Representing Abstract Dialectical Frameworks with Binary Decision Diagrams[⋆]

Stefan Ellmauthaler[1][0000−0003−3882−4286], Sarah A. Gaggl[2][0000−0003−2425−6089], Dominik Rusovac[2][0000−0002−3172−5827], and Johannes P. Wallner[3][0000−0002−3051−1966]

[1] Knowledge-Based Systems Group, cfaed, TU Dresden, Germany,
[2] Logic Programming and Argumentation Group, TU Dresden, Germany,
`firstname.lastname@tu-dresden.de`
[3] Institute of Software Technology, TU Graz, Austria,
`wallner@ist.tugraz.at`

Abstract. Abstract dialectical frameworks (ADFs) are a well-studied generalisation of the prominent argumentation frameworks due to Phan Minh Dung. In this paper we propose to use reduced ordered binary decision diagrams (roBDDs) as a suitable representation of the acceptance conditions of arguments within ADFs. We first show that computational complexity of reasoning on ADFs represented by roBDDs is milder than in the general case, with a drop of one level in the polynomial hierarchy. Furthermore, we present a framework to systematically define heuristics for search space exploitation, based on easily retrievable properties of roBDDs and the recently proposed approach of weighted faceted navigation for answer set programming. Finally, we present preliminary experiments of an implementation of our approach showing promise both when compared to state-of-the-art solvers and when developing heuristics for reasoning.

Keywords: abstract dialectical frameworks · binary decision diagrams.

1 Introduction

Computational argumentation is an active research topic within the broader field of Artificial Intelligence, which provides dialectical reasons in favour of or against disputed claims [3]. Deeply rooted in non-monotonic reasoning and logic programming, formal frameworks for argumentation provide the basis for heterogeneous application avenues, such as in legal or medical reasoning [2]. Within the field, formalisms in so-called abstract argumentation have proven to be useful for argumentative reasoning. Here arguments are represented as abstract entities, and only the inter-argument relations decide argumentative acceptance, which is

[⋆] 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.

formalized via argumentation semantics. Several semantics exist, ranging from a more skeptical to a more inclusive stance towards acceptance of arguments.

Based on the prominent approach by Dung [12], a core formal approach to abstract argumentation are abstract dialectical frameworks [7], or ADFs for short, which also represent arguments as abstract entities, and allow for flexible relations between arguments, modelled as Boolean functions. Recently, ADFs were shown to be applicable in the legal field [1], in online dialog systems [20], and also for text exploration [9].

However, ADFs face the barrier of high complexity of reasoning [24,16], reaching up to the third level of the polynomial hierarchy. To address this obstacle, several approaches were proposed and studied: considering various fragments of the ADF language [18], quantified Boolean formulas [11], and utilizing advanced techniques in answer set programming [23,6].

A method for addressing high complexity, nevertheless, was not considered so far in depth for (abstract) argumentation: knowledge compilation [10]. A key principle behind knowledge compilation is that tasks of high complexity are translated to formal languages where reasoning has milder complexity, while at the same time taking possible translation performance issues into account. Applying techniques of knowledge compilation to abstract argumentation appears natural: abstract argumentation formalisms themselves can be seen as "argument compilations" of knowledge bases, e.g., ADFs can be instantiated from knowledge bases [22].

In this paper we take up the opportunity to fill this gap in the research landscape and propose to model a lingering source of complexity of ADFs, namely that of representing acceptance conditions per argument, via the prominent language of binary decision diagrams (BDDs) [8], with the following main contributions.

- We first formally introduce ADFs whose acceptance conditions are represented via BDDs.
- We show that complexity of reasoning for ADFs represented via reduced and ordered BDDs enjoys the same complexity (drop) as bipolar ADFs [24] or argumentation frameworks [13], after the compilation procedure.
- The representation via BDDs opens up a different opportunity: poly-time decidability of several tasks on BDDs allows to extract various kinds of information from the ADF. We use a recently proposed framework [15] that allows for exploring search spaces to arrive at a framework for developing heuristics for reasoning in ADFs.
- We present preliminary experiments showing promise of our approach in two directions: while at current we do not outperform the state-of-the-art ADF solver k++ADF [18], our results suggest that (i) computing the grounded semantics is as good via our approach than for k++ADF (including compilation times) and (ii) heuristics based on our framework show promise of performance increase.

$$\begin{tikzcd}
& a \arrow[loop above] & \varphi_a = (a \to (b \lor \neg c)) \land (\neg d \lor b) \\
b \arrow[ur] \arrow[r, leftrightarrow] & c \arrow[u] \arrow[r, leftrightarrow] & d \arrow[ul] \\
\varphi_b = \neg c & \varphi_c = b \land \neg d & \varphi_d = c
\end{tikzcd}$$

Fig. 1: Components of example ADF D_1, where the node labels represent statements and the attached formulae represent the respective acceptance condition.

