A Computer Game for Abstract Argumentation

Tangming Yuan, Viðar Svansson
Department of Computer Science, Faculty of Business and Science
University of Akureyri, Akureyri 600, Iceland
yuan@unak.is, vidar@teikn.is

David Moore, **Alec Grierson**
School of Computing, Leeds Metropolitan University,
Leeds LS6 3QS, United Kingdom
{d.moore, a.j.grierson}@leedsmet.ac.uk

Abstract

This paper reports our work concerning the development of a computer game for abstract argumentation. In particular, we discuss the dialogue protocol we have used to regulate players in making moves. We also discuss proposed strategies for a software agent to act as a game player. The computer game has been fully implemented, and enables human versus human, agent versus agent and human versus agent playing of the game. A user based evaluation has been undertaken, and suggests that the game is both challenging and entertaining and is easy to learn. It is anticipated that this work will contribute to the development of argumentative agents and of computer game based educational argument, and help to illuminate research issues in the field of argumentation systems.

1 Introduction

Much work in computational dialectics concerns exchange of concrete arguments, e.g. [Bench-Capon, 1998; Grasso et al., 2000; Yuan, 2004]. This paper however outlines our work in using abstract species of argument [Dung, 1995] to construct a computer game to enable human-human, agent-agent and human-agent interaction. The game is expected to be entertaining and at the same time to be used to educational advantage - to develop students' planning skills. The remainder of this paper is organised as follows. The concept of an abstract argumentation system is briefly introduced in section 2. The discussion of the dialogue protocol used in our implementation is presented in section 3. Section 4 provides the technical details of our computational implementation of the game. The strategy for a software agent to act as a worthy opponent is presented in section 5 and details of the user evaluation are discussed in section 6. The final section draws the conclusions and discusses our intended future work concerning the development of the abstract argumentation game.

2 Abstract Argumentation System

An abstract argumentation system A is defined in [Dung, 1995; Vreeswijk and Prakken, 2000; Wooldridge, 2002] as a pair

$$A = \langle X, \leftarrow \rangle,$$

where X is a set of arguments, and $\leftarrow$ is an attacking relation between pairs of arguments in X. The expression $a \leftarrow b$ is pronounced as "a is attacked by b" or "b is the attacker of a".

An example of an abstract argumentation system can be seen in figure 1, which represents the pair $A = \langle X, \leftarrow \rangle$ with arguments

$$X = \{a, b, c, f, g, j, k, o, p, q, t, v, y\}$$

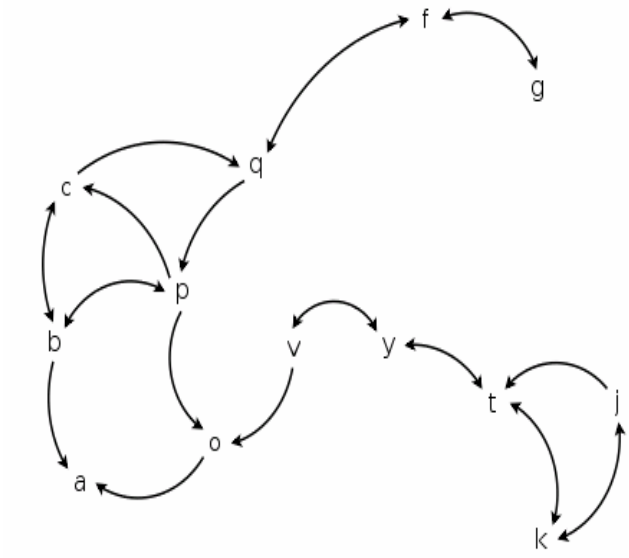

Figure 1 An example of abstract argumentation system

The abstract argumentation system has a number of interesting properties for computational utilisation. Firstly, it

concerns the attacking relation between arguments only; it does not concern the internal structure of each argument (ie its premises and conclusion) or where the attacking relations come from. Players operationalising an abstract argumentation system do not need to argue about the validity of the attacking relations between different arguments (e.g. argument *b* attacks argument *a* in figure 1). Secondly, the abstract system is represented as a directed graph, which enables the user to view the complete system and to select an argument directly from the graph. It therefore avoids the difficulty of substantive user input, as faced by systems such as those of Bench-Capon [1998] and Yuan [2004], who adopt a menu-based approach, of Grasso et al. [2000] who use first order predicates and of Ravenscroft and Pilkington [2000] who use rhetorical predicates. Given the above arguments, the abstract argumentation system is adopted in the computer game that will be described in the coming sections.

