Representational Succinctness of Abstract Dialectical Frameworks

Hannes Strass

Computer Science Institute, Leipzig University, Leipzig, Germany

Abstract Representational succinctness is the ability of a formalism with model-theoretic semantics to express interpretation sets in a space-efficient way. In this paper we analyse the representational succinctness of abstract dialectical frameworks (ADFs) under the two-valued model semantics. We also compare ADFs' succinctness to related formalisms like propositional logic, argumentation frameworks (under stable extension semantics), and normal logic programs (under supported model semantics). This concerns a fundamental computational aspect of (argumentation) formalisms, as representation size is important both for storing descriptions and reasoning over them.

1 Introduction

When can a formal model of argumentation be considered computational?

To us, being "computational" means that the model can – in principle and in practice – be stored and processed by a computer. One crucial aspect of storage and processing is representation size. If a model produces descriptions of infinite size, the model is not computational in principle, since infinite descriptions cannot be processed by finite machines. If a model produces descriptions of at least exponential size in the best case, it is computational in principle but not in practice, as reasoning cost (in terms of computation time) and representation size tend to correlate positively.

Clearly Dung's abstract argumentation frameworks (AFs) [1] are computational (in principle whenever they are finite, in practice if they are of "practical" size), since arguments and pairs of arguments (that is, attacks) could be represented by bit strings (among other possibilities). However, it has long been noted that the means of *expression* offered by AFs are quite limited. "Means of expression" here refers to expressiveness in the sense of realisability, that is, the interpretation-sets that can be produced by some AF. This has recently been made technically precise by Dunne et al. [2], who basically showed that introducing new, purely technical arguments is sometimes inevitable when using AFs for representation purposes. However, due to their very nature, the dialectical meaning of such technical arguments might be – ironically – debatable.

A more expressive alternative to AFs are the abstract dialectical frameworks (ADFs) of Brewka and Woltran [3,4]. There – even in the restricted subclass of *bipolar* ADFs – arguments can also support each other, in addition to the AF notion of attack. ADFs could be called the lovechild of AFs and logic programs, since they combine intuitions and semantics from Dung-style abstract argumentation as well as logic programming [4,5,6]. While on the abstract level, ADFs are intended to function as "argumentation middleware" – a target formalism for translations from more concrete formalisms

that is still sufficiently expressive. As part of the ADF success story, we just mention a reconstruction of the Carneades model of argument [7], an instantiation of simple defeasible theories into ADFs [8], and recent applications of ADFs for legal reasoning and reasoning with cases by Al-Abdulkarim et al. [9,10].

In this paper, we approach argumentation formalisms as knowledge representation formalisms, since they are used to represent knowledge about arguments and relationships between these arguments. We employ this view to analyse the representational capabilities of ADFs. Due to their roots in AFs and logic programs [3,5], we also compare the representational capabilities of these formalisms in the same setting. In this initial study we restrict ourselves to looking at two-valued semantics, more specifically the ADF model semantics, which corresponds to AF stable extension semantics, and the supported model semantics for logic programs. One of our main results is that ADFs are – in a formally defined sense – representationally strictly more efficient than normal logic programs. Moreover, even bipolar ADFs can polynomially express some model sets that normal logic programs cannot. This is especially significant given that ADFs and normal logic programs (under the semantics we consider here) are equally expressive [11] and have the same computational complexity.[1]

Analysing the expressiveness of argumentation formalisms is a quite recent strand of work. Its ascent can be attributed to Dunne et al. [2], who studied realisability for argumentation frameworks (allowing to introduce new arguments as long as they are never accepted). Likewise, Dyrkolbotn [15] analysed AF realisability under projection (allowing to introduce new arguments) for three-valued semantics. Baumann et al. [16] studied the expressiveness of the subclass of "compact" AFs, where each argument is accepted at least once. In previous work of our own, we analysed the expressiveness of ADFs and compared it with that of AFs and logic programs, also restricted to a two-valued setting, however only studying expressive power without addressing representational efficiency [11]. Finally, and most recently, Puehrer [17] analysed the realisability of three-valued semantics for ADFs. In the present paper we take the next step and also consider the sizes of realisations, as is not uncommon in logic-based AI [18,19,20,21]. Indeed, representation size is a fundamental practical aspect of knowledge representation languages: universal expressiveness is of little use if the model sets to express require exponential-size knowledge bases even in the best case!

On the technical side of our analysis, we can make use of a powerful methodology to analyse and compare the expressive power and expressive efficiency of knowledge representation formalisms, a methodology that was introduced in a landmark paper by Gogic et al. [22]. According to their definition, a formalism X is *exponentially more succinct* than a formalism Y if and only if every knowledge base of Y has an equivalent knowledge base in X that is at most polynomially larger, but there is a knowledge base of formalism X whose smallest equivalent knowledge base in Y is exponentially larger [22]. The word "equivalent" here means syntactical equality of the sets of models, and so explicitly rules out introducing and projecting out new variables. As [22] pointed out, representational succinctness is different from computational complexity, as the latter is only interested in preserving the answer to a decision problem, and the former is interested in preserving the precise set of models under a fixed vocabulary.

[1] More precisely, the model existence problem is equally complex, NP-complete [12,13,4,14].

It might seem that viewing formalisms as sets of knowledge bases associated with a two-valued semantics is a restricting assumption. However, this language representation model is universal in the sense that it is just another way of expressing languages as sets of words over $\{0,1\}$. Using an n-element vocabulary $A_n = \{a_1, \ldots, a_n\}$, a binary word $w = x_1 x_2 \cdots x_n$ of length n is encoded as the set $M_w = \{a_i \in A_n \mid x_i = 1\} \subseteq A_n$. For example, using the vocabulary $A_3 = \{a_1, a_2, a_3\}$, the binary word 101 of length 3 corresponds to the set $M_{101} = \{a_1, a_3\}$. Consequently, a set L_n of words of length n can be represented by a set $X_{L_n} \subseteq 2^{A_n}$ of subsets of A_n; conversely, each sequence $(X_n)_{n\geq 0}$ of sets with $X_n \subseteq 2^{A_n}$ uniquely determines a language $L = \bigcup_{n\geq 0} L_n$ over $\{0,1\}$. In this paper we use "language" to refer to object-level languages while "formalism" refers to meta-level languages, such as propositional logic, argumentation frameworks, abstract dialectical frameworks, and logic programs.

