Using Code Generation to Build a Platform for Developing and Testing Dialogue Games

Tangming YUAN [a,1], Suresh MANANDHAR [a] and Simon WELLS [b]
[a] *Department of Computer Science, University of York*
[b] *School of Computing, Edinburgh Napier University*

Abstract. Despite increasing research into their use as a vehicle for Human-Computer Dialogue and Inter-Agent Communication, Dialogue Games have not seen good uptake in industry. One of the reasons for this is the lack of methodologies and tooling for the development, evaluation, and exploitation of such systems. In this paper we build on the ProtOCL methodology to demonstrate the construction of a complete computational dialogue platform which supports the use of different dialogue games.

Keywords. Dialogue Games, Intelligent Agents, Dialogue Game Execution Platforms, Code Generation, Agent Dialogue

Introduction

Dialogue games (DG) are becoming increasingly popular tools for Human Computer Dialogue and Agent communication [3,6,5,17]. A number of computerized dialogue game systems have been developed. For example, [7] introduces a platform called MAgtALO that supports online debate using simple dialogue games. [1] presents a game using the PARMA protocol. [8] presents the "IACAS" system and [14] describes a number of debating systems for Human-Computer and Computer-Computer games. [9] performs a similar task but with a focus on comparison and evaluation between dialogue game rules.

However, whilst there is an increasing body of theoretical underpinning that demonstrates the value and utility of dialogue games, and also a range of novel implementations within specific problem domains, there remain very few methods and tools to support deployment of dialogue game based solutions within new problem domains. A drawback of most of these systems is the hard-wired nature of the protocol. Each system implements a particular game, and in some cases, where the rules are describe in natural language and are imprecise, a particular interpretation of a game. Hard-wiring makes reuse difficult, for example the DE implementation in one system cannot be easily saved and reused by another system. To address this, considerable research effort has therefore been devoted in the past decade in order to develop a generic dialogue game platform for use

[1] Corresponding Author: Department of Computer Science Deramore Lane, University of York, Heslington, York, YO10 5GH, UK ; E-mail: tommy.yuan@york.ac.uk

by dialogue game researchers and developers to construct and test their dialogue games and strategies in a convenient way. For example, [12] introduces the Dialogue Game Description Language (DGDL) and it's grammar for describing syntactically correct dialogue games and in [10] a number of games are collected which can all be described using DGDL. However there are currently no freely available run-times that will execute games described using DGDL.

In the remainder of this paper we describe a method and tools for developing *executable* computational dialectical systems. Section 1 introduces the architecture and the major functionalities of the platform. Section 2 discusses the ProtOCL method that we have developed to specify dialogue games and generate dialogue game component. Section 3 discusses individual agent design and outlines the development of a knowledge base creator that can be used to visually create argument knowledge base for agents. Finally, section 4 indicates directions for future work and concludes the paper.

1. The Dialogue Game Platform Overview

The aim of this platform is to demonstrate the integration of dialogue games, specified using ProtOCL, into a complete platform which can be used to engage in dialogue and to explore the efficacy of different game play techniques and strategies. The user interface is a conventional GUI design, illustrated in Figure 1. When a user initiates a new game, they can set parameters, for example, choosing game rules to play by, selecting agents and knowledge bases for the dialogue. Game rules and agents are implemented in Java, and knowledge bases are external XML files. In our example, the system has two agents, namely 'student' and 'agent1', who conduct a debate on the topic of capital punishment (CP). The dialogue is proceeds with one agent starting the dialogue by posing the question of whether CP is acceptable and the players assume opposing positions. The game proceeds with the players selecting from the available move types and then from the list of allowed propositional content. Of particular interest is the support for selecting the game to play, and that the supported games can be expanded by defining and loading new rules. Rules are defined using UML and OCL. Code is automatically generated from the definitions and incorporated into the dialogue game platform. In the following sections we shall review the ProtOCL method, the design of the agents used in the platform, and the knowledge base creation tool which provides the locutional content used in playing games on the platform.

2. The ProtOCL method

ProtOCL is an approach to specifying and implementing dialogue games using the Object Constraint Language (OCL), which has strong tool support and broad industry acceptance, to define game rules within the context of a wider Unified Modelling Language (UML) description of a generic, high-level dialogue game [19]. The UML model describes the generic core elements of dialogue games and provides the archetype for an API for generated code as well as a core game engine. Specific games are described in OCL and stub classes are generated which can interact with the engine via the API. End-developers must implement these stubs with game specific code to get a complete