3 The Argument Game

There must be some rules regulating the players to ensure fair play. The rules should be simple so that a human user can easily adopt them and quickly develop his/her winning strategies. Different argument games have been developed in the area of computational dialectics, e.g. [Vreeswijk and Prakken, 2000; Wooldridge, 2002; Dunne and Bench-Capon, 2003]. We adopt the game presented in [Wooldridge, 2002: 153-154] for reasons of simplicity. The argument game is formalised as a quadruple

$$G = \langle A, D, R, P \rangle,$$

where A is the argumentation system (e.g. the one outlined in section 2), D is the dialogue history $\langle \alpha_0, \alpha_1, \alpha_{2,\ldots} \alpha_n \rangle$ which contains a set of moves made by game participants, R is a set of rules regulating players to make a move, and P is a pair of players $\{0, 1\}$.

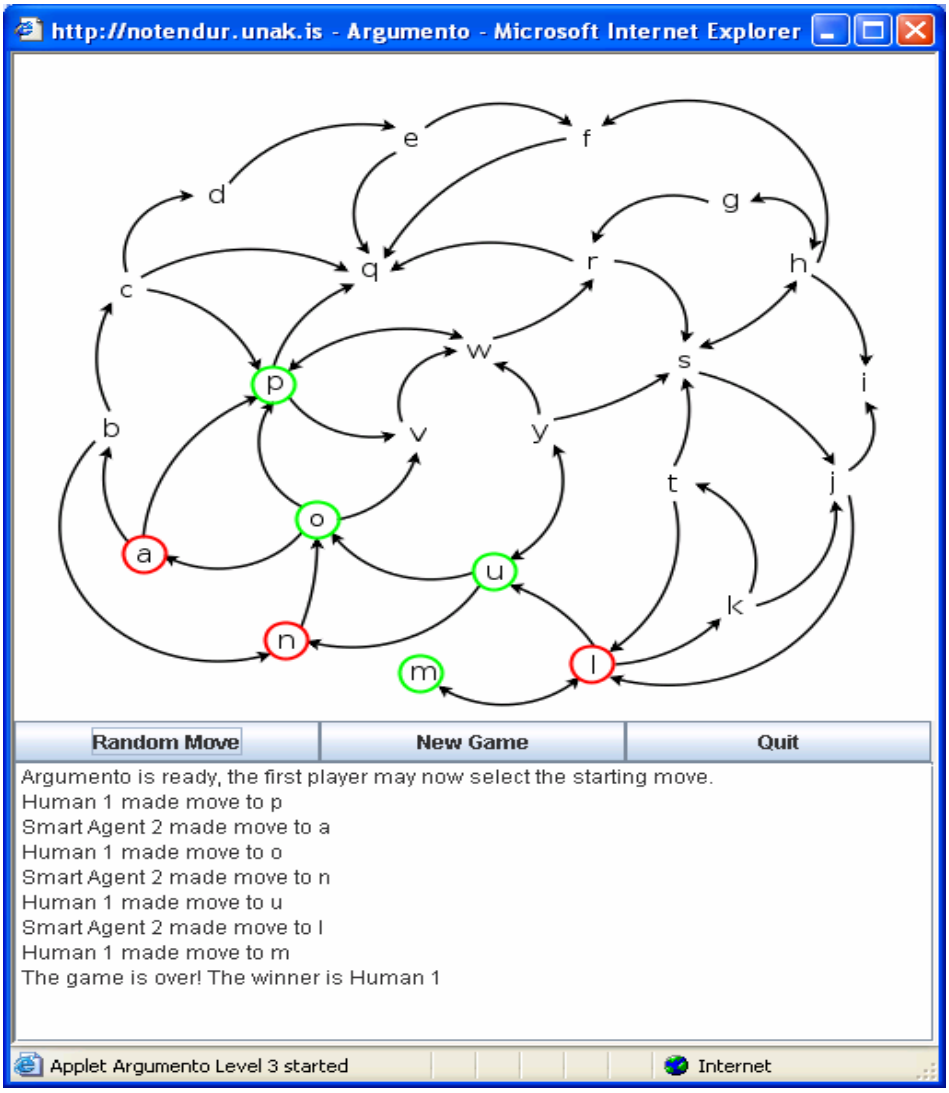

Figure 2 An example of the system interface

The set R contains six rules:

1. First move in D is made by P_0.
$$P_0 = 0$$
2. Players take turns making moves.
$$P_i = P_{i \bmod 2}$$
3. Players cannot repeat a move.
$$\forall \alpha_i, \alpha_j \in D,\ \alpha_i \neq \alpha_j$$
4. Each move has to attack (defeat) the previous move
$$\alpha_i \rightarrow \alpha_{i-1}$$
5. The game is ended if no further moves are possible
$$\forall \alpha_i \in A \wedge \notin D, \alpha_i \not\rightarrow \alpha_n$$
6. The winner of the game is the player that makes the final move.
$$G_{winner} = P_{n \bmod 2}$$

4 The Computational Implementation

The approach, then, is to use the argument game outlined above as the basis for a computer argumentation game. A fully functional system has been built, using the XML and Java programming languages, and deployed on the internet (*http://notendur.unak.is/not/yuan/game/index.php*). The system is designed to have