Formally, the syntax of ADFs is defined via Boolean functions. However, we are interested in representations of ADFs. So we have to fix a representation of ADFs via fixing a representation of Boolean functions. We choose to use propositional formulas, as is customary in most of the literature [3,4,14]. Exceptions to this custom are the works of Brewka et al. [23], who use Boolean circuits, and [5], where we used characteristic models (that is, represented the formulas in disjunctive normal form). For the subclass of bipolar ADFs, yet no uniform representation exists, which is another question that we will address in this paper.

By propositional formulas over a vocabulary A we mean formulas over the Boolean basis $\{\wedge, \vee, \neg\}$, that is, trees whose leaves (sinks) are atoms from A or the logical constants true $\top$ or false $\bot$, and internal nodes are either unary ($\neg$) or binary ($\wedge$,$\vee$). We also make occasional use of Boolean circuits, where "trees" above is replaced by "directed acyclic graphs"; in particular, we allow unbounded fan-in, that is, reusing subcircuits. As usual, the depth of a formula (circuit) is the length of the longest path from the root to a leaf (sink). Figure 1 below shows an example of depth 3.

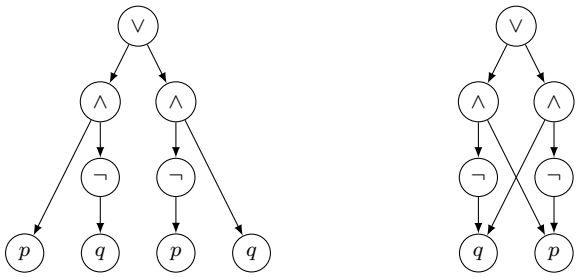


Figure 1: Representing $(p \wedge \neg q) \vee (q \wedge \neg p)$ as a formula tree (left) and a circuit (right).

Analysing the expressive power and representation size of Boolean circuits is an established subfield of computational complexity [24]. This has led to a number of language classes whose members can be recognised by Boolean circuits satisfying certain restrictions. We will need the class AC^0, which contains all languages $L = \bigcup_{n\geq 0} L_n$

for which there exist $d, k \in \mathbb{N}$ such that for each $n \in \mathbb{N}$, there exists a Boolean circuit C_n of depth at most d and size at most n^k where the models of C_n exactly express X_{L_n}. In other words, every language $L \in \mathsf{AC}^0$ can be recognised by a family of polynomial-size Boolean circuits of a fixed maximal depth that is independent of word length.

The rest of the paper proceeds as follows. We next formally define the succinctness relation for formalisms, and what it means for a formalism to polynomially express an object-level language. Then we define the notion of bipolar propositional formula and show a correspondence to the semantical notion of bipolar Boolean function as implicitly defined by Brewka and Woltran [3]. The main part of the paper analyses the succinctness of (bipolar) ADFs and compares it to the other mentioned languages. We then conclude with a discussion of remaining problems and possible future work.

2 Background

We presume a finite set A of atoms (statements, arguments), the *vocabulary*. A knowledge representation formalism interpreted over A is then some set $\mathcal{F}$; a (two-valued) semantics for $\mathcal{F}$ is a mapping $\sigma : \mathcal{F} \to 2^{2^A}$ that assigns sets of two-valued models to knowledge bases $\mathsf{kb} \in \mathcal{F}$. (So A is implicit in $\mathcal{F}$.) Below, we write the set of realisable model sets of a formalism as $\sigma(\mathcal{F}) = \{\sigma(\mathsf{kb}) \mid \mathsf{kb} \in \mathcal{F}\}$.

Definition 1. *Let A be a finite vocabulary, $\mathcal{F}_1, \mathcal{F}_2$ be formalisms that are interpreted over A, have size measures $\|\cdot\|_1$ and $\|\cdot\|_2$, and two-valued semantics σ_1 and σ_2, respectively. Define $\mathcal{F}_1^{\sigma_1} \leq_s \mathcal{F}_2^{\sigma_2}$ if and only if there is a $k \in \mathbb{N}$ such that for all $\mathsf{kb}_1 \in \mathcal{F}_1$ with $\sigma_1(\mathsf{kb}_1) \in \sigma_1(\mathcal{F}_1) \cap \sigma_2(\mathcal{F}_2)$, there is a $\mathsf{kb}_2 \in \mathcal{F}_2$ with $\sigma_1(\mathsf{kb}_1) = \sigma_2(\mathsf{kb}_2)$ and $\|\mathsf{kb}_2\|_2 \leq \|\mathsf{kb}_1\|_1^k$.*

Intuitively, any knowledge base from $\mathcal{F}_1$ with an equivalent counterpart in $\mathcal{F}_2$ must have an equivalent counterpart *that is at most polynomially larger*. Note that succinctness talks only about those model sets that both can express, so it is most meaningful when comparing languages that are equally expressive, that is, whenever $\sigma_1(\mathcal{F}_1) = \sigma_2(\mathcal{F}_2)$. As usual, we define $\mathcal{F}_1 <_s \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_s \mathcal{F}_2$ and $\mathcal{F}_2 \not\leq_s \mathcal{F}_1$, and $\mathcal{F}_1 \cong_s \mathcal{F}_2$ iff $\mathcal{F}_1 \leq_s \mathcal{F}_2$ and $\mathcal{F}_2 \leq_s \mathcal{F}_1$. The relation $\leq_s$ is reflexive, but not necessarily antisymmetric or transitive.

Definition 2. *A formalism $\mathcal{F}$ can polynomially express a language $L = \bigcup_{n \geq 0} L_n$ under semantics $\sigma : \mathcal{F} \to 2^{2^A}$ if and only if there is a $k \in \mathbb{N}$ such that for each positive $n \in \mathbb{N}$ there is a knowledge base $\mathsf{kb}_n \in \mathcal{F}$ of that formalism such that $\sigma(\mathsf{kb}_n) = L_n$ and $\|\mathsf{kb}_n\| \in O(n^k)$.*

We next introduce some specific object-level languages that we will use. First of all, the language PARITY contains all odd-element subsets of the vocabulary. Formally, for $A_n = \{a_1, \ldots, a_n\}$ with $n \geq 1$ we have

$$\text{PARITY}_n = \{M \subseteq A_n \mid \exists m \in \mathbb{N} : |M| = 2m + 1\}$$