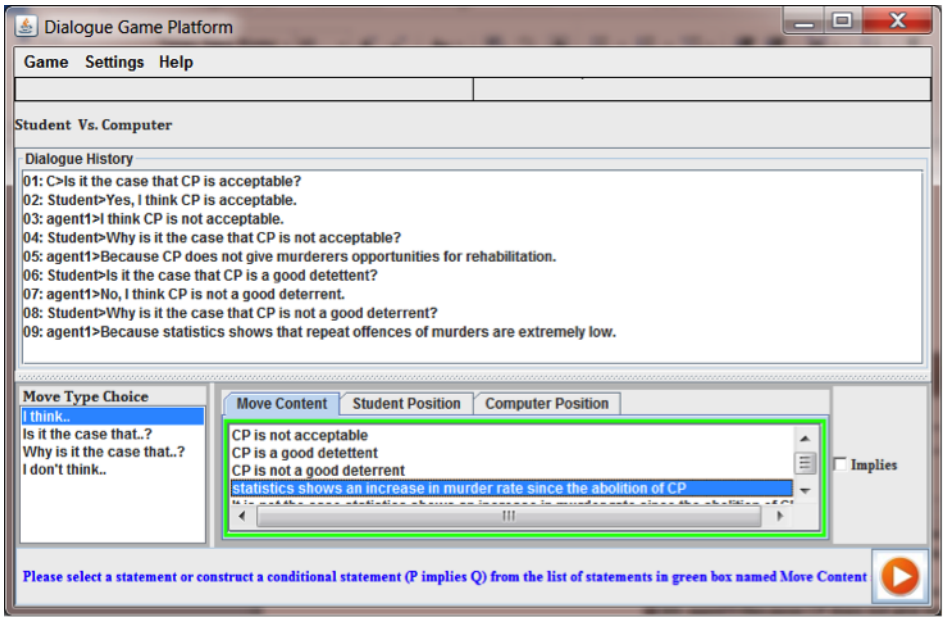

Figure 1. The dialogue game platform showing an ongoing human-agent dialogue in the CP domain.

executable implementation. So long as the class model API, generated from the UML, remains unchanged, different games can be defined using industry standard OCL tools, and implemented. By taking this approach a broad range of dialogue games, similar in basic form to [4] can be specified by using the UML/OCL language.

3. Agents & their Knowledge

An increasingly important topic in dialectical game research is concerned with how agents can play games; the strategies and tactics that can lead to effective goal-oriented play. Therefore, having constructed a system, ProtOCL that is used to define an executable dialogue game platform, we now turn to the topic of constructing agents that can play the games offered by that platform. Classes that define abstract agents that can engage with a ProtOCL game are a part of the platform. These classes can be extended to provide specific game-related functionality and a number of exemplar agents have been developed that can play specific games, e.g. DC, DE, etc. These agents use a three level decision strategy and adopt the heuristics developed in [15,16,17]. One of our aims is to enable different strategies to be developed and for their effectiveness to be tested. To achieve this, one approach might be to develop an arguing agent competition [18,11] in which entrants provide agents which compete against each other to achieve certain goals in a dialogue. The system, and implemented agents are provided with an XML-based knowledge base, constructed using the KBManager graphical tool [13,2] showing Figure 2, which is a part of the ProtOCL platform. Parsers for this format are provided which can be used to load a KB into an agent or to dump the KB into an alternative format, such as AML of AIF, for further processing.

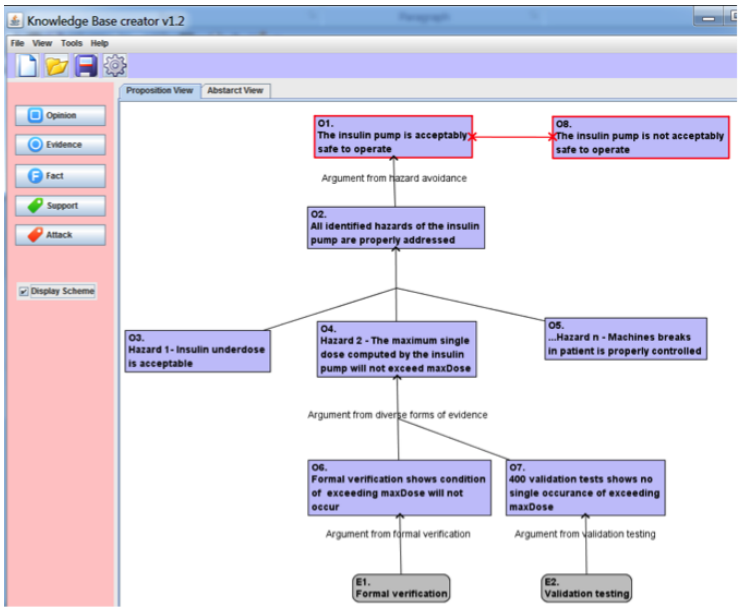

Figure 2. Screenshot of the Argument Knowledge Creator being used to create a knowledge base within the medical domain.