three levels: level 1, level 2 and level 3, according to the complexity of the argumentation system. The level 1 argumentation system contains 6 arguments, level 2 13 arguments, and level 3 24 arguments. A user can select his/her preferred level to play the game. In addition, the system is designed to enable the user to select his/her opponent. There are three choices for this: another human player, a random agent or a smart agent. A random agent is the one making a move by randomly picking up a legally available argument. A smart agent has been given strategies to select the best possible arguments in order to win the game. Rather than being a game player, the user can also set up two software agents and observe them playing the game.

An example game involving a user playing with a software agent can be seen in figure 2. The user made the first move *p*, the agent made the second move *a* though it had other available options *c*, *w*, *o*, in attacking *p*. In the third turn, the user made the only available move *o* in attacking *a*. The game continues until the user made the argument *m*. The software agent in this situation cannot locate any further arguments attacking *m*, the system therefore proclaims the user the winner.

The system architecture is shown in figure 3. There are five main units of the system: *visualisation*, *control*, *knowledge base*, *environment* and *agent* units. The *visualisation* unit provides a user interface for the user to interact with the system. An example user interface is shown in Figure 2. The top panel of the interface displays the argumentation system as a directed graph where letters represent abstract arguments and the arcs represent the attacking relations. The graphs are created by using Adobe Photoshop and saved as .png image files. When a new game starts, the appropriate

