DIAMOND 3.0 – A Native C++ Implementation of DIAMOND

Stefan ELLMAUTHALER [a] and Hannes STRASS [a]
[a] *Intelligent Systems Group, Computer Science Institute, Leipzig University, Germany* [1]

Abstract. We present a reimplementation of the DIAMOND system for computing with abstract dialectical frameworks. The original DIAMOND was a script-based tool that called an external ASP solver. This reimplementation uses the clingo library in a native C++ environment and thus avoids communication overhead.

Keywords. abstract argumentation, abstract dialectical frameworks, DIAMOND, answer set programming, clingo, potassco

1. Motivation

Abstract dialectical frameworks (ADFs, [2,1]) are a logic-based formalism for representing knowledge about arguments (statements) and relationships between them. They can be represented by directed graphs, where each node denotes an argument and an edge from a to b encodes that a has an influence on b. The precise nature of this influence (and the influences of other parent nodes of b) are expressed using an acceptance condition, a Boolean function on the parents of b. The semantics of ADFs indicate (according to different justification standards) what statements can be collectively justified [1].

To compute ADF semantics, researchers quickly came to capitalise on the advantages of answer set programming (ASP, [8]). The first ADF system, `ADFsys`, was presented at COMMA 2012 [6]. With the introduction of additional semantics, also a new system was presented: DIAMOND [1,5].

The DIAMOND system uses ASP encodings of ADF semantics to solve reasoning problems. (For example deciding whether an ADF has a (non-trivial) interpretation for a specific semantics, or a statement is sceptically/credulously accepted/rejected by an ADF according to a semantics.) As an ASP back-end, DIAMOND uses tools from the Potsdam answer set solving collection, Potassco [7], more specifically the solver clingo.

Since ADFs are a generalisation of Dung's abstract argumentation frameworks (AFs, [4]), DIAMOND can also be used to compute semantics for AFs. In this function, we submitted DIAMOND 2.0.1 to the First International Competition on Computational Models of Argumentation, ICCMA 2015 (`http://argumentationcompetition.org`). From the competition results, we could observe that DIAMOND 2.0.1 gave an unexpectedly large amount of wrong answers to decision problems. We traced these problems back to technical problems in DIAMOND's communcation with its solver back-end. In this paper, we announce the next version of DIAMOND, version 3.0.x, that aims at overcoming those problems with a change in architecture.

[1] email: {ellmauthaler,strass}@informatik.uni-leipzig.de

2. Main Changes

The "old" DIAMOND (versions up to 2.0.x) basically consisted of a python script that took user commands and an instance filename to combine the right ASP encodings to give to a solver. It used python's interface to operating system pipes and command line calls to communicate with its solver back-end. However, python makes no assertions about data loss in pipes; and indeed, for large ADF instances we observed differences between manual ASP solver calls and the python wrapper script.

The DIAMOND system was completely re-written in C++. The new DIAMOND now uses the clingo library and calls the solver clingo internally by creating a solver object and directly communicating with it within the program instead of invoking the operating system.

The ASP encodings themselves (actually computing the semantics) have not changed, but are now compiled into the program at compile time. In principle, this allows for a standalone executable.

The new version of DIAMOND is available at sourceforge at `https://sourceforge.net/p/diamond-adf/code/ci/cdevelop/tree/`.

3. Conclusion

For future work, we want to do an extensive experimental evaluation of DIAMOND 3.0.x. In particular, we want to compare DIAMOND 2.0.2 and DIAMOND 3.0.x using the probo software [3].

References

[1] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract Dialectical Frameworks Revisited. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 803–809. IJCAI/AAAI, August 2013.

[2] Gerhard Brewka and Stefan Woltran. Abstract Dialectical Frameworks. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 102–111, 2010.

[3] Federico Cerutti, Nir Oren, Hannes Strass, Matthias Thimm, and Mauro Vallati. A benchmark framework for a computational argumentation competition. In *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 459–460. IOS Press, September 2014.

[4] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.

[5] Stefan Ellmauthaler and Hannes Strass. The DIAMOND System for Computing with Abstract Dialectical Frameworks. In *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA)*, volume 266 of *FAIA*, pages 233–240. IOS Press, September 2014.

[6] Stefan Ellmauthaler and Johannes P. Wallner. Evaluating Abstract Dialectical Frameworks with ASP. In *Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA)*, volume 245 of *FAIA*, pages 505–506. IOS Press, 2012.

[7] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):105–124, 2011. Available at `http://potassco.sourceforge.net`.

[8] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(3):1–238, 2012.