Argument and Computation
Vol. 2, Nos. 2–3, June–September 2011, 107–129

Taylor & Francis
Taylor & Francis Group

Complexity of logic-based argumentation in Post's framework[†]

Nadia Creignou[a], Johannes Schmidt[a]*, Michael Thomas[b‡] and Stefan Woltran[c]

[a]*LIF, UMR CNRS 6166, Aix-Marseille Université, 163 Avenue de Luminy, 13288 Marseille Cedex 9, France;* [b]*TWT GmbH, Bernhäuser Straße 40–42, 73765 Neuhausen a.d.F., Germany;* [c]*Institut für Informationssysteme E184/2, Technische Universität Wien, Favoritenstr. 9–11, 1040 Wien, Austria*

(*Received 19 May 2011; final version received 23 September 2011*)

Many proposals for logic-based formalisations of argumentation consider an argument as a pair (Φ, α), where the support Φ is understood as a minimal consistent subset of a given knowledge base which has to entail the claim α. In case the arguments are given in the full language of classical propositional logic reasoning in such frameworks becomes a computationally costly task. For instance, the problem of deciding whether there exists a support for a given claim has been shown to be Σ_2^{p}-complete. In order to better understand the sources of complexity (and to identify tractable fragments), we focus on arguments given over formulæ in which the allowed connectives are taken from certain sets of Boolean functions. We provide a complexity classification for four different decision problems (existence of a support, checking the validity of an argument, relevance and dispensability) with respect to all possible sets of Boolean functions. Moreover, we make use of a general schema to enumerate all arguments to show that certain restricted fragments permit polynomial delay. Finally, we give a classification also in terms of counting complexity.

Keywords: logic based argumentation; complexity; Post's lattice; counting; enumeration

1. Introduction

Argumentation is nowadays a main research topic within the area of Artificial Intelligence (Bench-Capon and Dunne 2007; Besnard and Hunter 2008; Rahwan and Simari 2009) aiming to formally analyse the pros and cons of statements within a certain scenario in order to understand implicit conflicts and to support decision-making. The overall process of formal argumentation can be considered as a sequence of the following steps; see, for instance, the work by Caminada and Amgoud (2007) or Gorogiannis and Hunter (2011): Given a knowledge base Δ, in the first step arguments are formed, and then conflicts between the arguments are identified. Next, one abstracts from the concrete contents of the arguments and uses certain semantics to find acceptable subsets of arguments by analysing solely the graph obtained from the arguments and conflicts; this particular step was first proposed by Dung (1995) who invented the concept of abstract argumentation frameworks. Finally, by inspecting selected arguments certain conclusions can be made. Towards practicably efficient realisations, a good understanding of the computational complexity of all these single steps is thus of high importance.

We focus here on the first step of this process, i.e. forming valid and plausible arguments from a given set of formulæ. This step is also sometimes referred to as logic-based argumentation

*Corresponding author. Email: johannes.schmidt@lif.univ-mrs.fr

[†]An earlier version appeared in the *Proc. of 12th European Conference on Logics in Artificial Intelligence*, JELIA'2010, Helsinki, Finland.

[‡]Work performed while employed at the Institut für Theoretische Informatik, Gottfried Wilhelm Leibniz Universität, Appelstr. 4, 30167 Hannover, Germany.

