004.

[79] P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. *Proceedings of the First AAMAS*, pages 402–409, 2002.

[80] P. McBurney, R. M. van Eijk, S. Parsons, and L. Amgoud. A dialogue-game protocol for agent purchase negotiations. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(3):235–273, 2003.

[81] Peter McBurney, Rogier M. van Eijk, Simon Parsons, and Leila Amgoud. A dialogue-game protocol for agent purchase negotiations. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(3):235–273, 2001.

[82] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[83] J. McCarthy. AI as Sport. *Science*, 276(5318):1518–1519, 1997.

[84] J. McCarthy. Elaboration tolerance, 1998.

[85] M. L. Minsky. A framework for representing knowledge. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, 1975.

[86] D. Moore and D. Hobbes. Computational uses of philosophical dialogue theories. *Informal Logic*, 18(2 and 3):131–163, 1996.

[87] T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. In *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 39–48. ACM Press, 2005.

[88] J. F. Nash. Two-person cooperative games. *Econometrica*, 21, 1953.

[89] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, 1944.

[90] A. Newell, J. Shaw, and H. Simon. Chess playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2:320–335, 1977.

[91] D. J. O'Keefe. Two concepts of argument. *The Journal of the American Forensic Association*, 13(3):121–128, 1977.

[92] N. Oren, T. J. Norman, and A. Preece. Loose lips sink ships: a heuristic for argumentation. In *Third International Workshop on Argumentation in Multi-Agent Systems (ARGMAS'06)*, 2006.

[93] N. Oren, T. J. Norman, and A. Preece. A utility and information based heuristic for argumentation. In *Proceedings of the 6th Workshop on Computational Models of Natural Argument (CMNA'06)*, 2006.

[94] S. Parsons and N. R. Jennings. Negotiation through argumentation. In *Proceedings of ICMAS'96*, pages 267–274, 1996.

[95] S. Parsons, C. Sierra, and N. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[96] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation*, 13(3):347–376, 2003.

[97] C. Perelman and L. Olbrechts-Tyteca. *The New Rhetoric*. University of Notre Dame Press, 1969.

[98] H. Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation*, forthcoming, 2005.

[99] H. Prakken. Combining sceptical epistemic reasoning with credulous practical reasoning. In P. E. Dunne and T. J. M Bench-Capon, editors, *Computational Models of Argument*, pages 311–322. IOS Press, 2006.

[100] SWI Prolog. http://www.swi-prolog.org/, 2006.

[101] I. Rahwan. *Interest-based Negotiation in Multi-Agent Systems*. PhD thesis, University of Melbourne, 2004.

[102] I. Rahwan, S. D. Ramchurn, N. R. Jennings, S. McBurney, P. Parsons, and L. Sonenberg. Argumentation-based negotiation. *Knowledge Engineering Review*, 18(4):343–375, 2003.

[103] C. Reed. Dialogue frames in agent communication. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, pages 246–253. IEEE Press, 1998.

[104] C. Reed and G. Rowe. Araucaria: Software for puzzles in argument diagramming and xml. Technical report, University Of Dundee, 2001.

[105] C. Reed and D. Walton. Towards a formal and implemented model of argumentation schemes in agent communication. In I. Rahwan, P. Moraitis, and C. Reed, editors, *First International Workshop on Argumentation in Multi-Agent Systems*, 2004.

[106] C. Reed and D. N. Walton. Applications of argumentation schemes. In *Proceedings of the 4th Conference of the Ontario Society for the Study of Argument (OSSA2001)*, 2001.

[107] C. Reed and D. N. Walton. Argumentation schemes in argument-as-process and argument-as-product. In *Proceedings of the Conference Celebrating Informal Logic @ 25*, 2003.

[108] N. Rescher. *Dialectics*. State University of New York Press, Albany, 1977.

[109] R. Robinson. *Plato's Earlier Dialectic (2nd Edition)*. Clarendon Press, Oxford, 1953.

[110] S. Russell and P Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2003.

[111] K. Salen and E. Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2004.

[112] K. Salen and E. Zimmerman. *The Game Design Reader: A Rules of Play Anthology*. The MIT Press, 2006.

[113] J. Schaeffer. The games computers (and people) play. In M. Zelkowitz, editor, *Advances in Computers 50*, pages 189–266. Academic Press, 2000.

