goDIAMOND 0.6.6
ICCMA 2017 System Description*

Hannes Strass and Stefan Ellmauthaler

Computer Science Institute, Leipzig University, Germany
{strass,ellmauthaler}@informatik.uni-leipzig.de

Abstract. We describe goDIAMOND as a submission to ICCMA 2017. goDIAMOND builds upon, re-implements and extends previous versions of the DIAMOND system. It has a special focus on dedicated AF technology and straightforward integration of improved solving back-ends.

1 Introduction

DIAMOND (standing for DIAlectical MOdels eNcoDing) loosely refers to a family of solvers for abstract dialectical frameworks (ADFs) [4]. All its members are based on answer set programming (ASP) [11]. More specifically, each member consists of several ASP encodings along with some (typically codified) information on what encodings have to be combined to obtain a specific desired solving/reasoning behaviour. So essentially DIAMOND reduces ADF problems to ASP problems and solves them using an ASP solver (usually clingo [10]). In this paper we describe the latest addition to the family, goDIAMOND, in its role as submission to the second installment of the International Competition on Computational Models of Argumentation (ICCMA 2017).[1]

An earlier version of DIAMOND [8], version 2.0.2 written in Python,[2] participated in the previous (first) installment of ICCMA [12]. Its performance was rather low. Along with a lack of dedication to software development and testing due to numerous other commitments, we identified the communication between DIAMOND and clingo via Python's `os.pipes` as one source of error. This motivated a re-implementation of DIAMOND in C++ [9], leading to versions 3.0.x. That family member, colloquially called CDIAMOND, aimed to mostly avoid such communication by making use of the clingo C++ library with its facilities for creating and invoking solver objects and communicating with them. A student of ours did an experimental comparison between the versions [2] using the probo software [5] and the ICCMA 2015 problem instances.[3] The new system (3.0.1) had fewer wrong answers, but was actually slower than the previous one (version 2.0.2, submitted to ICCMA 2015). We have not yet analysed the reasons for that, but suspect a difference in configuration defaults between stand-alone command-line clingo and the library used for CDIAMOND.

* This work has been partially supported by the German Research Foundation (DFG) under grants BR-1817/7-1 and FOR 1513.

[1] http://www.dbai.tuwien.ac.at/iccma17/

[2] http://python.org

[3] http://argumentationcompetition.org/2015/iccma2015_benchmarks.zip

The goDIAMOND system is (yet) another re-implementation of DIAMOND, using the programming language go.[4] In addition to re-implementing the "program logic" of how to combine encodings, we also improved some of the encodings and the general workflow, and added special functionality for dealing with abstract argumentation frameworks (AFs) [6]. The goDIAMOND software is available at Sourceforge:

`https://sourceforge.net/p/diamond-adf/code/ci/go/tree/go/`

2 Architecture

In principle, goDIAMOND is called on the command-line with arguments indicating the reasoning mode, semantics, input format, and instance file. goDIAMOND uses this information in a straightforward way to select a number of ASP encodings to pass to the solving back-end, in this case clingo 5.0.0. There is roughly one encoding per input format and semantics, and the instance file is passed on as-is. Possible input formats are ADFs in functional syntax, ADFs in formula syntax, bipolar ADFs in formula syntax, and AFs in ASPARTIX syntax. In terms of reasoning modes, goDIAMOND offers all those that are relevant for ICCMA (one interpretation, all interpretations, credulous acceptance, sceptical acceptance). It (currently) supports conflict-free, naive, stage, admissible, complete, preferred, semi-stable, stable, and grounded semantics for all input formats, and ideal semantics for the AF input format.

As one major novelty, goDIAMOND no longer computes preferred semantics using two sequential solver calls (one for computing complete interpretations and one for selecting the maximal ones), but instead uses a disjunctive encoding where the solver takes care of both tasks in one solver call. Likewise, we added disjunctive encodings for naive, stage, and semi-stable semantics. Finally, we re-implemented the encoding associated to AF input as a positive logic program in order to avoid unnecessary guessing.