ISSN 1946-2166 print/ISSN 1946-2174 online

http://dx.doi.org/10.1080/19462166.2011.629736
http://www.tandfonline.com

(Chesñevar, Maguitman, and Loui 2000; Prakken and Vreeswijk 2002; Besnard and Hunter 2008; Amgoud and Besnard 2010) in order to separate it from the abstraction as used in Dung's frameworks. One goal in logic-based argumentation is thus to find a concrete formal representation of an argument and then to define – on top of this concept – notions such as counterarguments, rebuttals or undercuts (see Besnard and Hunter 2001). Many proposals consider an argument as a pair (Φ, α), where the support Φ is a consistent set (or a minimal consistent set) of formulæ from a given knowledge base that entails the claim α which is a formula (Cayrol 1995; Besnard and Hunter 2001; Amgoud and Cayrol 2002; García and Simari 2004; Dung, Kowalski, and Toni 2006). Different logical formalisms provide different definitions for consistency and entailment and hence give different options for defining the notion of an argument. One natural candidate for formalising arguments is the full language of classical propositional logic. However, it is computationally challenging to generate arguments from a knowledge base Δ, using classical logic; in fact, the problem of deciding whether there exists a support $\Phi \subseteq \Delta$ for a given claim α has been shown to be Σ_2^{p}-complete by Parsons, Wooldridge, and Amgoud (2003).

Computing the support for an argument underlies many reasoning problems in logic-based argumentation, for instance, the computation of argument trees as proposed by Besnard and Hunter (2001). Since the basic task of finding a support is already computationally involved, it is indispensable to understand its sources of complexity and to identify fragments for which that problem becomes tractable. In this paper, we contribute to this line of research by restricting the formulæ involved (i.e. formulæ in the knowledge base and thus in the support, as well as the formula used as the claim). In fact, we restrict formulæ to use only connectives from a given set B of Boolean functions and study the decision problems of existence, validity, relevance, and respectively, dispensability, which are defined as follows:

- ARG (Argumentation): given Δ, α, does there exist a support $\Phi \subseteq \Delta$ for α?
- ARG-CHECK (Argument Checking): given a pair (Φ, α), is it an argument?
- ARG-REL (Argument Relevance): given Δ, α, φ, is there an argument (Φ, α) such that $\varphi \in \Phi \subseteq \Delta$?
- ARG-DISP (Argument Dispensability): given Δ, α, φ, is there an argument (Φ, α) such that $\varphi \notin \Phi \subseteq \Delta$?

We understand here as arguments pairs (Φ, α) with *minimal* support, i.e. (Φ, α) is an argument if Φ is consistent, entails α, and no $\Phi' \subsetneq \Phi$ entails α. Moreover, we consider the problems of enumerating and counting arguments:

- #ARG (Argument Counting): given Δ, α, how many arguments (Φ, α) with $\Phi \subseteq \Delta$ exist?
- ENUM-ARG (Argument Enumeration): given Δ, α, output all arguments (Φ, α) such that $\Phi \subseteq \Delta$.

It can be seen that the minimality condition is only important for the decision problems ARG-CHECK and ARG-REL (for instance, in case of ARG-DISP, there exists a support Φ without φ for α exactly if there exists a minimal such support; we will make this more precise in Section 2.3) as well as for #ARG and ENUM-ARG. For the ENUM-ARG problem, we will provide a general scheme to compute all arguments making use of decision procedures for ARG-CHECK, ARG-REL and ARG-DISP. This also indicates the practical relevance of the decision problems we study, since enumerating all arguments is the central task in the overall process discussed above. However, also counting the number of arguments is of importance. Consider two different formulæ α, β, it might be of interest to know whether more arguments of form (Φ, α) than arguments of form (Ψ, β) exist for a given knowledge base. Usually, such counting problems are considered without explicitly making use of enumeration procedures. Note that the latter might have to

output an exponential number of solutions (thus requiring exponential time), although the number of solutions may be efficiently computable. In fact, counting complexity recently gained interest in AI as is witnessed by work from Hermann and Pichler (2010) or Durand and Hermann (2008). However, except recent work by Barnoi, Dunne, and Giacomin (2010), who consider counting complexity for abstract argumentation, we are not aware of such investigations in the area of argumentation.