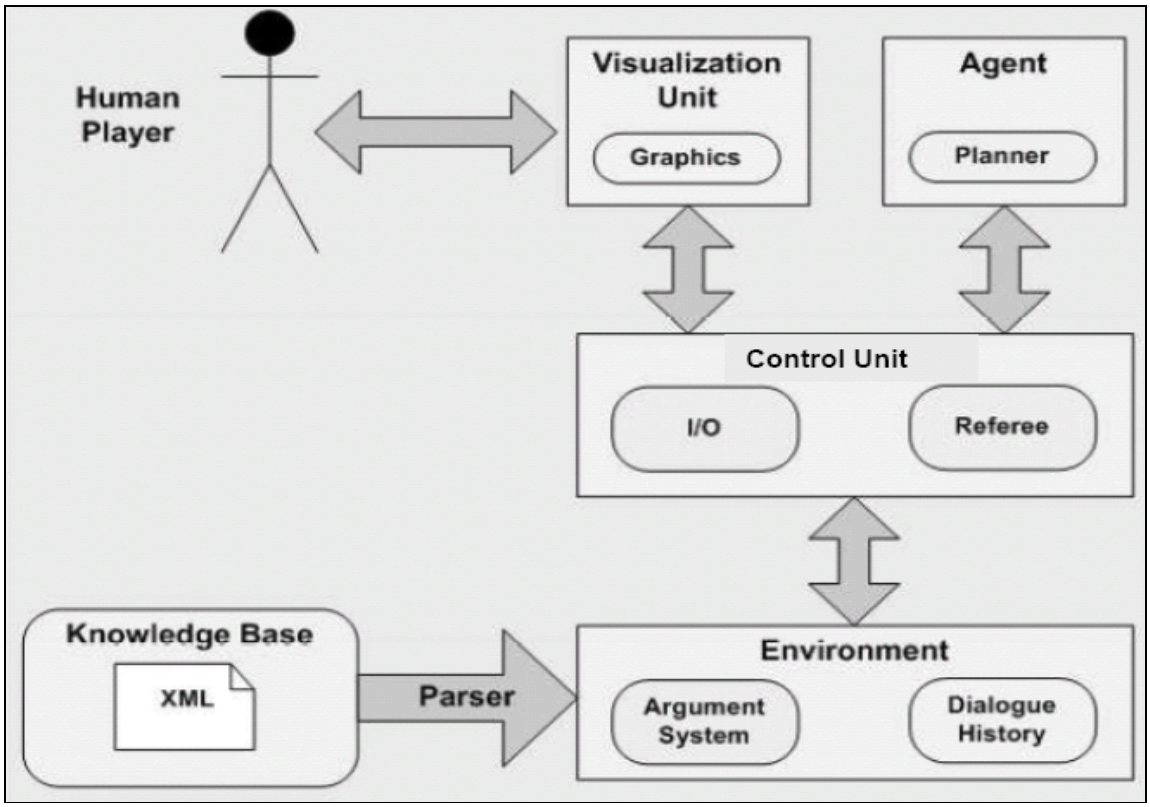

Figure 3 System architecture

graphical image is loaded depending on the level of complexity selected by the user. To make a move, the user needs simply to point the mouse to the target argument and click on it. The arguments made by P_0 are highlighted with green circles, and by P_1 red circles. (Current work involves an alternative arrangement usable by colour blind people.) The dialogue history is displayed at the bottom panel of the interface. The middle panel of the interface contains three buttons: New Game, Quit and Random Move. The former two are self-explanatory. The Random Move button is designed for the user when he/she cannot make up his/her mind on what move to make. The system will make a random legal move on behalf of the user when the button is pushed.

The *control unit* is designed to have an input manager, an output manager and a referee. The input manager waits for moves made by the agent and user, and then passes the move to the referee for validation. If the move is valid, then the dialogue history is updated, otherwise the move maker needs to redo the move. The output manager is responsible for displaying the newly updated dialogue history and relevant dialogue instructions on the user interface.

The *knowledge base* unit contains the XML files which are identical to each of the graphical image files. Each argument is represented as a XML tag, e.g. <param value= "*a*, 59, 307, *b*, *p*" name= "argument0">. The *name* field specifies that this is an argument followed by a numerical identifier to distinguish it from other arguments. The *value* field specifies parameters for this argument: the first is the abstract name of this argument, the second and the third indicate the X and Y-coordinates of the argument positioning on the graphical image. The remaining parameters indicate all the arguments attacked by this argument. The example argument says that the argument *a* is positioned at (59, 307), and the arguments b and p are attacked by a. It is important to encode an argumentation system using XML, because it is much easier for the Java program to process XML files than to process image files. This also has the advantage of allowing non-Java project members to construct argumentation systems using Adobe Photoshop and XML.

