

An Implementation of a Lightweight Argumentation Engine for Agent Applications

Daniel Bryant and Paul Krause

Department of Computing, University of Surrey, Guildford, GU2 7XH. UK
{d.bryant, p.krause}@surrey.ac.uk

Abstract. Argumentation is becoming increasingly important in the design and implementation of autonomous software agents. In this paper we discuss our current work on a prototype lightweight Java-based argumentation engine that can be used to implement a non-monotonic reasoning component in Internet or agent-based applications. As far as possible we are aiming towards implementing a general purpose argumentation engine that can be configured to conform to one of a range of semantics.

1 Introduction

Argumentation is becoming increasingly important in the design and implementation of autonomous software agents. In particular, it has been proposed that argumentation will facilitate agent-based systems that engage in cooperative problem solving, such as automated negotiation [1] and reasoning over proposals for action [2]. In these situations classical logic-based approaches are often unsuitable. Pertinent information may be insufficient or in contrast there may be too much relevant, but partially incoherent information, and in the case of multi-agent systems, conflicts of interest are inevitable [3].

In this paper we discuss our current work on a prototype lightweight Java-based argumentation engine that can be used to implement a non-monotonic reasoning component in Internet or agent-based applications. The core engine has been built using tuProlog [4] [5], an existing open-source Prolog engine, as its foundation. Although our ultimate goal is to create a general purpose argumentation engine that can be configured to conform to one of a range of semantics, the current version of the engine implements the argumentation-based framework presented in [3] (allowing our engine to determine the acceptability of arguments and construct proofs using an argument game approach to constructing proofs of acceptance [6]), and also standard Prolog inference (allowing us to prototype a variety of metainterpreters that support other forms of argumentation). This paper is structured as follows: In Section 2 we provide motivation for our work and introduce tuProlog. Section 3 introduces the ASPIC argumentation framework, and in Section 4 we discuss how we have implemented this in our engine. We conclude the paper with an overview of planned future work.

2 Laying the Foundations - tuProlog

There has been much recent work on argumentation-based engines, for example, Vreeswijk's *IACAS* [7], and García and Simari's *DeLP* [8] (and later an extension to this work, *P-DeLP*, by Chesñevar and colleagues [9]). However, to our knowledge none of these engines implement support for more than one form of argumentation semantics. It is our belief that agents engaged in reasoning should have access to a general purpose argumentation engine that can be configured to conform to one of a range of semantics.

Our prototype argumentation engine has been built using tuProlog [5] as its foundation. tuProlog is a Java-based Prolog engine which has been designed from the ground up as a thin and lightweight engine that is easily deployable, dynamically configurable and easily integrated into Internet or agent applications [4]. Utilising the Prolog inference provided by the tuProlog engine we can implement a series of metainterpreters for a variety of forms of argumentation. However, this way of implementing an argumentation engine has both a serious performance overhead and a less than ideal interface. In order to avoid these problems and produce an argumentation engine that fully conforms to the spirit of a lightweight Internet enabled tool, we are re-engineering tuProlog by implementing a series of core argumentation algorithms in Java. The first algorithm we have implemented in our engine is presented in [3].

3 The Acceptability of Arguments

In [3] a framework for argument games is presented that is concerned with establishing the acceptability of arguments. Argument games between two players, a proponent (PRO) and opponent (OPP), can be interpreted as constructing proofs of acceptance utilising a dialectical structure [6]. The proponent and opponent share the same (possibly inconsistent) knowledge base and the proponent starts with a main claim to be "proved". The proponent attempts to build an admissible set to support the claim and endeavors to defend any argument against any attack coming from the opponent. The proponent wins the game (proving acceptability of the claim) if all the attacking arguments have been defeated, and the opponent wins if they can find an attacking argument that cannot be defeated. In [3] a prototype web-based implementation (coded in RUBY) of the framework algorithms, entitled "Argumentation System" (AS), is also presented.

4 The Implementation of our Engine

As with AS, AtuP accepts formulas in an extended first-order language and returns answers on the basis of the semantics of credulously preferred sets (as defined in [3]). Facts and beliefs can be expressed in AtuP using standard Prolog syntax with an additional numerical qualifier e.g. `london(raining) 0.8.` and rules can be expressed in a similar way, for example, `flies(X) :- bird(X) 0.8.`

In both of these cases the numerical value is a number in $(0,1]$ that acts as the degree of belief (DOB), or the credibility, of a proposition [3]. As stated in [3], the DOB is currently provided to allow experimentation with different methods of argument evaluation and is not intended to express probabilities or represent values from other numerical theories to reason with uncertain or incomplete information. However, in future work we plan to enhance arguments with possibilities as discussed in, for example, Krause [10], Amgoud [11] or Chesñevar and colleagues [9]. Queries for the support for a claim are expressed using the standard syntax, for example, `?-flies(tweety)`.