2 Background

Abstract Dialectical Framework. We recall basics of Abstract Dialectical Frameworks (ADFs) and refer the interested reader to the recent Handbook of Formal Argumentation [3,7] for more details.

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

Example 1. Let $D_1 = (\{a, b, c, d\}, \{(a \to (b \lor \neg c)) \land (\neg d \lor b), \neg c, b \land \neg d, c\})$. Figure 1 illustrates the components of D_1.

The semantics of ADFs are based on three-valued interpretations. An interpretation is a function $I : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ that maps each statement to either $\mathbf{t}$ (true), $\mathbf{f}$ (false) or $\mathbf{u}$ (undefined). An interpretation I is *two-valued*, denoted by I_2, if $I(s) \in \{\mathbf{t}, \mathbf{f}\}$ for each $s \in S$. We define an *information ordering* $\leq_i$ such that $\leq_i$ is the reflexive transitive closure of $<_i$ and $\mathbf{u} <_i v$ for $v \in \{\mathbf{t}, \mathbf{f}\}$. This ordering is extended 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 $\varphi[I] := \varphi[s/\top : I(s) = \mathbf{t}][s/\bot : I(s) = \mathbf{f}]$ we define the *partial evaluation* of φ with respect to I.

Definition 2. *Let $D = (S, C)$ be an ADF 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*

$$I'(s) = \begin{cases} \mathbf{t} & \textit{if } \models \varphi_s[I]; \\ \mathbf{f} & \textit{if } \varphi_s[I] \models \bot; \\ \mathbf{u} & \textit{otherwise.} \end{cases}$$

We are now in position to define Dung's standard semantics for ADFs.

Definition 3. *Let $D = (S, C)$ be an ADF. A three-valued interpretation I*

- *is admissible in D if $I \leq_i \Gamma_D(I)$;*
- *is complete in D if $I = \Gamma_D(I)$;*
- *is grounded in D if I is the least fixed-point of Γ_D;*
- *is preferred in D if I is $\leq_i$-maximal admissible in D.*

It is well-known that an ADF has a unique grounded interpretation. A two-valued interpretation is called a model of D, if this interpretation is complete in D. Stable models are defined as follows.

Definition 4. *Let $D = (S, C)$ be an ADF and I_2 be a two-valued interpretation. Define the* reduced ADF *$D^{I_2} := (S^{I_2}, C^{I_2})$ where $S^{I_2} := \{s \in S \mid I_2(s) = \mathbf{t}\}$ and $C^{I_2} := \{\varphi_s[s'/\bot : I_2(s') = \mathbf{f}] \mid s \in S^{I_2}, s' \in S\}$. If I_2 is a model of D and for the grounded interpretation G of D^{I_2} it holds that $I_2(s) = \mathbf{t}$ implies $G(s) = \mathbf{t}$, then I_2 is a* stable model *of D.*

Main reasoning tasks on ADFs under a semantics σ include credulous reasoning, i.e., asking whether there is a σ interpretation assigning a queried statement to true, and skeptical reasoning, i.e., is it the case that all σ interpretations assign a queried statement to true. Verification refers to the task of deciding whether a given interpretation is a σ interpretation.

Example 2 (cont'd). The grounded interpretation of D_1 is $\{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\}$. Note that therefore $\emptyset$ is also complete in D_1. The other interpretation complete in D_1 is $\{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{f}\}$. In fact, D_1 has no stable models.

Reduced Ordered Binary Decision Diagram. A (reduced ordered) binary decision diagram [8] is an efficient representation of a Boolean function. We follow the convention of referring to reduced ordered binary decision diagrams as BDDs.

Definition 5. *A* binary decision diagram *$\mathcal{B}_\varphi$ over variables X of a formula φ 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 $var(u) = x$. It 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.

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*

1. *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*
2. *removing u.*

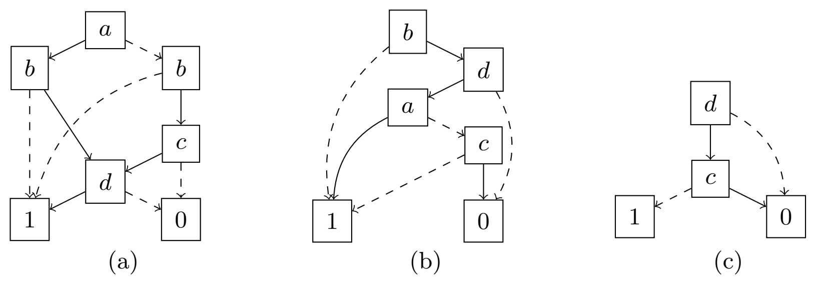

Fig. 2: BDDs of the acceptance condition φ_a with respect to D_1: (a) with ordering $a < b < c < d$; (b) with ordering $b < d < a < c$; and (c) which corresponds to (b) restricted such that a is set to 1 and b is set to 0. Solid lines denote low and dashed lines denote high.