As explained before, then $\text{PARITY} = \bigcup_{n \in \mathbb{N}, n \geq 1} \text{PARITY}_n$. It is a textbook result that PARITY is expressible by polynomial-size propositional formulas [25]; for example,

we can define $\Phi_1^{\text{PARITY}}(a_1) = a_1$ and for $n \geq 2$ set

$$\Phi_n^{\text{PARITY}}(a_1, \ldots, a_n) = (\Phi_{n_\downarrow}^{\text{PARITY}}(a_1, \ldots, a_{n_\downarrow}) \wedge \neg\Phi_{n_\uparrow}^{\text{PARITY}}(a_{n_\downarrow+1}, \ldots, a_n)) \vee (\neg\Phi_{n_\downarrow}^{\text{PARITY}}(a_1, \ldots, a_{n_\downarrow}) \wedge \Phi_{n_\uparrow}^{\text{PARITY}}(a_{n_\downarrow+1}, \ldots, a_n))$$

with $n_\downarrow = \lfloor \frac{n}{2} \rfloor$ and $n_\uparrow = \lceil \frac{n}{2} \rceil$. (This construction yields a formula of logarithmic depth and therefore polynomial size.) It is also a textbook result that PARITY cannot be expressed by depth-bounded polynomial-size circuits, that is, PARITY $\notin \mathsf{AC}^0$ [25].

As another important class, threshold languages are defined for $n, k \in \mathbb{N}$ with $n \geq 1$ and $k \leq n$:

$$\text{THRESHOLD}_{n,k} = \{M \subseteq A_n \mid k \leq |M|\}$$

That is, $\text{THRESHOLD}_{n,k}$ contains all interpretations with at least k true atoms. The special case $k = \lceil \frac{n}{2} \rceil$ leads to the majority languages, $\text{MAJORITY}_n = \text{THRESHOLD}_{n,\lceil \frac{n}{2} \rceil}$ containing all interpretations where at least half of the atoms in the vocabulary are true.

The following subsections introduce the particular knowledge representation formalisms we study in this paper. All will make use of a vocabulary A; the results of the paper are all considered parametric in such a given vocabulary.

2.1 Argumentation Frameworks

Dung [1] introduced argumentation frameworks as pairs $F = (A, R)$ where A is a set of (abstract) arguments and $R \subseteq A \times A$ a relation of attack between the arguments. The purpose of semantics for argumentation frameworks is to determine sets of arguments (called *extensions*) which are acceptable according to various standards. For a given extension $S \subseteq A$, the arguments in S are considered to be accepted, those that are attacked by some argument in S are considered to be rejected, and all others are neither, their status is undecided. We will only be interested in so-called *stable* extensions, defined as follows: A set $S \subseteq A$ of arguments is *conflict-free* iff there are no $a, b \in S$ with $(a, b) \in R$. A set S is a *stable extension* for (A, R) iff it is conflict-free and for all $a \in A \setminus S$ there is a $b \in S$ with $(b, a) \in R$. In stable extensions, each argument is either accepted or rejected by definition, thus the semantics is two-valued. The size of an argumentation framework $F = (A, R)$ is $\|F\| = |A| + |R|$.

2.2 Abstract Dialectical Frameworks

An *abstract dialectical framework* is a tuple $D = (A, L, C)$ where A is a set of statements (representing positions one can take or not take in a debate), $L \subseteq A \times A$ is a set of links (representing dependencies between the positions), $C = \{C_a\}_{a \in A}$ is a collection of total functions $C_a : 2^{par(a)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement a. The function C_a is called *acceptance condition of* a and expresses whether a can be accepted, given the acceptance status of its parents $par(a)$, that is, nodes with a direct link to a. In this paper, we represent each C_a by a propositional formula φ_a over $par(a)$. Then, clearly, for $M \subseteq A$ we have $C_a(M \cap par(a)) = \mathbf{t}$ iff M is a model for φ_a, i.e. $M \models \varphi_a$.

[3] introduced a useful subclass of ADFs: an ADF $D = (A, L, C)$ is *bipolar* iff all links in L are supporting or attacking (or both). A link $(b, a) \in L$ is *supporting in*

D iff for all $M \subseteq par(a)$, we have that $C_a(M) = \mathbf{t}$ implies $C_a(M \cup \{b\}) = \mathbf{t}$. Symmetrically, a link $(b, a) \in L$ is *attacking in* D iff for all $M \subseteq par(a)$, we have that $C_a(M \cup \{b\}) = \mathbf{t}$ implies $C_a(M) = \mathbf{t}$. If a link (b, a) is both supporting and attacking then b has no influence on a, the link is redundant (but does not violate bipolarity). We will sometimes use this circumstance when searching for ADFs; there we simply assume that $L = A \times A$, then links that are actually not needed can be expressed by acceptance conditions that make them redundant.

A set $M \subseteq A$ is a *model of* D iff for all $a \in A$ we find that $a \in M$ iff $C_a(M) = \mathbf{t}$. The set of models of D is written as $su(D)$. The size of an ADF D over A is given by $\|D\| = \sum_{a \in A} \|\varphi_a\|$; the size $\|\varphi\|$ of a formula φ is the number of its nodes.

2.3 Logic Programs

For a vocabulary A define $not\ A = \{not\ a \mid a \in A\}$ and the set of literals over A as $A^{\pm} = A \cup not\ A$. A *normal logic program rule* over A is then of the form $a \leftarrow B$ where $a \in A$ and $B \subseteq A^{\pm}$. The set B is called the *body* of the rule, we abbreviate $B^+ = B \cap A$ and $B^- = \{a \in A \mid not\ a \in B\}$. A *logic program (LP)* P over A is a set of logic program rules over A. The body of a rule $a \leftarrow B \in P$ is *satisfied* by a set $M \subseteq A$ iff $B^+ \subseteq M$ and $B^- \cap M = \emptyset$. M is a *supported model* for P iff $M = \{a \in A \mid a \leftarrow B \in P, B \text{ is satisfied by } M\}$. For a logic program P we denote the set of its supported models by $su(P)$. As size measure we define $\|a \leftarrow B\| = |B| + 1$ for rules and $\|P\| = \sum_{r \in P} \|r\|$ for programs.