4. Conclusions & Future Work

Together, the ProtOCL platform, a set of implemented agents, and a knowledge base constitute a basic extensible system for playing dialogue games. A key element of this approach is the development and implementation of a core dialogue game API. By conforming to the API, individual new components and techniques can be implemented and deployed which work with the existing components. The platform can be extended by implementing and loading new games, new agents, and new knowledge bases. This enables three factors, the game, the strategies, and the knowledge to be varied whilst exploring and evaluating dialectical games.

One area of future work is to connect the current approach, using UML and OCL to specify and generate executable dialogue games using industry-standard, human-oriented, graphical tools, with existing work on dialogue game specification using the Dialogue Game Description Language (DGDL) which supports formal definition and syntactic verification. This suggests an avenue toward lossless, bi-directional movement between specification and execution platforms that aim for increased industrial uptake and research systems that seek to gain increased insight into both ideal, and real-world dialogical interaction. In section 2 we identify that this approach supports a range of games similar to DC, however the shape and extent of the space of possible games that can be described using this approach is not clear and requires further investigation.

To conclude, this paper has reported on our progress developing methods and tools for the rapid construction, deployment and testing of computational dialectical systems. In particular, we have developed a technology called ProtOCL which uses code generation to produce dialogue games which have been specified using UML/OCL. The overall architecture of the dialogue game platform is component-based so that the individual

components of the system can be developed individually and then assembled together using the dialogue game platform. The paper provides sufficient design rationales and implementation details for reuse and reproduction. It is anticipated that this work contributes significantly to the development of computerised dialectical systems.

References

[1] K. Atkinson. *What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning*. PhD thesis, University of Liverpool, 2005.

[2] S. Bartlett, H. Wang, A. Zolotas, E. Panah, and V. Menon. Intelligent debating system, MSc software engineering team project report, Department of Computer Science, University of York, 2011.

[3] T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10–15):619–641, 2007.

[4] J. D. Mackenzie. Question begging in non-cumulative systems. *Journal Of Philosophical Logic*, 8:117–133, 1979.

[5] I. Rahwan and P. McBurney. Argumentation technology: introduction to the special issue. *IEEE Intelligent Systems*, 22(6):21–23, 2007.

[6] C. Reed and F. Grasso. Recent advances in computational models of argument. *International Journal of Intelligent Systems*, 22(1):1–15, 2007.

[7] C. Reed and S. Wells. Using dialogical argument as an interface to complex debates. *IEEE Intelligent Systems Journal: Special Issue on Argumentation Technology*, 22(6):60–65, 2007.

[8] G. A. W. Vreeswijk. Iacas: an implementation of Chisholm's principles of knowledge. In *In Proceedings of the 2nd Dutch/German Workshop on Non-monotonic Reasoning, Utrecht*, page 225234, 1995.

[9] S. Wells. *Formal Dialectical Games in Multiagent Argumentation*. PhD thesis, University of Dundee, 2007.

[10] S. Wells. Collation of formal dialectical games from the literature. Technical Report UOD-SOC-2012-001, University of Dundee, 2012.

[11] S. Wells and C. Reed. Towards an arguing agents competition: Architectural considerations. In *Proceedings of the 8th International Workshop on Computational Models of Natural Argument (CMNA8)*, 2008.

[12] S. Wells and C. Reed. A domain specific language for describing diverse systems of dialogue. *Journal of Applied Logic*, 10(4):309–329, 2012.

[13] Y. Ye. Argument knowledge base creator for computerized debating systems, MSc thesis, Department of Computer Science, University of York, 2010.

[14] T. Yuan. *Human Computer Debate, A Computational Dialectics Approach*. PhD thesis, Leeds Metropolitan University, 2004.

[15] T. Yuan, D. Moore, and A. Grierson. A human computer debating system and its dialogue strategies. *International Journal of Intelligent Systems*, 22(1):133–156, 2007.

[16] T. Yuan, D. Moore, and A. Grierson. A human-computer dialogue system for educational debate, a computational dialectics approach. *International Journal of Artificial Intelligence in Education*, 18(1):3–26, 2008.

[17] T. Yuan, D. Moore, C. Reed, A. Ravenscroft, and N. Maudet. Informal logic dialogue games in human-computer dialogue. *Knowledge Engineering Review*, 26(3):159–174, 2011.

[18] T. Yuan, J. Schulze, J. Devereux, and C. Reed. Towards an arguing agents competition: Building on argumento. In *Proceedings of the 8th International Workshop on Computational Models of Natural Argument (CMNA8)*, 2008.

[19] T. Yuan and S. Wells. Protocl: Specifying dialogue games using uml and ocl. In F. Grasso, N. Green, and C. Reed, editors, *Thirteenth International Workshop on Computational Models of Natural Argument (CMNA13)*, pages 74–85, 2013.