[114] J. Schaeffer. A gamut of games. *AI Magazine*, 22(3):29–46, 2001.

[115] J. R. Searle. *Speech Acts*. Cambridge University Press, 1969.

[116] J. R. Searle and D. Vanderveken. *Foundations of Illocutionary Logic*. Cambridge University Press, 1985.

[117] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(314):256–275, 1950.

[118] Y. Shoham. Agent-oriented programming. Technical Report STAN-CS-1335-90, Computer Science Department, Stanford University, 1990.

[119] H. Simon and J. Schaeffer. The game of chess. In *Handbook of Game Theory with Economic Applications*, pages 1–17. Elsevier Science, 1992.

[120] R. A. Sorensen. Unbeggable questions. *Analysis*, 56(1):51–55, 1996.

[121] P. V. Spade. Recent research on medieval logic. *Synthese*, 40:3–18, 1979.

[122] B. Stroustrup. *The C++ Programming Language (Special Edition)*. Addison Wesley, 2000.

[123] B. Suits. *The Grasshopper: Games, Life, and Utopia*. University of Toronto Press, 1978.

[124] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[125] K. Sycara. Resolving goal conflicts via negotiation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[126] K. Sycara. Persuasive argumentation in negotiation. *Theory And Decision*, 28:203–242, 1990.

[127] K. Sycara. Multiagent systems. *Artificial Intelligence*, 2(19):79–92, 1998.

[128] P. Tabuada, G. Pappas, and P. Lima. Composing abstractions of hybrid systems. *Lecture Notes in Computer Science*, 2289:436–450, 2002.

[129] S. Toulmin. *The Uses Of Argument*. Cambridge University Press, 1958.

[130] A. Turing. Digital computers applied to games. In *Faster Than Thought*, pages 286–295. Pitman, 1953.

[131] G. A. W. Vreeswijk. Interpolation of benchmark problems in defeasible reasoning. In *Proceedings of 2nd World Conference on the Fundamentals in AI (WOCFAI'95)*, pages 453–468, 1995.

[132] D. Walton. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, 1996.

[133] D. N. Walton. *Logical Dialogue-Games And Fallacies*. University Press Of America, 1984.

[134] D. N. Walton. Types of dialogue, dialectical shifts and fallacies. In *Argumentation Illuminated*, pages 133–147, 1992.

[135] D. N. Walton. *Argument Structure: A Pragmatic Theory*. University of Toronto Press Incorporated, 1996.

[136] D. N. Walton. Ethotic arguments and fallacies: The credibility function in multi-agent dialogue systems. *Pragmatics & Cognition*, 7(1):177–203, 1999.

[137] D. N. Walton. *Fundamentals of Critical Argumentation*. Cambridge University Press, 2006.

[138] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue*. SUNY series in Logic and Language. State University of New York Press, 1995.

[139] S. Wells. Cumulativeness in dialectical games. In *6th International Conference on Argumentation*, 2005.

[140] S. Wells and C. Reed. Knowing when to bargain: The roles of negotiation and persuasion in dialogue. In F. Grasso, R. Kibble, and C. Reed, editors, *Sixth Workshop on Computational Models of Natural Argument*, 2005.

[141] S. Wells and C. Reed. Knowing when to bargain. In P. E. Dunne and T. J. M Bench-Capon, editors, *Computational Models of Argument*, pages 235–246. IOS Press, 2006.

[142] T. Winograd. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972.

[143] J. Woods and D. N. Walton. Petito principii. *Synthese*, 31:107–128, 1977.

[144] J. Woods and D. N. Walton. Arresting circles in formal dialogues. *Journal Of Philosophical Logic*, 7:73–90, 1978.

[145] J. Woods and D. N. Walton. Question-begging and cumulativeness in dialectical games. *Nous*, 16:585–605, 1982.

[146] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, Ltd, 2002.

[147] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practise. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[148] T. Yuan. *Human Computer Debate, A Computational Dialectics Approach*. PhD thesis, Leeds Metropolitan University, 2004.

[149] T. Yuan, D. Moore, and A. Grierson. A conversational agent system as a test-bed to study the philosophical model DC. In *Proceedings of the 3rd Workshop on Computational Models of Natural Argument (CMNA'03)*, 2003.