A major tool when analysing the complexity of problems parameterised by a given set B of Boolean connectives is *Post's lattice*: a lattice showing the inclusion relations between all existing sets of Boolean functions that are closed under superposition (that is, roughly speaking, closed under arbitrary composition, see Section 2.2 for a formal definition). Several complexity classifications have already been obtained in this so-called *Post's framework*, such as the classical satisfiability problem (Lewis 1979), equivalence and implication problems (Reith 2003; Beyersdorff, Meier, Thomas, and Vollmer 2009a), satisfiability and model checking in modal and temporal logics (Bauland, Hemaspaandra, Schnoor, and Schnoor 2006; Bauland, Schneider, Schnoor, Schnoor, and Vollmer 2008), default logic (Beyersdorff, Meier, Thomas, and Vollmer 2009b), circumscription (Thomas 2009) and abduction (Creignou, Schmidt, and Thomas 2010).

The main contribution of this paper is a systematic complexity classification for the six aforementioned problems in terms of all possible sets of Boolean connectives.

- Concerning the four decision problems, we show that, depending on the chosen set of connectives, the problems range from inside P up to the second level of the polynomial hierarchy, and we identify those fragments complete for NP, coNP, and also for DP, the class of differences of problems in NP. We also show that unless the polynomial hierarchy collapses there exist particular sets of Boolean connectives such that: (i) deciding the existence of an argument is easier than verifying a given one; (ii) deciding the dispensability of a formula for some argument is easier than deciding its relevance.
- We provide a general schema for enumerating all arguments. For the fragments which have their decision problems in P, we also obtain positive results in terms of enumeration by showing that either each solution is computed with at most polynomial delay (for the fragments where the number of arguments might be exponential) or that all solutions can be computed in polynomial time (which obviously is only possible for fragments where the number of arguments is bounded polynomially in the size of the knowledge base and the claim). Finally, we complement our picture by considering the problem of counting the number of arguments in terms of all possible sets of Boolean connectives.

The paper is structured as follows. Section 2 contains the necessary background on complexity theory (Section 2.1) and on Post's lattice (Section 2.2). Moreover, we define the studied framework of argumentation and relevant problems in Section 2.3. We then turn to our main results which are given in Sections 3 and 4. More specifically, the complexity of the decision problems is classified in Sections 3.2–3.4, enumeration is studied in Section 4.1 and counting complexity is dealt with in Section 4.2. We summarise our results and discuss their relation to similar work (as e.g. in the area of abduction) in a dedicated Section 5. Finally, Section 6 concludes with a brief recapitulation of the achieved results as well as future research directions.

2. Background

We assume familiarity with propositional logic. The set of all propositional formulæ is denoted by $\mathcal{L}$. A *model* for a formula φ is a truth assignment to the set of its variables that satisfies φ. Further, we denote by $\varphi[\alpha/\beta]$ the formula obtained from φ by replacing all occurrences of α with β.

For a given formula φ, we denote by $\mathrm{Vars}(\varphi)$ the set of variables occurring in φ. We extend this definition on sets of formulæ Γ as $\mathrm{Vars}(\Gamma) = \bigcup_{\varphi\in\Gamma} \mathrm{Vars}(\varphi)$. We identify finite Γ with the conjunction of all the formulæ in Γ, $\bigwedge_{\varphi\in\Gamma} \varphi$. Naturally, $\Gamma[\alpha/\beta]$ then stands for $\bigwedge_{\varphi\in\Gamma} \varphi[\alpha/\beta]$. We say that two formulæ φ and ψ are *equivalent* (written $\varphi \equiv \psi$) if every assignment $\sigma : \mathrm{Vars}(\varphi) \cup \mathrm{Vars}(\psi) \to \{0,1\}$ on the combined variable sets satisfies φ if and only if it satisfies ψ. For any formula $\varphi \in \mathcal{L}$, we write $\Gamma \models \varphi$ if Γ entails φ, i.e. if φ is satisfied in every model of Γ.