2.4 Translations between the formalisms

From AFs to BADFs Brewka and Woltran [3] showed how to translate AFs into ADFs: For an AF $F = (A, R)$, define the ADF associated to F as $D_F = (A, R, C)$ with $C = \{\varphi_a\}_{a \in A}$ and $\varphi_a = \bigwedge_{(b,a) \in R} \neg b$ for $a \in A$. The resulting ADF is bipolar since parents are always attacking. Brewka and Woltran [3] proved that this translation is faithful for the AF stable extension and ADF model semantics (Proposition 1). The translation induces at most a linear blowup.

From ADFs to PL Brewka and Woltran [3] also showed that ADFs under supported model semantics can be faithfully translated into propositional logic: when acceptance conditions of statements $a \in A$ are represented by propositional formulas φ_a, then the supported models of an ADF D over A are given by the classical propositional models of the formula set $\Phi_D = \{a \leftrightarrow \varphi_a \mid a \in A\}$.

From AFs to PL In combination, the previous two translations yield a polynomial and faithful translation chain from AFs into propositional logic.

From ADFs to LPs We [5] showed that ADFs can be faithfully translated into normal logic programs. For an ADF $D = (A, L, C)$, its standard LP P_D is given by

$$\{a \leftarrow (M \cup not\ (par(a) \setminus M)) \mid a \in A, C_a(M) = \mathbf{t}\}$$

It is a consequence of Lemma 3.14 in [5] that this translation preserves the supported model semantics. The translation is size-preserving for the acceptance condition representation of [5] via characteristic models; when representing acceptance conditions via propositional formulas, this cannot be guaranteed as we will show later.

From AFs to LPs The translation chain from AFs to ADFs to LPs is compact, and faithful for AF stable semantics and LP supported semantics [5]. It is also size-preserving since the single rule for each atom contains all attackers once.

From LPs to PL It is well-known that logic programs under supported model semantics can be translated to propositional logic [26]. A logic program P becomes the propositional theory $\Phi_P = \{a \leftrightarrow \varphi_a \mid a \in A\}$ where (for $a \in A$)

$$\varphi_a = \bigvee_{a \leftarrow B \in P} \left(\bigwedge_{b \in B^+} b \wedge \bigwedge_{b \in B^-} \neg b \right)$$

From LPs to ADFs The predicate completion of a normal logic program [26] directly yields an equivalent ADF over the same signature [3]. The translation is computable in polynomial time and the blowup (with respect to the original logic program) is at most linear. The translation is faithful for the supported model semantics, which is a consequence of Lemma 3.16 in [5].

2.5 Relative Expressiveness

Given two formalisms $\mathcal{F}_1, \mathcal{F}_2$ with semantics σ_1, σ_2, we say that $\mathcal{F}_2$ under σ_2 is at least as expressive as $\mathcal{F}_1$ under σ_1 if and only if $\sigma_1(\mathcal{F}_1) \subseteq \sigma_2(\mathcal{F}_2)$. This induces a partial order, and its associated strict partial order can be defined as usual. According to this definition, argumentation frameworks (under stable extension semantics) are strictly less expressive than bipolar ADFs (under model semantics), which are in turn strictly less expressive than general ADFs (also under model semantics), normal logic programs (under supported model semantics) and propositional logic [11]. The latter three formalisms are all universally expressive, that is, using vocabulary A they can express any subset $X \subseteq 2^A$. This is relevant for our work since relative succinctness concerns only model sets that both considered formalisms can express.

3 Main Results

3.1 Representing Bipolar Boolean Functions

While bipolarity has hitherto predominantly been defined and used in the context of ADFs [3], it is easy to define the concept for Boolean functions in general. Let A be a set of atoms and $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ be a Boolean function. An atom $a \in A$ is *supporting* iff for all $M \subseteq A$, $f(M) = \mathbf{t}$ implies $f(M \cup \{a\}) = \mathbf{t}$; we then write $a \in \mathit{sup}(f)$. An atom $a \in A$ is *attacking* iff for all $M \subseteq A$, $f(M) = \mathbf{f}$ implies $f(M \cup \{a\}) = \mathbf{f}$; we then write $a \in \mathit{att}(f)$. A Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ is *semantically bipolar* iff each $a \in A$ is supporting or attacking or both.

We will now define bipolar propositional formulas for representing bipolar ADFs. This is important not only for our study, but also since (for three-valued semantics), bipolarity is the key to BADFs' low complexity in comparison to general ADFs [14]. Formally, the *polarity* of an atom $a \in A$ in a formula is determined by the number of negations on the path from the root of the formula tree to the atom. The polarity is *positive* if the number is even and *negative* if the number is odd.

Definition 3. *A propositional formula φ over A is* syntactically bipolar *if and only if no atom $a \in A$ occurs both positively and negatively in φ.*

We will now address the question how to represent bipolar Boolean functions. It is a textbook result that all Boolean functions can be represented by propositional formulas [24,25]. We modify this construction later and thus reproduce it here. For a Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$, its associated formula is

$$\varphi_f = \bigvee_{M \subseteq A, f(M)=\mathbf{t}} \varphi_M \quad \text{with} \quad \varphi_M = \bigwedge_{a \in M} a \wedge \bigwedge_{a \in A \setminus M} \neg a \tag{1}$$

That is, each φ_M has exactly one model M, and φ_f enumerates those models.

So in particular, all bipolar Boolean functions can be represented by propositional formulas as well. However, this only guarantees us the existence of such representations but gives us no way to actually obtain them. Our first fundamental result shows how we can construct a syntactically bipolar propositional formula from a given semantically bipolar Boolean function. The converse is straightforward, and thus the two notions of bipolarity are closely related. For a formula φ, its Boolean function f_φ returns $\mathbf{t}$ iff given a model of φ.

Theorem 1. *Let A be a set of atoms.*

1. *For each syntactically bipolar formula φ over A, its Boolean function f_φ is semantically bipolar.*
2. *For each semantically bipolar Boolean function $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$, there exists a syntactically bipolar formula ψ_f with $f_{\psi_f} = f$.*

Proof. 1. *Obvious: every atom occurring only positively is supporting, every atom occurring only negatively is attacking.*

2. *Let $f : 2^A \to \{\mathbf{t}, \mathbf{f}\}$ be semantically bipolar. We slightly modify construction* (1) *to define ψ_f:*

$$\psi_f = \bigvee_{\substack{M \subseteq A, \\ f(M)=\mathbf{t}}} \psi_M \quad \textit{with} \quad \psi_M = \bigwedge_{\substack{a \in M, \\ a \notin att(f)}} a \wedge \bigwedge_{\substack{a \in A \setminus M, \\ a \notin sup(f)}} \neg a$$

(Note that for any $M \subseteq A$ we have $\models \varphi_M \to \psi_M$.) It is easy to see that ψ_f is syntactically bipolar: Since f is semantically bipolar, each $a \in A$ is: (1) attacking and not supporting, then it occurs only negatively in ψ_f; or (2) supporting and not attacking, then it occurs only positively in ψ_f; or (3) supporting and attacking, then it does not occur in ψ_f. It remains to show that $f_{\psi_f} = f$; we show $\models \varphi_f \equiv \psi_f$.