3 Dedicated AF Algorithms

goDIAMOND contains several implementations of native AF reasoning algorithms:

Grounded: To compute the grounded semantics of an AF, goDIAMOND stores incoming and outgoing edges of each node, and then iteratively looks for nodes without incoming edges, marking them as accepted, marking their successors as rejected, and removing those rejected nodes from all incoming lists, until a fixpoint obtains.

Ideal: goDIAMOND implements the oracle call-based algorithm for computing ideal semantics given by Dunne [7, Theorem 7]. The oracle calls are performed by invoking clingo and parsing its answer.

Dung's Triathlon: For this special ICCMA 2017 task, goDIAMOND computes the grounded semantics via its "native" algorithm and then stable semantics via a call to clingo. Preferred semantics is computed via another solver call; to reduce redundancy, additional constraints are added to the ASP encoding for computing the preferred semantics of the instance, effectively leading the solver only to return those preferred interpretations that are not also stable.

[4] `http://golang.org`

4 Conclusion

While this paper concentrated more on AF aspects of goDIAMOND, there have also been improvements in its ADF solving capabilities. In the future, we want to compare its performance with that of YADF [3]. We would also like to extend goDIAMOND for dealing with further input formats [1].

References

1. Berthold, M.: Extending the DIAMOND system to work with GRAPPA. In: Thimm, M., Cerutti, F., Strass, H., Vallati, M. (eds.) Proceedings of the First International Workshop on Systems and Algorithms for Formal Argumentation (SAFA) co-located with the 6th International Conference on Computational Models of Argument (COMMA 2016), Potsdam, Germany, September 13, 2016. CEUR Workshop Proceedings, vol. 1672, pp. 52–62. CEUR-WS.org (2016), `http://ceur-ws.org/Vol-1672/paper_5.pdf`
2. Bordewisch, S.: Evaluierung von ADF-Solvern der DIAMOND-Familie. Bachelor's thesis, Leipzig University, Computer Science Institute (2017), in German.
3. Brewka, G., Diller, M., Heissenberger, G., Linsbichler, T., Woltran, S.: Solving advanced argumentation problems with answer-set programming. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI). pp. 1077–1083. AAAI Press (Feb 2017)
4. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks revisited. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI). pp. 803–809. IJCAI/AAAI (Aug 2013)
5. Cerutti, F., Oren, N., Strass, H., Thimm, M., Vallati, M.: A benchmark framework for a computational argumentation competition. In: Parsons, S., Oren, N., Reed, C. (eds.) Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA). FAIA, vol. 266, pp. 459–460. IOS Press, The Scottish Highlands, Scotland, United Kingdom (Sep 2014)
6. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *n*-Person Games. Artificial Intelligence 77, 321–358 (1995)
7. Dunne, P.E.: The computational complexity of ideal semantics. Artificial Intelligence 173(18), 1559–1591 (2009)
8. Ellmauthaler, S., Strass, H.: The DIAMOND System for Computing with Abstract Dialectical Frameworks. In: Parsons, S., Oren, N., Reed, C. (eds.) Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA). FAIA, vol. 266, pp. 233–240. IOS Press, The Scottish Highlands, Scotland, United Kingdom (Sep 2014)
9. Ellmauthaler, S., Strass, H.: DIAMOND 3.0 – A native C++ implementation of DIAMOND. In: Baroni, P. (ed.) Proceedings of the Sixth International Conference on Computational Models of Argument (COMMA). FAIA, vol. 287, pp. 471–472. IOS Press, Potsdam, Germany (Sep 2016)
10. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection. AI Communications 24(2), 105–124 (2011), available at `https://potassco.org`.
11. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer set solving in practice. Synthesis Lectures on Artificial Intelligence and Machine Learning 6(3), 1–238 (2012)
12. Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., Vallati, M.: Summary report of the First International Competition on Computational Models of Argumentation. AI Magazine 37(1), 102 (2016)