The *environment* unit contains an argumentation system *A* and a dialogue history *D*. When a new game starts, the corresponding XML file is loaded and parsed, as result of which the complete binary attacking structure of the argumentation system is encoded into a two dimensional boolean array, where $A_{\alpha i, \alpha j}$ is true if $\alpha_i \rightarrow \alpha_j$. Figure 4 shows an example argumentation system "$c \rightarrow b \rightarrow a$" encoded as a 2D boolean array, where 1 refers to true and 0 refers to false.

A	a	b	c
a	0	0	0
b	1	0	0
c	0	1	0

Table 1 An example argumentation system encoded as a 2D Boolean array

The dialogue history *D* is structured as a *stack* of coloured arguments. The colours are consistent with their colours displayed on the graphical user interface. The top of the *stack* points to the arguments last made. The dialogue history is dynamic while the argumentation system remains static once loaded.

The *agent* unit is used when a software agent is one (or both) of the game players. It has its own lifecycle control separated from the system main thread. The planner of the agent has been given necessary strategies to enable the agent to be a worthy game opponent. The agent strategy will be discussed in the next section.

5 Agent Strategy

Several dialogue strategies for computational dialectic systems have been proposed by different authors. Yuan [2004], for example, utilises Moore's [1993] three level decision making to enable a computer to participate in academic debate on a controversial issue. For dialogue types other than debate, other strategies may be appropriate. Grasso et al. [2000] adopt, for their nutritional advice-giving system, schemas derived from Perelman and Olbrechts-Tyteca's [1969] "New Rhetoric", and Ravenscroft and Pilkington [2000] utilise "a repertoire of legitimate tactics available for addressing common conceptual difficulties" (p283). Amgoud and Maudet [2002] suggest "meta-preferences", such as "choose the smallest argument", to drive the choice of move, Freeman and Farley (1996) delineate ordering heuristics as guidelines for selecting argument moves, and Oren et al. [2006] propose a heuristic for argumentation based on minimising the information revealed to other dialogue participants.

When designing strategies for computational agents competing with human users, there has been a genuine concern that a computer's superior memory when compared with human players may undermine the fairness of the game [Walton, 1984], in that users may be frustrated being constantly defeated. We have therefore proposed two levels of strategies for a software agent to act as game participant: a random strategy and a probability utility based strategy.

A random agent picks up an argument randomly from the set of legal arguments. The set of legal arguments is defined as follows:

$$\alpha \in A \wedge \alpha \rightarrow \text{top}[D] \wedge \alpha \notin D$$

where α is an element in the set, *A* is the argumentation system, *D* represents the stack of dialogue history, and top[*D*] is the last move in the dialogue history.

To compute the set of legally available moves, an algorithm needs to traverse *A* to collect all arguments attacking top[*D*], and then traverse the dialogue history *D* to make sure each of the collected arguments has not been used. The running time for the random strategy is $\boldsymbol{O}(n^2)$ in the worst case, where *n* is the number of arguments in the argumentation system.

A probability-utility based strategy has been proposed, which enables an agent to select a legal move with the highest probability of winning an abstract argumentation game. A probability-utility based agent first generates a dialogue tree *T* rooted at α_0 via the algorithm specified below:

Dialogue_Tree_Generator[1] *(A, α_0)*

```
1  for each α ∈ A
2      do π[α] <= NIL
3  Q <= ∅
4  ENQUEUE (Q, α0)
5  while Q ≠ ∅
6      do  u <= DEQUEUE (Q)
7          for each v→ u
8              do if v∉D[v]
9                  then  π[v] <= u
10                       ENQUEUE (Q, v)
```

where α_0 is the first argument made by its opponent, *A* represents the argumentation system, $\pi[\alpha]$ refers to the parent of α, *Q* is the standard queue data structure, ENQUEUE and DEQUEUE are queue operations, *D*[*v*] refers to the dialogue history up to the point of *v*, and <= refers to assignment.