By $vars(\mathcal{B}_\varphi) = \{var(u_0), \ldots, var(u_m)\}$ *we denote the variables of internal nodes of* $\mathcal{B}_\varphi$ *and by* $\#\mathcal{B}_\varphi \coloneqq 2 \cdot m$ *we denote the size of* $\mathcal{B}_\varphi$*, which corresponds to the number of edges for* m *internal nodes.*

It is well-known [17] that reduced ordered binary decision diagrams admit each of the following operations in time polynomial in the size of the BDD: consistency check, validity check, clausal entailment check, implicant check, equivalence check, sentential entailment check, model counting and model enumeration.

However, note that the ordering matters when it comes to the size of a BDD. Figure 2 illustrates that the lexicographic ordering leads to a BDD (a) with 10 edges including two nodes labeled with b, whereas $b < d < a < c$ leads to a BDD (b) of size 8, including exactly one node for each variable. In fact, finding an optimal variable ordering for ordered binary decision diagrams is an NP-hard problem [5]. Even approximating it, is hard [21].

3 Representing ADFs as BDDs

An ADF is defined by acceptance conditions (propositional formulas) over statements (propositions). Utilizing BDDs directly leads to the following definition. See Figure 3 for a BDD representation of our running example ADF.

Definition 6. *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* $i = 1 \ldots n$.

We show that complexity of reasoning on ADFs represented by BDDs coincides with complexity results of *argumentation frameworks* (AFs) [12,13]. Based on polytime procedures reducing and restricting BDDs [8], we can show a polytime result for computing the result of the characteristic operator.

Theorem 1. *Given the BDD representation* $\mathcal{B}(D)$ *of an ADF* D*, the result of applying* Γ_D *to any three-valued interpretation* I *can be computed in polynomial time.*

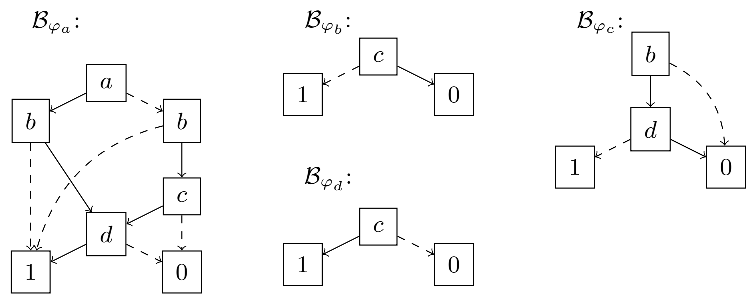
$\mathcal{B}_{\varphi_a}$:
a
b
b
c
d
1
0
$\mathcal{B}_{\varphi_b}$:
c
1
0
$\mathcal{B}_{\varphi_d}$:
c
1
0
$\mathcal{B}_{\varphi_c}$:
b
d
1
0

Fig. 3: The BDD representation of D_1 using lexicographic ordering.

Further, by previous results [24, Theorem 3.18], we obtain several upper bounds directly for ADFs represented by BDDs.

Theorem 2. *Given an ADF D represented by $\mathcal{B}(D)$, it holds that*

- *verification under admissibility and complete semantics is in P,*
- *credulous reasoning under admissibility, complete, and preferred semantics is in NP;*
- *verification under preferred semantics is in coNP;*
- *and skeptical reasoning under preferred semantics is in Π_2^P.*

Stable and grounded semantics are not covered by the corresponding theorem [24], nevertheless complexity exhibits a drop, as well. We first deal with grounded semantics, as an ingredient for stable semantics.

Theorem 3. *Given an ADF D represented by $\mathcal{B}(D)$, there is a polynomial algorithm that computes the grounded interpretation of D.*

Based on this result, it follows that verifying whether a model of an ADF represented by BDDs is stable is in P, and credulous and skeptical reasoning lies on the first level of the polynomial hierarchy, since all checks regarding the reduct are then polytime. For both credulous and skeptical reasoning, the membership results hold via a direct non-deterministic guess of an interpretation.

