ADF-BDD.DEV: Debug Abstract Dialectical Frameworks with Binary Decision Diagrams

Stefan Ellmauthaler[0000−0003−3882−4286] and Lukas Gerlach[0000−0003−4566−0224]

Knowledge-Based Systems Group
ScaDS.AI / Faculty of Computer Science / cfaed
TU Dresden, Germany
{firstname.lastname}@tu-dresden.de
https://kbs.inf.tu-dresden.de/

Abstract. Abstract Dialectical Frameworks (ADF) are a well known and understood generalisation of Dung's Argumentation frameworks. Multiple approaches to solve the computation and enumeration of the semantics have been proposed over the last decade. One recent approach is to solve the computational hard problems by translating the acceptance condition of a given ADF into reduced ordered binary decision diagrams (roBDD). While the number of solvers for ADFs is plentyful, they merely give text-based solutions to the problems. The use of roBDDs lays a foundation for straightforward graphical visualization of the underlying ADFs and their solutions. In this work, we present ADF-BDD.DEV, a web-service that generates graphical solutions to ADFs for different semantics, allowing their comparison and to spot the influence of yet undecided statements. We think that this is a first steps towards better explainability and simplifies debugging of ADFs.

1 Introduction

Abstract Dialectical Frameworks (ADFs) [6] are a knowledge representation and reasoning formalism, which generalises the seminal work of Dung [8], so-called Dung's Argumentation Frameworks. The general idea is to represent knowledge as abstract statements. Whether a statement can be accepted is devised by an acceptance condition, usually represented as a propositional formula, where the variables represent the statements of the framework. The semantics of a set of statements with its acceptance conditions map for each statement whether it is acceptable, rejected, or not decided. Various semantics have been defined for ADFs, to have different properties to build upon (i.e. uniqueness, existence, minimality, ...). Alas, most of the semantics are at least on the second level of the polynomial hierarchy and are in general one level higher than the same computational problems for Dung frameworks. The recent proposal [10] of using *reduced ordered binary decision diagrams* (roBDDs) [7] to represent ADFs offers a normal form for acceptance conditions (with a given variable order) and leads to a drop of complexity to match Dung frameworks[1]. The tool ADF-BDD [9]

[1] Intuitively using roBDDs as the input size and the property of roBDDs to answer sat queries in constant time leads to this result.

provides an implementation of that approach by encoding the acceptance conditions of an ADF as a forest of roBDDs [7]. This forest will share and reuse nodes from other roBDDs and allows for a compact graph-based representation. While ADF-BDD achieves outstanding performance, it is still rather technical to use since it is only accessible through a command line interface (CLI) and has mere text-based output. This is an issue ADF-BDD shares with other ADF solvers making problems in the ADF input harder to debug. However, the use of roBDDs allows for a natural graphical visualization of the computational models and solutions pruduced by ADF-BDD. In this work, we present ADF-BDD.DEV; offering ADF-BDD as a public web-service[2] that displays the underlying forest of roBDDs of a given ADF in different stages of solving. Thereby, our tool assists users to understand and compare the different possible semantics for solving ADFs and simplifies to debug the inputs. In particular, it offers a concise view of acceptance conditions for statements that remain undecided. To the best of our knowledge, ADF-BDD.DEV is the first ADF solver to offer this kind of visualisation.

2 Solving ADFs with roBDDs

We recall basics of Abstract Dialectical Frameworks and refer the interested reader to the recent Handbook of Formal Argumentation [1,5]. For more insights on roBDDs with ADFs, we kindly point to the respective previous work [10].

Definition 1. *An* ADF *is a triple* $D \coloneqq (S, L, C)$ *where* S *is a fixed finite set of* statements*;* $L \subseteq S \times S$ *is a set of* links*; and* $C \coloneqq \{\varphi_s\}_{s \in S}$ *consists of* acceptance conditions *for statements, which correspond to propositional formulas* $\varphi ::= s \in S \mid \bot \mid \top \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$ *over the parents* $P(s) \coloneqq \{s' \in S \mid (s', s) \in L\}$ *of statement* s*.*

Since links can be determined by acceptance conditions, throughout this paper we will mostly omit links and simply define ADFs as a tuple consisting of statements and their respective acceptance conditions. We are following the newly proposed representation ofADFs with roBDDs [10].