When AtUP has finished determining the support for a claim the engine generates a trace of the argument game dialogue, an example of which can be seen in Figure 1. In addition to providing an application programmers interface (API) to allow agent developers to utilise our engine, we have also modified the existing tuProlog graphical user interface to facilitate off-line experimentation with the engine (see Figure 1).

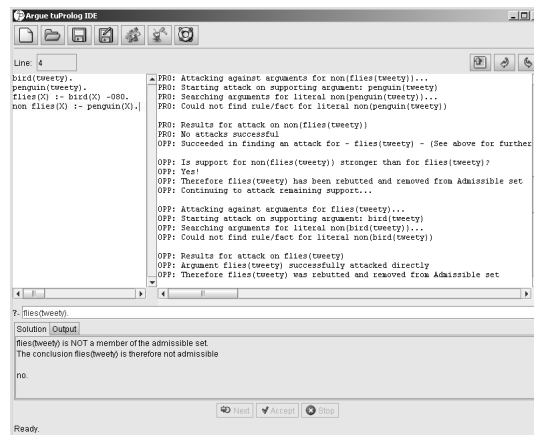


Fig. 1. Screenshot of "Argue tuProlog" GUI. (Left window) allows manipulation of the knowledge base, the (bottom window) allows query entry and displays the results, and the (right window) shows an argument game trace after a query has been executed.

5 Conclusion and Future Work

In this paper we have presented our current work on a lightweight Java-based argumentation engine. We have also discussed the integration of an argumentation-based framework for determining the acceptability of arguments, as presented in [3], into the engine. The end result is a flexible inference engine that is suitable for deployment into Internet and agent applications, and can be utilised to facilitate automated reasoning and decision-making. As far as possible we are aiming

towards implementing a general purpose argumentation engine that can be configured to conform to one of a range of semantics. Our basic position is that we have no prior disposition towards any one model of argumentation. Instead, our plan is to explore a range of models to provide an independent evaluation of their expressive power, performance and scalability.

Acknowledgements

This work was partially supported by the EU IST/STReP ASPIC project, Grant 002307, and an EPSRC PhD Studentship. We gratefully thank members of the ASPIC consortium for useful discussions and in particular Gerard Vreeswijk for his encouragement of our work. We also gratefully thank Mariam Tariq for use of her early implementation work and the tuProlog team at the University of Bologna for their enthusiastic support. Finally, we would like to thank the anonymous reviewers for the insightful and helpful comments.

References

1. I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Agent communication languages: Rethinking the principles. *The Knowledge Engineering Review*, 18(4):343–375, 2003.
2. K. Atkinson, T. K. Bench-Capon, and P. McBurney. A Dialogue Game Protocol for Multi-Agent Argument Over Proposals for Action. In I. Rahwan, P. Moraitis, and C. Reed, editors, *Argumentation in Multi-Agent Systems*, volume 3366 of *LNAI*, pages 149–161. Springer, 2004.
3. L. Amgoud, M. Caminada, S. Doutre, H. Prakken, and G. Vreeswijk. Draft formal semantics for ASPIC system. Technical Report ASPIC Deliverable 2.5, 2005.
4. E. Denti, A. Omicini, and A. Ricci. Multi-paradigm java-prolog integration in tuProlog. *Sci. Comput. Program.*, 57(2):217–250, 2005.
5. E. Denti, A. Omicini, and A. Ricci. tuProlog: A light-weight prolog for internet applications and infrastructures. In I. V. Ramakrishnan, editor, *PADL*, volume 1990 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2001.
6. H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *International Conference on Artificial Intelligence and Law*, pages 53–62, 1999.
7. G. A. W. Vreeswijk. IACAS: an implementation of Chisholm’s principles of knowledge. In *The proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning, Utrecht.*, pages 225–234, 1995.
8. A. J. Garcia and G. R. Simari. Defeasible logic programming: an argumentative approach. *Theory Pract. Log. Program.*, 4(2):95–138, 2004.
9. C. I. Chesnevar, G. R. Simari, T. Alsinet, and L. Godo. A logic programming framework for possibilistic argumentation with vague knowledge. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 76–84, Arlington, Virginia, United States, 2004. AUAI Press.
10. P. Krause, S. Ambler, M. Elvang-Goransson, and J. Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11:113–131, 1995.
11. L. Amgoud and H. Prade. Using arguments for making decisions: a possibilistic logic approach. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 10–17, Arlington, Virginia, United States, 2004. AUAI Press.