Corollary 1. *Verifying whether a three-valued interpretation is a model or is stable in an ADF represented by BDDs is in P. Moreover, credulous reasoning is in NP and skeptical reasoning in coNP.*

Regarding hardness, one can directly utilize hardness results for AFs (see [13] for an overview), since one can translate a given AF directly (in polytime) to an ADF under the BDD representation. Thus, credulous reasoning under admissible, complete, stable, and preferred semantics is NP-complete, verification of preferred interpretations is coNP-complete, and skeptical reasoning under stable is coNP-complete and Π_2^P-complete for preferred semantics.

4 Search Space Exploitation: Profiting from BDDs

Fichte et al. [15] define a navigation framework for answer set programming (ASP) [19], called *weighted faceted navigation*, that allows for quantifying the effect of navigation steps in a search space. So far it has been used to *explore* solution spaces, we utilize it to *exploit* information provided by a search space.

The idea is to navigate the search space (the set of interpretations) via heuristics that weight decisions (assigning truth values to statements). A decision, called here *facet*, is either inclusive (assigning true, denoted by $+s$) or exclusive (assigning false, denoted by $-s$). We use symbols $+$ and $-$ to distinguish true and false. A route is then an iteratively extended sequence of such facets (with the possibility of backtracking). That is, a route can be seen as a partial (two-valued) assignment on the statements, together with a partial evaluation of each acceptance condition under this partial assignment (like $\varphi[I]$ for a partial I). Weight functions then indicate heuristic goals, by assigning weights to facets, given a current route (current partial assignment). To make use of strengths of BDDs, we can include weights that are hard to compute on general formulas, but tractable on BDDs (such as number of models). We formalize these ingredients next into a generic framework for designing heuristics on ADFs represented by BDDs. We first define facets formally. The intuition behind a statement s being a facet is that it is contained in the variables of at least one BDD.

Definition 7. *We define* facets *of* $D = (S, C)$ *by* $\mathcal{F}(D) = \mathcal{F}(D)^+ \cup \mathcal{F}(D)^-$ *where* $\mathcal{F}(D)^+ = \{+s \mid s \in \bigcup_{B \in \mathcal{B}(D)} vars(B)\}$ *denotes* inclusive facets *and* $\mathcal{F}(D)^- = \{-s \mid s \in \bigcup_{B \in \mathcal{B}(D)} vars(B)\}$ *denotes* exclusive facets.

In ADFs with acceptance conditions represented via Boolean formulas, statements inside acceptance conditions might have no effect. For instance for $\phi_a = (b \vee \neg b) \wedge c$ it follows that the status of b is irrelevant for acceptance of a (formally in ADFs such links are called redundant). Reduced BDDs directly take care of such forms of redundancy, which leads to a simple observation: utilizing BDDs (for faceted navigation) reduces statements to consider. Formally, a navigation step towards facet $f \in \{+s, -s\} \subseteq \mathcal{F}(D)$ over an ADF D means that we modify $\mathcal{B}(D)$ with respect to f, denoted by $\mathcal{B}(D)[f]$, by applying a restriction to each BDD of the BDD representation $\mathcal{B}(D)$ of D s.t.

$$\mathcal{B}(D)[f] \coloneqq \begin{cases} (\mathcal{B}_{\varphi_{s_1}}[s/1], \ldots, \mathcal{B}_{\varphi_{s_n}}[s/1]), & \text{if } f = +s; \\ (\mathcal{B}_{\varphi_{s_1}}[s/0], \ldots, \mathcal{B}_{\varphi_{s_n}}[s/0]), & \text{if } f = -s. \end{cases}$$

We define a *route* $\delta \coloneqq \langle f_1, \ldots, f_n \rangle$ as a finite sequence of facets $f_i \in \mathcal{F}(D)$ denoting n arbitrary navigation steps over D. We assume that such a route does not contain complementary facets. By Δ_D we denote all possible routes over D. We define $\mathcal{B}(D)^\delta = \mathcal{B}(D)[f_1] \ldots [f_n]$, which means that $\mathcal{B}(D)$ is first restricted by f_1, then $\mathcal{B}(D)[f_1]$ is restricted by f_2 and so on. For simplicity, we write D^δ to denote the restriction $\mathcal{B}(D)^\delta$.

Example 3 (cont'd). Suppose we activate $+c \in \mathcal{F}(D_1)$, then $\mathcal{F}(D_1^{\langle +c \rangle}) = \mathcal{F}(D_1) \setminus \{+c, -c\}$. Proceeding by activating $-b \in \mathcal{F}(D_1^{\langle +c \rangle})$, we obtain $\mathcal{F}(D_1^{\langle +c, -b \rangle}) = \emptyset$.

That is, we "choose" to assign c true and b false, and iteratively shrink the remaining search space (available facets).

To make decisions during search, we need to make sense of the search space. The *weight* of a facet f is a parameter that quantifies what kind of effect activating f has on the search space.