A dialogue tree (such as that shown in figure 4) is generated by running the Dialogue_Tree_Generator(*A, p*), where *A* is the argumentation system in figure 1, *p* is the first move made by the user.

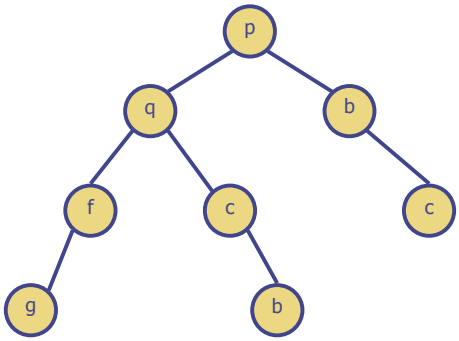

Figure 4 An example dialogue tree

Every path from the root down to a leaf is a possible dialogue sequence. Some sequences result in victory (utility=1) while some result in defeat (utility=0). Some branches contain more winning sequences, while some branches contain more defeat sequences. The agent is designed to select the move with the highest probability of winning.

The utility for each node in the dialogue tree T is then computed by using the algorithm below:

Probability_Utility (*T*, α)

```
1    P_α <= 0
2    if children[α] is empty
3      then P_α <= depth[α] mod 2
4      else for each β ∈ children[α]
5          do P_α <= P_α+ Probability-Utility (T, β)
6             P_α <= P_α/|childen[α]|
```

[1] We follow the pseudo-code convention of [Cormen et al., 2001], except that we use "<=" to refer to an assignment instead of a slashed arrow, because we are using a slahed arrow to refer to an attacking relation between arguments in this paper.

where P_α refers to the probability utility of node α, *children*[α] refers to set of children of node α and depth[α] refers to the depth of node α. The utility for a leaf is computed against its depth (line 2-3). The utility for an internal node is the sum of the utility value of its children (line 4-5), divided by the number of its children (line 6). The occurrence probabilities of its children are equal.

By using the Probability-Utility algorithm, the utility values for each node are computed as shown in figure 5 (values are after the /).

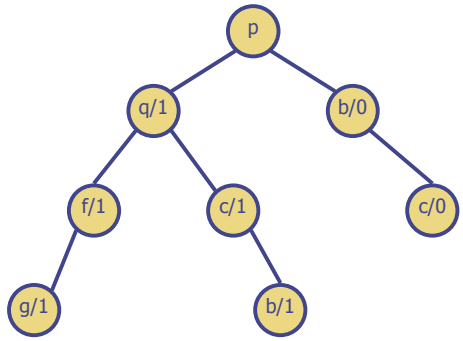

Figure 5 A dialogue tree with utility values

Given the utility value of each node, the agent would select the argument with the highest value from the set of legally available arguments {*q*, *b*} to response to user's move *p*; in this example, the agent will select *q* rather than *b*.

The running time for the Dialogue_Tree_Generator algorithm is ***O***(*r*n*), for the Probability_Utility algorithm ***O***(*r*), and in total the time cost for the probability utility strategy is ***O***(*r*n*) where r is the total number of attacking relations and n is the total number of arguments in the argumentation system. In the worst case where all arguments attack all the other arguments, the algorithm requires factorial running time ***O***(*n*!). In a realistic scenario, however, the number of attackers *k* for each argument is much less than *n,* and the approximate running time would be ***O***(*k!).*

6 Evaluation

An initial usability evaluation of the system has been conducted. The purpose of the evaluation was to assess the usability of the game and the proposed strategies. Five participants, 3 male and 2 female, were invited to take part in the evaluation: three Bsc Computer Science students, one high school student and one 10 year old primary school student. After a brief introduction to the game, they each played 12 different game setups:

Human vs. random agent, level 1
Human vs. random agent, level 2
Human vs. random agent, level 3
Human vs. intelligent agent, level 1
Human vs. intelligent agent, level 2
Human vs. intelligent agent, level 3
Random agent vs. human, level 1
Random agent vs. human, level 2
Random agent vs. human, level 3

Intelligent agent vs. human, level 1
Intelligent agent vs. human, level 2
Intelligent agent vs. human, level 3

The system kept a record of the results of each game. Each participant played several times for each game setup until they won. The user reactions were observed by the experimenter. After playing the game, participants were asked to fill in a questionnaire concerning their opinion of the game.