Definition 2. *A* binary decision diagram (BDD) $\mathcal{B}$ *over variables* X *is a rooted directed acyclic graph with two external nodes labeled with* 0 *or* 1 *and internal nodes* u *with two outgoing edges given by* $low(u)$ *and* $high(u)$*. Each internal node* u *is associated with a variable* $x \in X$*, denoted by* $var(u) = x$*. A BDD is* ordered*, if on all paths the variables respect a linear order* $x_1 < x_2 < \cdots < x_n$ *and it is reduced if it satisfies the following two conditions:*

(a) if $var(u) = var(v)$*,* $low(u) = low(v)$ *and* $high(u) = high(v)$*, then* $u = v$*, for each pair of internal nodes* u, v*; and*
(b) $low(u) \neq high(u)$ *for each internal node* u*.*

[2] ADF-BDD.DEV- https://adf-bdd.dev

Paths from the root to 1 *correspond to partial assignments on* X *(true for high and false for low), and their completions (assigning remaining variables in* X*) to models of* $\mathcal{B}$*. For a formula* φ*, we use* $\mathcal{B}_\varphi$*, to denote a binary decision diagram for* φ *over the variables of* φ *s.t. the models of* φ *coincide with the models of* $\mathcal{B}_\varphi$*. Define* restriction $\mathcal{B}_\varphi[x_1/v_1, \ldots, x_n/v_n]$ *of* $\mathcal{B}_\varphi$ *s.t. each* x_i *is set to* $v_i \in \{0,1\}$ *by redirecting incoming edges of each node* u *with* $var(u) = x_i$ *to* $low(u)$*, if* $v_i = 0$*, and to* $high(u)$*, if* $v_i = 1$*; and removing* u*.*

By representing ADFs as roBDDs the two previous definitions are combined:

Definition 3. *The* BDD representation $\mathcal{B}(D) = (\mathcal{B}_{\varphi_{s_1}}, \ldots, \mathcal{B}_{\varphi_{s_n}})$ *of an ADF* $D = (S, C)$ *is a tuple consisting of one BDD for each acceptance condition* φ_{s_i} *of* $s_i \in S$ *where* $1 \leq i \leq n = |S|$*.*

The semantics of a given ADF are based on three-valued interpretations. Such an interpretation is a function $I : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ that maps each statement to either true, false, or undecided. We call an interpretation *two-valued*, denoted by I_2, if $\forall s \in S : I(s) \in \{\mathbf{t}, \mathbf{f}\}$. Additionally the *information ordering* $\leq_i$ is defined as the reflexive transitive closure of the relation $<_i$ with $\mathbf{u} <_i v$ for $v \in \{\mathbf{t}, \mathbf{f}\}$. We lift $\leq_i$ and $<_i$ to interpretations by $I' \leq_i I$ iff $I'(s) \leq_i I(s)$ for each $s \in S$, and $I' <_i I$ if $I' \leq_i I$ and for some $s \in S$ we have $I'(s) <_i I(s)$. By $\mathcal{B}_\varphi[I] \coloneqq \mathcal{B}_\varphi[s/1 : I(s) = \mathbf{t}][s/0 : I(s) = \mathbf{f}]$ we define the *partial evaluation* of $\mathcal{B}_\varphi$ with respect to I.

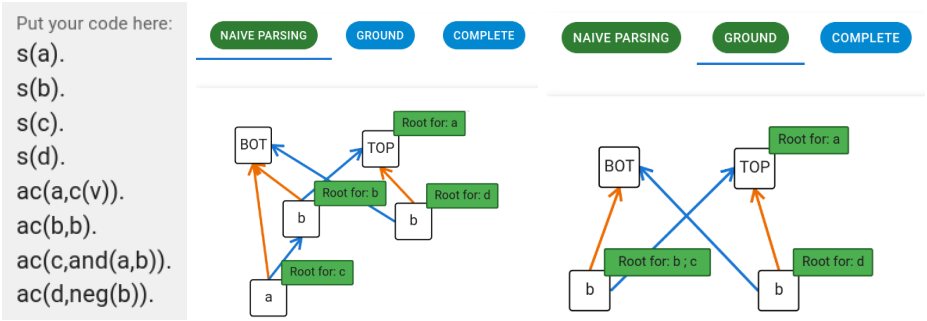
Definition 4. *Let* $D = (S, B)$ *be an ADF,* $\mathcal{B}(D)$ *its BDD-representation, and* I *be a three-valued interpretation over* S*. The characteristic operator* $\Gamma_D(I) = I'$ *is defined by the* revisited interpretation I' *of* I*, such that for each* $s \in S$

$$I'(s) = \begin{cases} \mathbf{t} & \text{if the reduced } \mathcal{B}_{\varphi_s}[I] \text{ is a tautology (i.e. is a 1 node);} \\ \mathbf{f} & \text{if the reduced } \mathcal{B}_{\varphi_s}[I] \text{ is an inconsistency (i.e. is a 0 node);} \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

We are now in position to define Dung's standard semantics for ADFs that is currently supported by ADF-BDD.DEV.