Definition 8. *Let $D = (S, C)$, $\delta \in \Delta_D$ and $f \in \mathcal{F}(D^\delta)$. The* weight *of f is a function $\omega : \mathcal{F}(D^\delta) \times \Delta_D \to \mathbb{N}$.*

That is, $\omega(f, \delta)$ gives a weight of a facet (a potential next decision) with respect to a current route δ. In the following, we introduce several weights.

Definition 9. *Let $D = (S, C)$, $\delta \in \Delta_D$, $f \in \mathcal{F}(D^\delta)$ correspond to $s \in S$ and $\mathcal{B}_{\varphi_s} \in \mathcal{B}(D)^\delta$. We define the following weights*

- $\omega_M(f, \delta) := |M(\varphi_s)|$ *if $f = +s$, otherwise $\omega_M(f, \delta) := n - |M(\varphi_s)|$ where $M(\varphi_s)$ denotes models of φ_s and $n := 2^{|vars(\mathcal{B}_{\varphi_s})|}$* (model-counting weight)
- $\omega_{AI}(f, \delta) := |\{\mathcal{B} \in \mathcal{B}(D^\delta) \mid s \in vars(\mathcal{B})\}|$ (active impact weight)
- $\omega_{PI}(f, \delta) := |vars(\mathcal{B}_{\varphi_s})|$ (passive impact weight)
- $\omega_P(f, \delta)$ *is the number of paths leading to 1 (resp. 0) in $\mathcal{B}_{\varphi_s}$, if $f = +s$ (resp. $f = -s$)* (path-counting weight)
- $\omega_{MD}(f, \delta)$ *is the length of the largest simple path in $\mathcal{B}_{\varphi_s}$* (max-depth weight)

Each of the introduced weights refers to a value that can be computed in polynomial time using the BDD representation. Every weight tries to approximate information about the search space of each BDD in the ADF. The most obvious one is the model-counting weight, which counts how many models exist and is completely based on semantics notions. A bit more exact is the path-counting weight, by considering the semantics notions as well as the representation in BDD structures. The max-depth is a bit more exotic, as it computes the maximum length of the given BDD. Intuitively, this is a measurement on the maximum number of variables needed to decide the truth value of the BDD, and allows one to approximate how many additional values are required to be decided in order to ensure that the BDD represents a truth constant. Passive impact weight follows the same idea, but only allows one to see how many variables will have an impact on the BDD over all possible paths in the BDD. The active impact weight has the same idea as max-depth and the passive impact weight, but operates on a more global estimation by computing how many other BDDs might be impacted by the chosen facet.

Navigation-based heuristics, as introduced next, use weights for computing semantics. In a preliminary analysis we focus on enumerating stable interpretations. As computing stable models relies on finding two-valued models, here the objective is to use weights in order to find facets (make decisions) that ease the search for two-valued models using the characteristic operator Γ_D.

Similar to the notion of navigation modes in previous works [15], we are interested in minimal and maximal weighted facets of an ADF D^δ with respect

Table 1: Facet weights of D_1 on the empty route $\langle\rangle$.

	$+a$	$+b$	$+c$	$+d$	$-a$	$-b$	$-c$	$-d$
ω_M	11	1	1	1	5	1	3	1
ω_{AI}	1	2	3	2	1	2	3	2
ω_{PI}	4	1	2	1	4	1	2	1
ω_P	4	1	1	1	3	1	2	1
ω_{MD}	4	1	1	2	4	1	1	2

to weight ω as defined by $min_\omega(D^\delta) := \{f \in \mathcal{F}(D^\delta) \mid \forall f' \in \mathcal{F}(D^\delta) : \omega(f,\delta) \leq \omega(f',\delta)\}$ and $max_\omega(D^\delta) := \{f \in \mathcal{F}(D^\delta) \mid \forall f' \in \mathcal{F}(D^\delta) : \omega(f,\delta) \geq \omega(f',\delta)\}$. That is, we rank facets according to maximum (minimum) weight (given the current route). A navigation-based heuristic h suggests a set of facets to activate on the current route, by recursively determining minimal or maximal weighted facets in a given order with respect to specified weights. For a given ADF, a heuristic is determined by a current route and a list of weighting functions, to flexibly allow that decision shall be reached according to prioritized weight functions. For instance, $\Omega = \langle max_{\omega_M}, min_{\omega_{PI}}\rangle$ specifies that facets shall be ordered by considering maximally ω_M (highest priority) and, in case of equal ranking, consider minimizing ω_{PI} (second-level priority).