$\models \varphi_f \to \psi_f$: *Let $v : A \to \{\mathbf{t}, \mathbf{f}\}$ with $v(\varphi_f) = \mathbf{t}$. Then there is an $M \subseteq A$ such that $f(M) = \mathbf{t}$ and $v(\varphi_M) = \mathbf{t}$. (Clearly $v = v_M$.) By $\models \varphi_M \to \psi_M$ we get $v(\psi_M) = \mathbf{t}$ and thus $v(\psi_f) = \mathbf{t}$.*

$\models \psi_f \to \varphi_f$: *For each model v of ψ_f, there is an $M \subseteq A$ with $f(M) = \mathbf{t}$ such that $v(\psi_M) = \mathbf{t}$. To show that each model of ψ_f is a model of φ_f, we show that for all $M \subseteq A$ with $f(M) = \mathbf{t}$, each model v of ψ_M is a model of φ_f. Let $|A| = n$. Then each φ_M contains exactly n literals. For the corresponding*

ψ_M *there is a* $k \in \mathbb{N}$ *with* $0 \leq k \leq n$ *such that* ψ_M *contains exactly* $n - k$ *literals. For two interpretations* $v_1 : A \to \{\mathbf{t}, \mathbf{f}\}$ *and* $v_2 : A \to \{\mathbf{t}, \mathbf{f}\}$*, define the* difference *between them as* $\delta(v_1, v_2) = \{a \in A \mid v_1(a) \neq v_2(a)\}$*. (Note that for* $|A| = n$ *we always have* $|\delta(v_1, v_2)| \leq n$*.) We will use induction on* k *to show the following: for each* $M \subseteq A$ *with* $f(M) = \mathbf{t}$*, each* $v : A \to \{\mathbf{t}, \mathbf{f}\}$ *with* $v(\psi_M) = \mathbf{t}$ *and* $|\delta(v, v_M)| = k$ *we find that* $v(\varphi_f) = \mathbf{t}$*. This covers all models* v *of* ψ_f *(since* $|\delta(v, v_M)| \leq |A|$*) and thus establishes the claim.*

$k = 0$: $\delta(v, v_M) = 0$ *implies* $v = v_M$ *whence* $v(\varphi_f) = v_M(\varphi_f) = v_M(\varphi_M) = \mathbf{t}$ *by definition of* φ_M *and* φ_f*.*

$k \rightsquigarrow k + 1$: *Let* $M \subseteq A$ *with* $f(M) = \mathbf{t}$*, and* $v : A \to \{\mathbf{t}, \mathbf{f}\}$ *with* $v(\psi_M) = \mathbf{t}$ *and* $|\delta(v, v_M)| = k + 1$*. Since* $k + 1 > 0$*, there is some* $a \in \delta(v, v_M)$*, that is, an* $a \in A$ *with* $v(a) \neq v_M(a)$*.*

- *(a)* a *is supporting and not attacking. Then necessarily* $v(a) = \mathbf{t}$*. (If* $v(a) = \mathbf{f}$*, then* $v_M(a) \neq v(a)$ *implies* $v_M(a) = \mathbf{t}$*, that is,* $a \in M$ *whence* $\{\psi_M\} \models a$ *and* $v(\psi_M) = \mathbf{f}$*, contradiction.) Define the interpretation* $w : A \to \{\mathbf{t}, \mathbf{f}\}$ *such that* $w(a) = \mathbf{f}$ *and* $w(c) = v(c)$ *for* $c \in A \setminus \{a\}$*. Clearly* $\delta(v, w) = \{a\}$ *and* $|\delta(w, v_M)| = k$*. Hence the induction hypothesis applies to* w *and* $w(\varphi_f) = \mathbf{t}$*. Now* $w(a) = \mathbf{f}$*,* $v(a) = \mathbf{t}$ *and* $w(\varphi_f) = \mathbf{t}$*. Since* a *is supporting, also* $v(\varphi_f) = \mathbf{t}$*.*
- *(b)* a *is attacking and not supporting. Symmetric to the opposite case above.*
- *(c)* a *is both supporting and attacking. Define the interpretation* $w : A \to \{\mathbf{t}, \mathbf{f}\}$ *such that* $w(a) = v_M(a)$ *and* $w(c) = v(c)$ *for* $c \in A \setminus \{a\}$*. It follows that* $|\delta(w, v_M)| = k$*, whence the induction hypothesis applies to* w *and* $w(\varphi_f) = \mathbf{t}$*. Since* a *is both supporting and attacking, we get that* $v(\varphi_f) = w(\varphi_f) = \mathbf{t}$*.* □

3.2 Relative Succinctness

From the intertranslation results reviewed in the background section, we can infer the following relationships:

$$\text{AF} \leq_s \text{BADF}^{su} \leq_s \text{ADF}^{su} \leq_s \text{PL and LP}^{su} \leq_s \text{ADF}^{su}$$

It is easy to see that AFs have a somewhat special role as they are representationally succinct in any case: for a vocabulary A_n, there is syntactically no possibility to specify a knowledge base (an AF) of exponential size, since the largest AF over A_n has size $\|(A_n, A_n \times A_n)\| = n + n^2$ and is thus polynomially large. So anything that can be expressed with an AF can be expressed in reasonable space by definition. However, this "strength" of AFs should be taken with a grain of salt, since they are comparably inexpressive [11]. This can already be seen from a simple counting argument: even if all syntactically different AFs over A_n were semantically different (which they are not), they could express at most 2^{n^2} different model sets, which is – for increasing n – negligible in relation to the 2^{2^n} possible model sets over A_n.

In contrast, it turns out that ADFs (under the model semantics) are not only as expressive, but also as succinct as propositional logic. This is due to the fact that propositional formulas can be transformed into ADFs (over the same vocabulary) with at most

linear blowup. This next result improves upon our result in [11], where we provided only a semantic realisation – a construction taking a set of models and yielding an ADF of a size that is (while linear in the size of the given set of models) worst-case exponential in the number of statements.