Definition 5. *Let* $D = (S, C)$ *be an ADF,* $\mathcal{B}(D)$ *its BDD-representation, and* I *a three-valued interpretation.* I *is complete in* D *if* $I = \Gamma_D(I)$*, and* I *is grounded in* D *if* I *is the least fixed-point of* Γ_D *for* $I_\mathbf{u}$ *with* $I_\mathbf{u}(s) = \mathbf{u}$ *for each* $s \in S$*.*

We additionally define the *reduced ADF* $D^{I_2} \coloneqq (S^{I_2}, C^{I_2})$ for a two-valued interpretation (i.e. all statements are mapped to $\mathbf{t}$ or $\mathbf{f}$) I_2, a where $S^{I_2} \coloneqq \{s \in S \mid I_2(s) = \mathbf{t}\}$ and $C^{I_2} \coloneqq \{\varphi_s[s'/\bot : I_2(s') = \mathbf{f}] \mid s \in S^{I_2}, s' \in S\}$. Analogously, we define the corresponding BDD-representation $\mathcal{B}_D^{I_2} \coloneqq \mathcal{B}_D[s/0 : I_2(s) = \mathbf{f}]$ and remove all statements and corresponding roBDDs, where $I_2(s) = \mathbf{t}$. Let G be the grounded interpretation of $\mathcal{B}_D^{I_2}$, I_2 is a *stable model* of D if for all $s \in S^{I_2} : I_2(s) = \mathbf{t}$ implies $G(s) = \mathbf{t}$.



(a) Input (b) Parsed (c) Grounded

Fig. 1: Screenshots: Analysing an ADF in ADF-BDD.DEV. Orange lines represent low-edges and blue lines high-edges of a given roBDD.

3 Visualising ADF Semantics using roBDDs

To debug an ADF and to illustrate the different semantics, it is natural to give a visualisation of the corresponding roBDD-representation as a graph. The underlying forest of roBDDs is presented in a single graph marked with multiple root nodes where nodes from different roBDDs are merged whenever possible. For ADF-BDD.DEV, we rely on a state of the art, feature-rich, web-based library for graph visualisation[3] [17]. The library has many layouting algorithms buillt-in that we can use for roBDDs; one of them is the so-called "dagre" layout.[4] This implementation combines various previous works [13,14,2,3,16] to rank nodes into a hierarchy while minimising the number of crossing edges. This rank-based layout comes very natural for the merged forest of roBDDs: Intuitively, all nodes without outgoing edges go to the first rank (i.e. the 0 and 1 nodes). Then on each next rank, we have all nodes that only have outgoing edges to nodes in the previous ranks. In the following, we give an example how we can analyse a given ADF using its roBDD-representations on ADF-BDD.DEV:

The ADF $D = (S, C)$ in the input in Figure 1a contains four statments, `a` through `d` (S), represented by the unary predicates `s`. Its four corresponding acceptance conditions (C) are represented by the binary predicate `ac` that relates each statement to the actual condition as follows: (1) `a` is assumed to be true ("verum"). (2) `b` is true if `b` is true (which is self-supporting). (3) `c` is true if `a` and `b` are true. (4) `d` is true if `b` is not true. We use a common syntactic representation for ADFs [4,15], first introduced by [12] and described in detail later [11]. In the future, we also plan to incorporate graphical ADF editing.

Figure 1b shows the roBDD representation $\mathcal{B}(D)$ for D as a forest of the underlying roBDDs with merged nodes as described above. The visualisation helps to see how the truth value of a statement s depends on other statements

[3] G6 - https://g6.antv.antgroup.com/en/
[4] Dagre original implementation https://github.com/dagrejs/dagre

by starting at the root for s and then following the possible paths to the top. To simplify the indentification of a subtree that belongs to a statement, our ADF-BDD.DEV visualisation allows to hightlight those trees by clicking the "root" nodes (and all other nodes as well). For instance, the root for `a` is directly at the "TOP" (i.e. 1) node, which indicates that `a` is true. To obtain the truth value for `c`, we start at the bottom left node. If `a` is known to be false, we follow the orange path, which is the low-edge in the roBDD, yielding that `c` is false as well. The blue path corresponds to the high-edge in the roBDD and represents the case, where `a` is true. Then, if `b` is also true, we find that `c` is true. Unsurprisingly, this direcly corresponds to acceptance condition (3) above.

The intuitive procedure of determining truth values iteratively by following paths and pluging in known values into the nodes of the roBDDs is exactly what is done by the characteristic operator Γ_D. Since we intuitively start with the interpretation $I_{\mathbf{u}}$ where all statements are undecided, this procedure gives us the grounded interpretation I_{ground} for D. The grounded interpretation (and the other semantics introduced in Section 2) can again be visualised with ADF-BDD.DEV. Figure 1c shows the partial evaluation of $\mathcal{B}(D)$ with respect to I_{ground}. This representation allows to debug the ADF D and to analyse why some statements are still undecided. One can see that the statements `b` and `c` are still dependent on the outcome of `b`. In addition, it is also shown that whatever the result for `b` and `c` will be, statement `d` will behave with the inverse truth value. This is an important step in understanding and explaining further results and semantics and allows a way to directly address yet undecided truth-value assignments and their reasons. To the best of our knowledge, this is the first work to give this kind of insight.