98 games were played, and participants won 60 times (62%). On level 1 argumentation system, participants won 53%. On level 2, participants won 69%. On level 3, participants won 61%. Intuitively, the fact that a user has a good chance of winning whilst not being guaranteed a victory, can be expected to encourage human users to play with the software agent. This alleviates a concern similar to that of Walton [1984], that a computer's superior memory when compared with human players may undermine the fairness of the game.

The random agent won 12 out of 42 games (29%). The intelligent agent won 26 out of 56 games (46%). These figures are in line with the user rating on how smart the agent is: the random agent received 3.2 out of 10, and the intelligent agent received 6.2. These figures provide evidence that the strategies proposed for the intelligent agent do perform much better than the random strategy.

Most participants adopted a search function similar to the agent's utility function in order to be victorious. They took considerable time travelling the dialogue tree (with finger pointing to the screen) before making a move. Male users enjoyed playing the game and said they felt that the game is entertaining. When asked whether playing the game can help to develop their planning skills, male players all gave positive responses. Participants even continued playing the game with each other after the experiment. They said they would like to play the game again were it available on the Internet. The young subject thought the random agent was more difficult to play with. His focus was on the argumentation system rather than the opponent. It took him several tries to win most of the scenarios, but he was clearly improving with practice. The female participants in the evaluation expressed that they did not really enjoy playing the game though they did well in the game.

There seems to be a very low learning curve for even a novice user to play the game. The "Random Move" was never used. Some participants were confused with the attacking relations in the beginning, e.g. whether $a \rightarrow b$ refers to *a* attacks *b* or *b* attacks *a*.

In sum, the proposed probability-utility strategy performs much better than the random strategy, and they both seem to encourage human users to play with a software agent. The game is both challenging and entertaining with a low learning curve.

7 Conclusion and Further Work

We have constructed a computer game for abstract argumentation. The game enables human-agent, human-human and agent-agent argumentation. An evaluation has been conducted which furnishes evidence of the usability of the system.

We believe that the work reported makes a valuable contribution to the fields of dialectics, of agent communication and of computer game based argumentation education. Concerning the first of these, we have proposed strategies to be utilised within the argument game. Further, because the computer system we have built can readily be adapted to function with a different dialogue protocol and/or a different set of strategies, it potentially provides people working in the field of dialectics with a test bed within which they can experiment with new models and new strategies they develop that deal with abstract Dung-style arguments.

The work also contributes to agent communication in general and argumentative agents in particular. The argumentation game we have developed enables two agents (human and/or computational) to exchange arguments, and this provides a basis for extending the game for use in argumentative agent systems. The current set of FIPA (Foundation for Intelligent Physical Agents) agent communication protocols (e.g. the contract net) are not flexible enough to cope with argumentation [Norman et al., 2004].

The work contributes to computer game based educational argument in that the game is both challenging and entertaining, and expected to be useful for enabling students to practice their planning ability. In particular, the system can potentially be used as an assistive tool for teachers who are teaching and for students who are studying argument games and abstract argumentation (e.g. as part of a Computer Science course in Agent and Multi-agent Systems).

There are several ways to carry this research forward. Current work involves refining the system in the light of evaluation feedback. We are also planning to investigate different strategies (e.g. min-max and alpha-beta pruning) for use in the current system, and subsequently enable them to compete with each other and then study the results.

We are also planning to enable the software agent to give some "hints" re a good move, and explain why they are seen as a good move. By doing this, the users can learn the strategies of playing the game. The current system has three levels and they were manually constructed. More levels of argumentation systems can be added to the system. It would be ideal if different levels of argumentation systems can be generated on the fly. Further evidence can then be collected concerning the usefulness of the game as a learning tool.