Theorem 2. *PL* $\leq_s$ *ADF*^{su}.

Proof. Let ψ *be a propositional formula over vocabulary* A *with* $|A| = n$*. Define the ADF* D_ψ *over* A *by setting* $\varphi_a = a \leftrightarrow \psi = (a \wedge \psi) \vee (\neg a \wedge \neg\psi)$ *for all* $a \in A$*. Thus* $\|\varphi_a\| \in O(\|\psi\|)$*, whence* $\|D_\psi\| \in O(n\,\|\psi\|)$*. It remains to show* $su(D_\psi) = mod(\psi)$*. Recall that for any ADF* D *over* A*,* $su(D) = mod(\Phi_D)$ *for* $\Phi_D = \bigwedge_{a\in A}(a \leftrightarrow \varphi_a)$*. Applying the definition of* φ_a *in* D_ψ *yields*

$$\Phi_{D_\psi} = \textstyle\bigwedge_{a\in A}(a \leftrightarrow (a \leftrightarrow \psi))$$

Now for any $a \in A$*, the formula* $(a \leftrightarrow (a \leftrightarrow \psi))$ *is equivalent to* ψ*. (The proof is by case distinction on* a*.) Thus* Φ_{D_ψ} *is equivalent to* $\bigwedge_{a\in A}\psi$*, that is, to* ψ*, and it follows that* $su(D_\psi) = mod(\Phi_{D_\psi}) = mod(\psi)$*.* □

For example, consider the vocabulary $A = \{a, b\}$ and the propositional formula $\psi = a \wedge b$. The canonical construction above yields ADF D_ψ with acceptance formulas $\varphi_a = a \leftrightarrow (a \wedge b)$ and $\varphi_b = b \leftrightarrow (a \wedge b)$. Now we have:

$$\varphi_a = a \leftrightarrow (a \wedge b) = (a \rightarrow (a \wedge b)) \wedge ((a \wedge b) \rightarrow a) \equiv \neg a \vee (a \wedge b) \equiv \neg a \vee b$$

Intuitively, $\varphi_a = \neg a \vee b$ expresses that a cannot be false, and is true if b is true. By a symmetrical argument, the acceptance formula of b is equivalent to $\neg b \vee a$. It is readily checked that $su(D_\psi) = \{\{a, b\}\}$ as desired. Since we know that the converse translation is also possible, we get the following.

Corollary 1. *PL* $\cong_s$ *ADF*^{su}

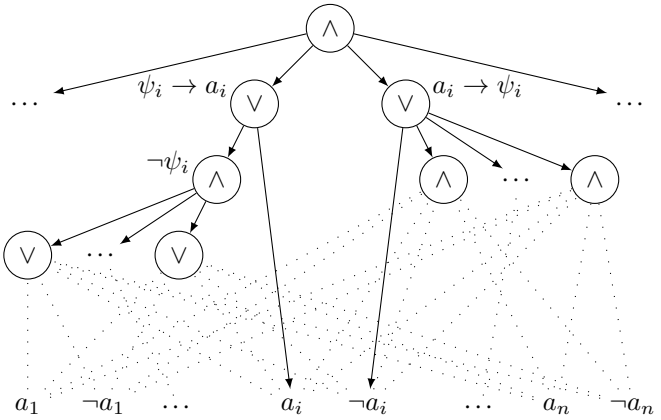
It is quite obvious that the canonical ADF constructed in Theorem 2 is not bipolar, since a as well as every atom mentioned by ψ occurs both positively and negatively in φ_a. From the general construction of Theorem 2 it follows that if ψ has a "small" conjunctive normal form (a conjunction of clauses) and disjunctive normal form (disjunction of monomials) representation, then there is also a "small" normal logic program representation for $mod(\psi)$.

For the opposite direction, it is easy to see that any language that is polynomially expressible by normal logic programs under supported semantics is in AC^0. For the stable semantics of so-called canonical logic programs, this has recently been shown by Shen and Zhao [21] (Proposition 2.1). The case we are interested in (supported semantics) works similarly, but we still present the proof for completeness.

The main technical result towards proving that is a lemma showing how to turn a logic program into an equivalent Boolean circuit of a fixed depth.

Lemma 1. *For every normal logic program* P*, there exists a circuit* C_P *over the basis* $\{\neg, \wedge, \vee\}$ *such that (1)* C_P *accepts all and only the supported models of* P*, (2) the size of* C_P *is linear the size of* P*, (3)* C_P *has depth 4.*

Proof. Let $A = \{a_1, \ldots, a_n\}$ be the vocabulary of P, and its Clark completion be $\Phi_P = \{a_i \leftrightarrow \psi_i \mid a_i \in A\}$ where the ψ_i are DNFs over literals from A. Clearly the circuit for Φ_P must compute $C_P = \bigwedge_{a_i \in A}(a_i \leftrightarrow \psi_i)$ where $a_i \leftrightarrow \psi_i$ can be replaced by $(\neg a_i \vee \psi_i) \wedge (a_i \vee \neg\psi_i)$ with $\neg\psi_i$ a CNF over literals from A. The construction can be depicted as follows, where the inner layers are shown for one i only, and dotted lines represent potential edges.



Now (1) follows since $su(P) = mod(\Phi_P)$ and C_P accepts all and only the models of Φ_P. For (2), if P contains $m = |P|$ rules, then $m \leq \|P\|$ and the total number of inner gates is bounded by $n(2m + 3) \leq n(2 \cdot \|P\| + 3)$. (3) is clear. □

While the statement of Lemma 1 is actually much stronger and gives a *constant* upper bound of the resulting circuit depth for arbitrarily-sized logic programs, it readily follows that the set of polynomially logic-program expressible languages is a subset of the languages expressible by alternating Boolean circuits with unbounded fan-in and constant depth.

Proposition 1. *If L is polynomially expressible by normal logic programs under supported semantics, then $L \in \mathsf{AC}^0$.*

It follows immediately that normal logic programs cannot polynomially express the language PARITY.[2] This is the supported-semantics counterpart of Theorem 3.1 in [21].

Corollary 2. PARITY *has no polynomial size normal logic program representation.*

Proof. By Proposition 1 and PARITY $\notin \mathsf{AC}^0$ *[25].* □

It follows that propositional logic is strictly more succinct than normal logic programs under supported semantics.