Definition 10. *Let $D = (S, C)$ be an ADF. A* ranking *$\Omega = \langle m_1, \ldots, m_n\rangle$ is a sequence with $m_i \in \{min_{\omega_0}, \ldots, min_{\omega_k}, max_{\omega_{k+1}}, \ldots, max_{\omega_\ell}\}$ and weights ω_j for $j = 0 \ldots \ell$. A* navigation-based heuristic *is a function*

$$h(\Omega, \mathcal{F}(D^\delta)) := m_n(m_{n-1}(\cdots(m_2(m_1(D^\delta)))\cdots)).$$

A heuristic is essentially defined by Ω, which specifies preferences of minimal or maximal weighted facets. Aiming at enumerating stable models, we conducted experiments using two heuristics h_0 and h_1, which should add intelligence to the search for two-valued models using BDDs as described by Algorithm 1. Heuristics h_0, h_1 are defined by $\Omega_0 = \langle max_{\omega_{PI}}, min_{\omega_{AI}}, min_{\omega_P}\rangle$ and $\Omega_1 = \langle min_{\omega_P}, max_{\omega_{PI}}\rangle$, respectively. The intuition behind Ω_0 is to find those statements, which have the highest impact on the BDDs, with a small amount of own variables and view choices in reaching a specific truth value. Ω_1 represents the approach to reduce the possible choices in one BDD to reach a specific truth value and maximises the impact on other BDDs afterwards. If the heuristic does not find a unique best option, we follow the BDD ordering.

Example 4 (cont'd). Applying rankings Ω_0 and Ω_1 on the data illustrated in Table 1, we see that heuristic h_0 suggests facets $\{-a\}$, since $max_{\omega_{PI}}(\mathcal{F}(D_1)) = \{+a, -a\}$, $min_{\omega_{AI}}(\{+a, -a\}) = \{+a, -a\}$ and finally $min_{\omega_P}(\{+a, -a\}) = \{-a\}$ for D_1 on the empty route. However, heuristic h_1 suggests to activate facet $+c$, since h_1 first considers $min_{\omega_P}(\mathcal{F}(D_1)) = \{+b, -b, +c, +d, -d\}$ and afterwards $max_{\omega_{PI}}(\{+b, -b, +c, +d, -d\}) = \{+c\}$.

The recursive Algorithm 1 uses a specified heuristic (such as Ω_0 and Ω_1), an ADF in BDD representation and an empty set of nogoods. If one of the nogoods

Algorithm 1 Recursively Enumerating Two-valued Models

```
Procedure: models
In: BDD representation $\mathcal{B}(D^\delta)$; set of nogoods $F \subseteq \mathcal{F}(D)$; heuristic $\Omega$
Out: two-valued models of $D$;
1: set the set of two-valued models $M := \emptyset$;
2: if $f' = +s$ (resp. $f' = -s$) implies $\not\models \varphi_s$ (resp. $\varphi_s \not\models \bot$) for each $f' \in F$
3:     if $\mathcal{F}(D^\delta) \neq \emptyset$ then choose a facet $f \in h(\Omega, \mathcal{F}(D^\delta))$ and activate $f$ on $\delta$
4:         traverse routes $\delta' \in \Delta(D^\delta)$
5:             set $\mathcal{B}'$ to the BDD that corresponds to $s$ in $\mathcal{B}(D^{\delta'})$
6:             update $\mathcal{B}(D^{\delta'})$ to obtain $\mathcal{B}''(D^{\delta'})$ where
7:             $B'$ is set to $\top$, if $f = +s$, otherwise $B'$ is set to $\bot$
8:             set $M := M \cup$ models($\mathcal{B}''(D^{\delta'})$, $F$, $\Omega$);
9:         add the inverse facet $\overline{f}$ of $f$ to nogoods $F$ and activate $\overline{f}$ on $\delta$
10:        set $M := M \cup$ models($\mathcal{B}(D^\delta)$, $F$, $\Omega$);
11:    else return $\{\{s \in S \mid C \in D^\delta, C = \top, S \in D\}\}$
12: return $M$
```

is violated by the current state of the BDDs, we cannot find a two valued model with the given nogoods (Line 2 and 12). Otherwise we choose and activate a facet, based on the given heuristic. In Line 4 we now identify all interpretations of the BDD, which correspond to the activated facet value. The following four lines then propagate the truth values of one of these interpretations to all other BDDs and reduce the facet corresponding BDD to be only $\top$ (resp. $\bot$) and go one step down in the recursion by using this new BDD representation as the updated input. After the propagation of each of the corresponding interpretations, we can now deduce by tertium non datur that the truth value of the chosen facet needs to be its inverse. Therefore we now assume the inverse and add the chosen facet as a nogood. This will allow to go down the recursion depth on Line 10 with the updated nogoods. Finally in Line 11 we see that no further facet can be chosen, therefore all BDDs are either $\top$ or $\bot$ and we have found a two-valued model. Afterwards a simple stability check for the two valued models can be done.