The system can also be expanded to enable hyperlinks from the abstract arguments to concrete arguments in some particular domain, and thus enable the dialogue participants to exchange concrete arguments as well as the abstract ones.

A further possible investigation is to extend the system for use in agent systems, e.g. by adding an additional function to calculate and display preferred extensions of argument systems.

References

[Amgoud and Maudet, 2002] Leila Amgoud and Nicolas Maudet. Strategical Considerations for Argumentative Agents. In Proceedings of the Ninth International Work-

shop on Non-Monotonic Reasoning (NMR'2002), Special Session on Argument, Dialogue, and Decision, Toulouse.

[Bench-Capon, 1998] Trevor J.M. Bench-Capon. Specification and Implementation of Toulmin Dialogue Game. In *Proceedings of JURIX 98*, GNI, Nijmegen, pp.5-20.

[Cormen et al., 2001] Thomas H. Cormen, Sharles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*, 2nd edition. The MIT express, Cambridge, Massachusetts, London, England.

[Dunne and Bench-Capon, 2003] Paul E. Dunne and Trevor J.M. Bench-Capon. Two Party Immediate Response Disputes: Properties and Efficiency. *Artificial Intelligence* 149(2):221-50, 2003.

[Dung, 1995] Phan M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77 (2): 321-357, 1995.

[Freeman and Farley, 1996] Kathleen Freeman and Arthur M. Farley. A Model of Argumentation and Its Application to Legal Reasoning. *Artificial Intelligence and Law,* 4: 163-197, 1996.

[Grasso et al., 2000] Floriana Grasso, Alison Cawsey and Ray Jones. Dialectical Argumentation to Solve Conflicts in Advice Giving: a Case Study in the Promotion of Healthy Nutrition. *International Journal of Human Computer Studies,* 53: 1077-1115, 2000.

[Moore, 1993] David Moore. *Dialogue Game Theory for Intelligent Tutoring Systems*. Unpublished Doctoral Dissertation, Leeds Metropolitan University, 1993.

[Norman et al., 2004] Tim J. Norman, Daniela V. Carbogim, Eric C. W Krabbe, and Douglas N. Walton. Argument and Multi-Agent Systems. In Chris. A. Reed and Tim. J. Norman (editors) *Argumentation Machines: New Frontiers in Argument and Computation*, volume 9 of *Argumentation Library*, pages 15-54, 2004. Kluwer Academic Publishers, Dordrecht.

[Oren et al. 2006] Nir Oren, Tim. J. Norman and Alun Preece. A Utility and Information Based Heuristic for Argumentation, In *Proceedings of the ECAI'2006 Workshop on Computational Models of Natural Argument* (CMNA06), Riva del Garda, Italy, August 2006.

[Pereman and Olbrechts-Tyteca, 1969]Chaim Perelman and Lucie Olbrechts-Tyteca. *The New Rhetoric: a Treatise on Argumentation*. Notre Dame Press, 1969.

[Ravenscroft and Pilkington, 2000] Andrew Ravenscroft and Rachael M. Pilkington. Investigate by Design: Dialogue Models to Support Reasoning and Conceptual Change. *International Journal of Artificial Intelligence in Education,* 11: 237-298, 2000.

[Vreeswijk and Prakken, 2000] Gerard A.W. Vreeswijk and Henry Prakken. Credulous and Sceptical Argument Games for Preferred Semantics. In M. Ojeda-Aciego, I.P. de Guzman, G. Brewka, & L. Moniz Pereria (Eds.), *Proceedings of JELIA'2000, The 7th European Workshop on Logic for Artificial Intelligence* pp. 239-253. Berlin: Springer Verlag.

[Walton, 1984] Douglas N. Walton. *Logical Dialogue Games and Fallacies*. University Press of America.

[Wooldridge, 2002]Mike Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, New York, NY, USA, 2002.

[Yuan, 2004] Tangming Yuan. *Human-Computer Debate, a Computational Dialectics Approach*. Unpublished Doctoral Dissertation, Leeds Metropolitan University, 2004.