Corollary 3. *PL $\not\leq_s$ LP^{su} and thus LP^{su} $<_s$ PL.*

[2] Logic programs under supported models are universally expressive, so they *can* express PARITY, just not in polynomial size.

While PARITY allows us to separate propositional logic from normal logic programs, we cannot use the same language for bipolar ADFs. BADFs cannot even express PARITY, since there is no BADF D over A_3 such that its model set is given by[3]

$$su(D) = \text{PARITY}_3 = \{\{a_1\}, \{a_2\}, \{a_3\}, A_3\}$$

However, the MAJORITY language does the trick in this case.

Theorem 3. *BADF^{su} $\not\leq_s$ LP^{su}*

Proof. We show that the language MAJORITY *can be polynomially expressed by BADF^{su}, but not by LP^{su}. The latter fact follows from* MAJORITY $\notin \mathsf{AC}^0$ *[25] and Proposition 1. We now show the first part by constructing a series of BADFs D_n over $A_n = \{a_1, \ldots, a_n\}$ ($n \in \mathbb{N}$, $n \geq 1$) such that $su(D_n) = \text{MAJORITY}_n$. We use results of [27,28], who show that – for all positive $n \in \mathbb{N}$ and $k \leq n$, the language $\text{THRESHOLD}_{n,k}$ has negation-free propositional formulas $\Phi^{\text{THRESHOLD}}_{n,k}$ of polynomial size s, where we use the bound of Boppana, $s \in O\big(k^{4.27} n \log n\big)$. Define D_1 by $\varphi_{a_1} = \top$, and for $n \geq 2$ set $k = \lceil \frac{n}{2} \rceil$ and for $1 \leq i \leq n$,*

$$\varphi_{a_i} = a_i \vee \neg \Phi^{\text{THRESHOLD}}_{n-1,k}(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$$

Intuitively, the formula φ_{a_i} checks whether the remaining variables could achieve a majority without a_i. If so, then a_i can be set arbitrarily; otherwise, a_i must be set to true. Clearly the Boolean function computed by φ_{a_i} is bipolar, since a_i is supporting and all other parents are attacking. For the size of D_n, we observe that

$$\|D_n\| \in O\big(n \, \big\|\Phi^{\text{THRESHOLD}}_{n-1,k}\big\|\big)$$

whence the overall size is polynomial. It remains to show that $su(D_n) = \text{MAJORITY}_n$.

"$\supseteq$": *Let $M \in \text{MAJORITY}_n$. We have to show $M \in su(D_n)$, that is, $a \in M$ iff $M \models \varphi_a$ for all $a \in A_n$. For $a \in M$, it is immediate that $M \models \varphi_a$, so let $a_j \notin M$ for some $j \in \{1, \ldots, n\}$. We have to show $M \not\models \varphi_{a_j}$. Since $M \in \text{MAJORITY}_n$, we have $|M| = m$ for $k = \lceil \frac{n}{2} \rceil \leq m \leq n-1$ and $M \in \text{THRESHOLD}_{n-1,k}$, that is, we have $M \models \Phi^{\text{THRESHOLD}}_{n-1,k}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n)$. Together with $M \not\models a_j$, it follows that $M \not\models \varphi_{a_j}$.*

"$\subseteq$": *Let $M \notin \text{MAJORITY}_n$. Then $|M| = m$ for $0 \leq m < \lceil \frac{n}{2} \rceil = k$. In particular, there is some $a_j \in A_n \setminus M$. Now $m < k$ implies that there is no $N \in \text{THRESHOLD}_{n-1,k}$ with $|N| = m = |M|$. Thus $M \not\models \Phi^{\text{THRESHOLD}}_{n-1,k}(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n)$ whence it follows that $M \models \varphi_{a_j}$. Together with $M \not\models a_j$ we conclude that $M \notin su(D_n)$.* □

Since every BADF is an ADF of the same size, we get:

Corollary 4. *ADF^{su} $\not\leq_s$ LP^{su}*

In combination with the translation from logic programs to ADFs (implying the relation LP^{su} $\leq_s$ ADF^{su}), this means that also ADFs are strictly more succinct than logic programs.

[3] This can be proven just like Theorem 4 in [11].

	BADF^{su}	ADF^{su}	LP^{su}	PL
BADF^{su}	$=$	$\leq_s$	$\not\leq_s$	$\leq_s$
ADF^{su}	?	$=$	$\not\leq_s$	$\cong_s$
LP^{su}	?	$<_s$	$=$	$<_s$
PL	?	$\cong_s$	$\not\leq_s$	$=$

Table 1: Relative succinctness results for (bipolar) ADFs under the model semantics, normal logic programs under the supported semantics, and classical propositional logic. An entry $\circ$ in row $\mathcal{F}_1$ and column $\mathcal{F}_2$ means $\mathcal{F}_1 \circ \mathcal{F}_2$.

Corollary 5. *LP^{su} $<_s$ ADF^{su}*

In the next and last section, we provide an overview over and discussion of the results obtained in this paper.

4 Overview and Discussion

We analysed the representational capabilities of abstract dialectical frameworks under the two-valued model semantics and compared it to the like capabilities of abstract argumentation frameworks, normal logic programs, and propositional logic. The most significant results are presented at a glance in Table 1. Among other things, we have shown that ADFs (under model semantics) are exponentially more succinct than normal logic programs (under supported model semantics), and that even bipolar ADFs (under model semantics) – although being less expressive – can succinctly express some model sets where equivalent normal logic programs (under supported model semantics) over the same vocabulary must necessarily blow up exponentially in size. The table also shows that we are only "three results away" from having a complete picture. It is easy to show that ADFs and propositional logic behave equivalently in relation to bipolar ADFs, since they are equally expressive and equally succinct; that is, ADF^{su} $\leq_s$ BADF^{su} iff PL $\leq_s$ BADF^{su}. Thus the three open problems in Table 1 are really only two.

Why are these hard problems? Firstly, precisely characterising the expressiveness of BADFs is a hard open problem. While it is clear that BADFs can express arbitrary $\subseteq$-antichains, it is not clear how much more they can express. In other words, a non-trivial characterisation of model sets that are not bipolarly realisable is still missing. Note that "X is not bipolarly realisable" means that *for all* ADFs realising X (there is at least one), there *exists* a statement whose acceptance function is not bipolar. However, this statement need not be the same in each realisation. And while we showed in earlier work that there are model sets that BADFs cannot express, this depended on a computer-aided proof, providing further evidence that BADF expressiveness is a hard problem [11]. Indeed, succinctness results are often "only" conditional, that is, depend on some widely believed complexity-theoretic assumption [22,18,19] (see also Footnote 4 below).