5 Preliminary Experiments

We have implemented the presented ideas as a tool, called `adf-bdd`. It stands for *"**A**bstract **D**ialectical **F**rameworks solved by **B**inary **D**ecision **D**iagrams, developed in **D**res**d**en"*. The tool allows one to compute the grounded, complete, and stable models of a given ADF. We support the currently prevalent input format for ADFs [14]. Due to this choice of compatibility we need to note that all presented tests have transformed the given acceptance conditions into ROBDDs. In a nutshell, the system allows the use of two different BDD libraries, a state of the art competitive library by the `Biodivine` tool [4] and our own implementation. Our own implementation can compute all of the previously introduced weights and has more efficient computations of backtracking, as well as a data structure which allows to exploit the common signature of all acceptance condi-

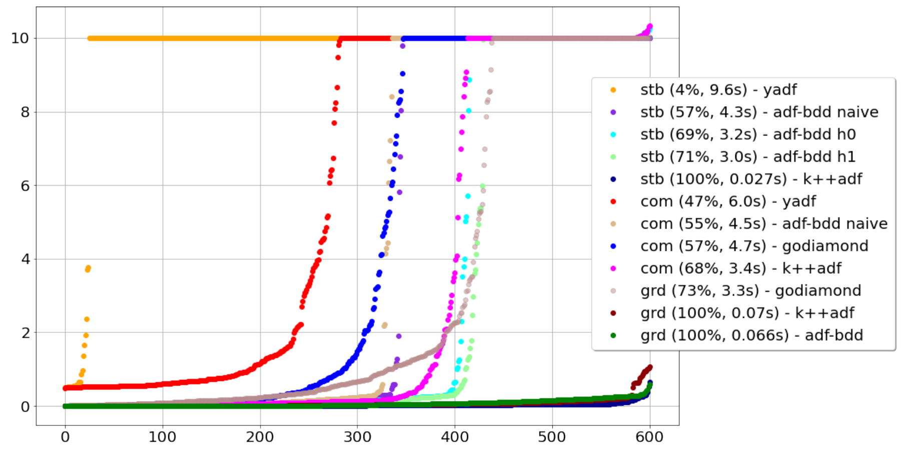

Fig. 4: Experiments on the mean run-times (seconds) of various solvers.

tions and similar properties of BDDs and ADFs. `Biodivine` is more efficient in the instantiation, so we provide a combined approach too.

For the implementation we have a deterministic and straight forward approach for the grounded semantics, which computes the least fixed point of the Γ_D operator. The complete semantics are handled by a naive approach, where all possible three valued interpretations are constructed in a lazily evaluated list and are checked by applying relational operations on the corresponding roBDDs. Stable semantics have been implemented in a similar naive way as well as with the proposed Algorithm 1 and the two discussed heuristics. Note that we do not exploit different variable orders so far and that we use the occurrence order of statements from the input file as the only used variable order.

We have chosen to compare our approach[4] with the currently fastest solver k++adf [18](version 2021-03-31), goDiamond [23] (version goDiamond 2017-06-26), and yadf [6](version 0.1.1). The latter two tools are using answer set solving (ASP) to solve the computational problems. The test machine specifications are as follows: An Intel Xeon E5-2637v4 Quadcore 64bit Processor with 3.7GHz frequency, 384 GB working memory, running a Debian 9.13 Linux, with exclusive computation time for the tests. Note that none of the tools used an excessive amount of the provided memory and has been capped by CPU-performance. Due to the very different running times of the tools, we have chosen to use `hyperfine` as a benchmarking tool harness. The tool decided how many runs shall be done to reduce the load bias and provided mean performance times over up to 900 runs per test-feature. Therefore all times are the mean run-times of all runs per instance for each tool. In addition we imposed a ten second time-out limit for all the computations. We have seen in preliminary tests that the behaviour on the timeout-count does not derivate noticeable with a twenty seconds time limit. As

[4] https://github.com/ellmau/adf-obdd/releases/tag/v0.2.4-beta.1 v0.2.4-beta.1

the test cases, we have chosen the already multiple times used benchmarking set of 600 instances, already used by YADF and K++ADF. Due to text limitations we need to keep this analysis short, full evaluations and references of the data and the datasets can be found at https://doi.org/10.5281/zenodo.6498235.