4 Outlook

In the future, we plan to further improve user experience on ADF-BDD.DEV. For example, we want to allow graphical editing of ADFs instead of only text based input and we want to allow to edit the produced roBDD representation directly. As a long term goal, the graphical editing can be enhanced with on-the-fly analysis and other advanced features to provide a fully-fledged "IDE" for ADF editing. Furthermore, better tooltips and hints shall assist even untrained users to become familiar with ADFs and their sematics by making use of the roBDD presentation in a didactic fashion. In its current form, we are convinced that ADF-BDD.DEV simplifies debugging of ADFs for individuals that are already familiar with ADFs. Looking further, we think that our powerfully backed yet easy to access tool ADF-BDD.DEV bears great potential for making work on ADFs more approachable for already experienced users but also for newcomers that want to get some first hands-on experience.

Acknowledgements This work was supported in DFG grant 389792660 (TRR 248), by BMBF in grants ITEA-01IS21084 (InnoSale), and in DAAD grant 57616814 (SECAI).

References

1. Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.): Handbook of Formal Argumentation. College Publications (2018)
2. Barth, W., Mutzel, P., Jünger, M.: Simple and Efficient Bilayer Cross Counting. Journal of Graph Algorithms and Applications **8**(2), 179–194 (2004). https://doi.org/10.7155/jgaa.00088, http://jgaa.info/getPaper?id=88
3. Brandes, U., Köpf, B.: Fast and Simple Horizontal Coordinate Assignment. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) Graph Drawing. pp. 31–44. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_3
4. Brewka, G., Diller, M., Heissenberger, G., Linsbichler, T., Woltran, S.: Solving advanced argumentation problems with answer set programming. TPLP **20**(3), 391–431 (2020)
5. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran., S.: Abstract dialectical frameworks. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 5, pp. 237–285. College Publications (2018)
6. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks. an overview. IfCoLog Journal of Logics and their Applications **4**(8), 2263–2317 (October 2017)
7. Bryant, R.E.: Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Comput. Surv. **24**(3), 293–318 (1992)
8. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
9. Ellmauthaler, S., Gaggl, S.A., Rusovac, D., Wallner, J.P.: Adf - BDD : An ADF solver based on binary decision diagrams. In: Toni, F. (ed.) Proceedings of the 9th International Conference on Computational Models of Argument (COMMA 2022). FAIA, vol. 220146, pp. 355–356. IOS Press (September 2022). https://doi.org/10.3233/FAIA220170
10. Ellmauthaler, S., Gaggl, S.A., Rusovac, D., Wallner, J.P.: Representing abstract dialectical frameworks with binary decision diagrams. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Proceedings of the 16th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR 2022). Lecture Notes in Computer Science, vol. 13416, pp. 177–198. Springer (2022). https://doi.org/10.1007/978-3-031-15707-3_14
11. Ellmauthaler, S., Straß, H.: The DIAMOND system for argumentation: Preliminary report. In: Fink, M., Lierler, Y. (eds.) Proceedings of the Sixth International Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP) (September 2013)
12. Ellmauthaler, S., Wallner, J.P.: Evaluating Abstract Dialectical Frameworks with ASP. In: Verheij, B., Szeider, S., Woltran, S. (eds.) Proc. COMMA. vol. 245, pp. 505–506. IOS Press (2012)
13. Gansner, E., Koutsofios, E., North, S., Vo, K.P.: A technique for drawing directed graphs. IEEE Transactions on Software Engineering **19**(3), 214–230 (Mar 1993). https://doi.org/10.1109/32.221135
14. Jünger, M., Mutzel, P.: 2-Layer Straightline Crossing Minimization: Performance of Exact and Heuristic Algorithms. Journal of Graph Algorithms and Applications **1**(1), 1–25 (1997). https://doi.org/10.7155/jgaa.00001

15. Linsbichler, T., Maratea, M., Niskanen, A., Wallner, J.P., Woltran, S.: Advanced algorithms for abstract dialectical frameworks based on complexity analysis of subclasses and SAT solving. Artif. Intell. **307**, 103697 (2022)
16. Sander, G.: Layout of compound directed graphs. workingPaper (1996). https://doi.org/10.22028/D291-25806, https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/25862, accepted: 2005-06-23
17. Wang, Y., Bai, Z., Lin, Z., Dong, X., Feng, Y., Pan, J., Chen, W.: G6: A web-based library for graph visualization. Visual Informatics **5**(4), 49–55 (Dec 2021). https://doi.org/10.1016/j.visinf.2021.12.003