*June 2021*

# ADF-BDD: An ADF Solver Based on Binary Decision Diagrams[1]

Stefan ELLMAUTHALER [a], Sarah A. GAGGL [b], Dominik RUSOVAC [b], and Johannes P. WALLNER [c]

[a] *Knowledge-Based Systems Group, cfaed, TU Dresden, Germany*

[b] *Logic Programming and Argumentation Group, TU Dresden, Germany*

[c] *Institute of Software Technology, Graz University of Technology, Austria*

**Keywords.** abstract dialectical frameworks, binary decision diagrams, rust-lang

Abstract Dialectical Frameworks [1] (ADF) are a generalisation of Dung's Argumentation frameworks [2]. Multiple approaches for reasoning under various semantics have been proposed over the last decade [3,4,5,6]. We present *"**A**bstract **D**ialectical **F**rameworks solved by **B**inary **D**ecision **D**iagrams, developed in **D**res**d**en"* (ADF-BDD)[2], a novel approach that relies on the translation of the acceptance conditions of a given ADF into reduced ordered binary decision diagrams (roBDD) [7]. Our system is based on the consideration that many otherwise hard to decide problems in ADF semantics (e. g., answering SAT-questions) can be solved in polynomial time on roBDDs (see [8] for an in-depth analysis). Our novel approach differs to the currently used systems, like the SAT-based approach K++ADF [5] or the wide spectrum of answer set programming (ASP) focused approaches like the DIAMOND family (e. g., DIAMOND [3] or GODIAMOND [4]) and YADF [6]. ADF-BDD is written in RUST [9] to provide good performance while enforcing a high amount of memory- and type-safety. In addition the rust-compiler produces highly optimised machine code, while keeping the whole tech stack simple.

ADF-BDD accepts the established input format, introduced first in [10]. There statements are unary predicates `s`, defining the labels and the acceptance conditions are binary predicates `ac`, relating the label to a formula. It allows to enumerate the grounded and complete interpretations, and stable models of the given input instance. The set of statements is the shared signature of all acceptance conditions, hence our implementation uses a single structure to store the nodes of all the roBDDs, which represent each acceptance condition. This allows for efficient caching of nodes and to eliminate duplicate node candidates. Another side-effect is that shared sub-BDDs are computed only once. ADF-BDD provides the explained implementation of roBDDs as the representation of the acceptance conditions. As the instantiation of roBDDs is a computational hard task, it is possible to utilise another state-of-the art competitive library called Biodivine/LibBDD[3]. It is part of the Biodivine software in the AEON project [11]. While LibBDD is faster in

[1] This work is partly supported by the BMBF, Grant 01IS20056_NAVAS, by the Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), and by the DFG through the Collaborative Research Center, Grant TRR 248 project ID 389792660.

[2] `https://github.com/ellmau/adf-obdd`, version 0.2.4, `https://crates.io/crates/adf_bdd`

[3] `https://crates.io/crates/biodivine-lib-bdd`

the instantiation, it is unfortunately not providing all the used features for the efficient application and backtracking of operations on roBDDs of ADF-BDDs implementation. The user can either use one of the two libraries to handle the representation of roBDDs or a hybrid approach, combining the fast instantiation and the efficient operations.

The grounded interpretation is computed via a deterministic approach computing the least fixed point of the approximate operator for ADFs. For the complete interpretations we have chosen to implement a naive approach which lazily checks all possible three valued interpretations. The stable model computation supports this naive approach of lazily checking all possible two-valued interpretations. Furthermore a simple heuristics-based approach is implemented. It allows to incorporate various easily accessible information about the acceptance condition, provided by the roBDD representation. The heuristics then approximate which of the two truth values one statement can have is less costly and computes its influence to the other statements. For the other value we use a no-good like list of value assertions, to steer the further enumeration of possible two valued models.

The performance of our tool[4] is positioned in between the fastest SAT-based approach and the ASP based approaches. This is achieved although the complete semantics are computed in a naive manner. The use of the heuristics based approach for stable models runs faster and more reliable than the naive implementation. This shows that the representation with roBDDs is a promising approach. Future optimisation, more sophisticated learning algorithms, and better heuristics will reduce the gap to K++ADF further.

We present a library (*"adf_bdd"*) for an easy use of the functionality in other software-products and provide an executable (*"adf-bdd"*) to use the library as a straightforward and simple to use solver.

## References

[1] Brewka G, Ellmauthaler S, Strass H, Wallner JP, Woltran S. Abstract Dialectical Frameworks. In: Baroni P, Gabbay D, Giacomin M, van der Torre L, editors. Handbook of Formal Argumentation. College Publications; 2018. p. 237-85.

[2] Dung PM. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif Intell. 1995;77(2):321-58.

[3] Ellmauthaler S, Strass H. The DIAMOND System for Computing with Abstract Dialectical Frameworks. In: Proc. COMMA. vol. 266 of FAIA. IOS Press; 2014. p. 233-40.

[4] Strass H, Ellmauthaler S. GoDIAMOND 0.6.6–ICCMA 2017 System Description. Second International Competition on Computational Models of Argumentation. 2017.

[5] Linsbichler T, Maratea M, Niskanen A, Wallner JP, Woltran S. Advanced algorithms for abstract dialectical frameworks based on complexity analysis of subclasses and SAT solving. Artif Intell. 2022;307:103697.

[6] Brewka G, Diller M, Heissenberger G, Linsbichler T, Woltran S. Solving Advanced Argumentation Problems with Answer Set Programming. TPLP. 2020;20(3):391-431.

[7] Bryant RE. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. ACM Comput Surv. 1992;24(3):293-318.

[8] Darwiche A, Marquis P. A Knowledge Compilation Map. J Artif Intell Res. 2002;17:229-64.

[9] Matsakis ND, II FSK. The rust language. In: Proc. HILT. ACM; 2014. p. 103-4.

[10] Ellmauthaler S, Wallner JP. Evaluating Abstract Dialectical Frameworks with ASP. In: Verheij B, Szeider S, Woltran S, editors. Proc. COMMA. vol. 245. IOS Press; 2012. p. 505-6.

[11] Beneš N, Brim L, Kadlecaj J, Pastva S, Šafránek D. AEON: Attractor Bifurcation Analysis of Parametrised Boolean Networks. In: Proc. CAV. Springer; 2020. p. 569-81.

[4] Dataset and runtimes can be found at `https://doi.org/10.5281/zenodo.6498235`