We summarize our results in Figure 4. Note that a missing tool indicates it does not support that semantics. For the grounded semantics our tool competes with K++ADF, suggesting that the BDD representation does not present a significant barrier for the considered instances (BDD compilation time is included in running times). The computation of the complete semantics shows that the naive approach is already as good as the ASP based GODIAMOND. For the stable models we see that our approach is better than the ASP based YADF, while there is still a gap towards K++ADF. Regarding heuristics, our results suggest that use of the two heuristics improved overall performance, suggesting that the heuristics improved search space navigation.

6 Conclusions

In this work we proposed to utilize knowledge compilation in the form of BDDs for the abstract argumentation formalism of ADFs. After showing milder complexity after the compilation process, we proceeded to present a generic framework for devising heuristics using the recently proposed framework of faceted navigation, which makes use of features (weights) that are computationally hard to obtain on Boolean formulas, but direct to retrieve from BDDs. Our preliminary experiments suggest that heuristics arising from the framework can indeed be help in the search space navigation, but cannot compete with the current state-of-the-art SAT-based approach of K++ADF. This latter solver is based on a candidate generation and subsequent verification procedure. Combining heuristics for search space navigation using BDDs and the SAT-based approach appear intriguing: potentially one could combine interesting heuristics on argumentation problems together with advanced SAT techniques.

References

1. Al-Abdulkarim, L., Atkinson, K., Bench-Capon, T.J.M.: A methodology for designing systems to reason with legal cases using abstract dialectical frameworks. Artif. Intell. Law **24**(1) (2016) 1–49
2. Atkinson, K., Baroni, P., Giacomin, M., Hunter, A., Prakken, H., Reed, C., Simari, G.R., Thimm, M., Villata, S.: Towards artificial argumentation. AI Magazine **38**(3) (2017) 25–36
3. Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L., eds.: Handbook of Formal Argumentation. College Publications (2018)
4. Beneš, N., Brim, L., Kadlecaj, J., Pastva, S., Šafránek, D.: AEON: Attractor bifurcation analysis of parametrised boolean networks. In: Proc. CAV, Springer (2020) 569–581
5. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. Computers **45**(9) (1996) 993–1002

6. Brewka, G., Diller, M., Heissenberger, G., Linsbichler, T., Woltran, S.: Solving advanced argumentation problems with answer set programming. Theory Pract. Log. Program. **20**(3) (2020) 391–431
7. 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. College Publications (2018) 237–285
8. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers **100**(8) (1986) 677–691
9. Cabrio, E., Villata, S.: Abstract dialectical frameworks for text exploration. In: Proc. ICAART, SciTePress (2016) 85–95
10. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. **17** (2002) 229–264
11. Diller, M., Wallner, J.P., Woltran, S.: Reasoning in abstract dialectical frameworks using quantified boolean formulas. Argument Comput. **6**(2) (2015) 149–177
12. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2) (1995) 321–358
13. Dvořák, W., Dunne, P.E.: Computational problems in formal argumentation and their complexity. In Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L., eds.: Handbook of Formal Argumentation. College Publications (2018) 631–688
14. Ellmauthaler, S., Wallner, J.P.: Evaluating Abstract Dialectical Frameworks with ASP. In: Proc. COMMA. Volume 245., IOS Press (2012) 505–506
15. Fichte, J.K., Gaggl, S.A., Rusovac, D.: Rushing and strolling among answer sets – navigation made easy. In: Proc. AAAI'22. (2022)
16. Gaggl, S.A., Rudolph, S., Straß, H.: On the decomposition of abstract dialectical frameworks and the complexity of naive-based semantics. J. Artif. Intell. Res. **70** (2021) 1–64
17. Lai, Y., Liu, D., Wang, S.: Reduced ordered binary decision diagram with implied literals: A new knowledge compilation approach. Knowl. Inf. Syst. **35**(3) (2013) 665–712
18. 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** (2022) 103697
19. Marek, V.W., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: a 25-Year Perspective. Artificial Intelligence. (1999) 375–398
20. Neugebauer, D.: Generating defeasible knowledge bases from real-world argumentations using D-BAS. In: Proc. AI ˆ3@AI*IA 2017. Volume 2012 of CEUR Workshop Proceedings., CEUR-WS.org (2017) 105–110
21. Sieling, D.: On the existence of polynomial time approximation schemes for OBDD minimization (extended abstract). In: Proc. STACS. Volume 1373 of Lecture Notes in Computer Science., Springer (1998) 205–215
22. Strass, H.: Implementing instantiation of knowledge bases in argumentation frameworks. In: Proc. COMMA. Volume 266 of FAIA., IOS Press (2014) 475–476
23. Strass, H., Ellmauthaler, S.: Godiamond 0.6.6–iccma 2017 system description. Second ICCMA (2017)
24. Strass, H., Wallner, J.P.: Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. Artif. Intell. **226** (2015) 34–74