Parts of the expressiveness results for normal logic programs carry over to further LP classes. For example, *canonical logic programs* provide a limited form of nesting by allowing literals of the form $not\ not\ a$ in rule bodies [29]. This makes it quite

easy to see how normal logic programs under supported semantics can be translated to equivalent canonical logic programs, namely by replacing each positive body atom a by $not\ not\ a$ in all rule bodies. Recently, Shen and Zhao [21] showed that canonical logic programs and propositional logic programs are succinctly incomparable (under an assumption[4]), and also provide interesting avenues for further succinctness studies.

In their original paper, Gogic et al. [22] also used a relaxed version of succinctness, where they allowed to introduce a linear number of new variables. For the formalisms we study, adding (even only linearly many) extra variables leads to a collapse of all observed differences (a table full of the linear-blowup version of $\cong_s$), since AFs can equivalently express any propositional formula with at most linear blowup [11]. However, a linear blowup in knowledge base size – from n to cn for a $c \in \mathbb{N}$ with $c > 1$ – leads to a *polynomial* increase in search space size – from 2^n to $2^{cn} = (2^n)^c$.

In future work, we plan on studying further semantics (for example the stable model semantics for normal logic programs and (bipolar) abstract dialectical frameworks), as well as considering further knowledge representation formalisms.

References

1. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence **77** (1995) 321–358
2. Dunne, P.E., Dvořák, W., Linsbichler, T., Woltran, S.: Characteristics of Multiple Viewpoints in Abstract Argumentation. In: Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR), Vienna, Austria (July 2014) 72–81
3. Brewka, G., Woltran, S.: Abstract Dialectical Frameworks. In: Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR). (2010) 102–111
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), IJCAI/AAAI (August 2013) 803–809
5. Strass, H.: Approximating operators and semantics for abstract dialectical frameworks. Artificial Intelligence **205** (December 2013) 39–70
6. Alviano, M., Faber, W.: Stable model semantics of abstract dialectical frameworks revisited: A logic programming perspective. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina (July 2015) In press.
7. Brewka, G., Gordon, T.F.: Carneades and Abstract Dialectical Frameworks: A Reconstruction. In: Computational Models of Argument: Proceedings of COMMA 2010. Volume 216 of Frontiers in Artificial Intelligence and Applications., IOS Press (September 2010) 3–12
8. Strass, H.: Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond. Journal of Logic and Computation (2015) To appear in a special issue on Computational Logic in Multi-Agent Systems (CLIMA XIV).
9. Al-Abdulkarim, L., Atkinson, K., Bench-Capon, T.J.M.: Abstract dialectical frameworks for legal reasoning. In Hoekstra, R., ed.: Proceedings of the Twenty-Seventh Annual Conference on Legal Knowledge and Information Systems (JURIX). Volume 271 of Frontiers in Artificial Intelligence and Applications., IOS Press (December 2014) 61–70

[4] $\mathsf{P} \not\subseteq \mathsf{NC}^1_{/\mathsf{poly}}$, the circuit equivalent of the assumption $\mathsf{NP} \not\subseteq \mathsf{P}$.

10. Al-Abdulkarim, L., Atkinson, K., Bench-Capon, T.J.M.: Evaluating an approach to reasoning with cases using abstract dialectical frameworks. In: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law (ICAIL). (June 2015)
11. Strass, H.: The relative expressiveness of abstract argumentation and logic programming. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI), Austin, TX, USA (January 2015) 1625–1631
12. Bidoit, N., Froidevaux, C.: Negation by default and unstratifiable logic programs. Theoretical Computer Science **78**(1) (1991) 85–112
13. Marek, V.W., Truszczyński, M.: Autoepistemic logic. Journal of the ACM **38**(3) (1991) 587–618
14. Strass, H., Wallner, J.P.: Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. Artificial Intelligence **226**(0) (2015) 34–74
15. Dyrkolbotn, S.K.: How to argue for anything: Enforcing arbitrary sets of labellings using AFs. In: Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR), Vienna, Austria (July 2014) 626–629
16. Baumann, R., Dvořák, W., Linsbichler, T., Strass, H., Woltran, S.: Compact Argumentation Frameworks. In: Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI), Prague, Czech Republic (August 2014) 69–74
17. Puehrer, J.: Realizability of three-valued semantics for abstract dialectical frameworks. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI). IJCAI/AAAI, Buenos Aires, Argentina (July 2015) In press.
18. Darwiche, A., Marquis, P.: A Knowledge Compilation Map. Journal of Artificial Intelligence Research (JAIR) **17** (2002) 229–264
19. Lifschitz, V., Razborov, A.: Why are there so many loop formulas? ACM Transactions on Computational Logic **7**(2) (April 2006) 261–268
20. French, T., van der Hoek, W., Iliev, P., Kooi, B.: On the succinctness of some modal logics. Artificial Intelligence **197** (2013) 56–85
21. Shen, Y., Zhao, X.: Canonical logic programs are succinctly incomparable with propositional formulas. In: Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR), Vienna, Austria (July 2014) 665–668
22. Gogic, G., Kautz, H., Papadimitriou, C., Selman, B.: The comparative linguistics of knowledge representation. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kaufmann (1995) 862–869
23. Brewka, G., Dunne, P.E., Woltran, S.: Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI), IJCAI/AAAI (2011) 780–785
24. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press (2009)
25. Jukna, S.: Boolean Function Complexity: Advances and Frontiers. Volume 27 of Algorithms and Combinatorics. Springer (2012)
26. Clark, K.L.: Negation as Failure. In Gallaire, H., Minker, J., eds.: Logic and Data Bases, Plenum Press (1978) 293–322
27. Friedman, J.: Constructing $O(n \log n)$ size monotone formulae for the k-th elementary symmetric polynomial of n Boolean variables. SIAM Journal on Computing **15** (1986) 641–654
28. Boppana, R.B.: Threshold functions and bounded depth monotone circuits. Journal of Computer and System Sciences **32**(2) (1986) 222–229
29. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence **25**(3–4) (1999) 369–389