A *literal* l is a variable x or its negation $\neg x$. A *positive* literal is a variable x, a *negative* literal is the negation of a variable $\neg x$. Given a set of variables V, $\mathrm{Lits}(V)$ denotes the set of all literals formed upon the variables in V, i.e. $\mathrm{Lits}(V) := V \cup \{\neg x \mid x \in V\}$. A *clause* is a disjunction of literals and a *term* is a conjunction of literals.

2.1. *Complexity theory*

We require standard notions of the complexity theory. For the decision problems, the arising complexity degrees encompass the classes LOGSPACE, P, NP, coNP, DP and Σ_2^{p}. The class DP is defined as the set of languages recognisable by the difference of two languages in NP, i.e. $\mathrm{DP} := \{L_1 \setminus L_2 \mid L_1, L_2 \in \mathrm{NP}\} = \{L_1 \cap L_2 \mid L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP}\}$. The class Σ_2^{p} is the set of languages recognisable by non-deterministic polynomial-time Turing machines with an NP oracle. For our hardness results, we employ *logspace many-one reductions*, defined as follows: a language A is logspace many-one reducible to some language B (written $A \leq_{\mathrm{m}}^{\log} B$) if there exists a logspace-computable function f such that $x \in A$ if and only if $f(x) \in B$. Thus, for $\mathcal{C}$ being any of the complexity class NP, coNP, DP or Σ_2^{p}, we call *$\mathcal{C}$-hard* a problem G if any instance of a generic problem in $\mathcal{C}$ can be reduced to an instance of G by means of a logspace many-one reduction. If in addition G belongs to the class $\mathcal{C}$, then it is called *$\mathcal{C}$-complete*. A complete problem for DP is CRITICAL-SAT, the problem to decide whether a given formula in 3CNF is unsatisfiable, but removing any of its clauses makes it satisfiable (Papadimitriou and Wolfe 1988). A prototypical Σ_2^{p}-complete problem is deciding the truth of an expression $\exists X \forall Y \beta$ where β is a propositional formula over the set of variables $X \cup Y$. This quantified formula is valid if and only if there exists a truth assignment to the variables of X such that for all truth assignments to the variables of Y the formula β is true. In this paper, we use a more restricted version, that we denote by $\mathrm{Qsat}_{2,\exists}$, in which the formula β is 3-DNF, and which is still Σ_2^{p}-complete.

When analysing an *enumeration algorithm*, polynomial time complexity is not a suitable yardstick of efficiency since the output size is usually exponential in the size of the input. We consider an enumeration algorithm to be efficient, when it runs in *polynomial delay*, i.e. if the delay until the first solution is output, thereafter the delay between any two consecutive solutions, and the delay between the last solution and termination is bounded by a polynomial $p(n)$ in the input size n. This notion of efficiency for enumeration has first been introduced by Johnson, Papadimitriou, and Yannakakis (1988). We denote the corresponding complexity class by DelayP.

A *counting problem* is represented using a *witness function* w, which for every input x returns a finite set of witnesses. This witness function gives rise to the following counting problem: given an instance x, find the cardinality $|w(x)|$ of the witness set $w(x)$. The class #P is the class of counting problems naturally associated with decision problems in NP. According to Hemaspaandra and Vollmer (1995), if $\mathcal{C}$ is a complexity class of decision problems, we define $\#\cdot\mathcal{C}$ to be the class of all counting problems whose witness function is such that the size of every witness y of x is polynomially bounded in the size of x, and checking whether $y \in w(x)$ is in $\mathcal{C}$. Thus, we have $\#\mathrm{P} = \#\cdot\mathrm{P}$ and $\#\mathrm{P} \subseteq \#\cdot\mathrm{NP}$. Completeness of counting problems is usually proved by means of *Turing reductions*. A stronger notion is the *parsimonious reduction* (Papadimitriou 1994), where the exact number of solutions is conserved by the reduction function. When there is a parsimonious reduction from a counting problem $\#A$ to a counting problem $\#B$ we write $\#A \leq_! \#B$.

For more background information on complexity theory, the reader is referred to Papadimitriou (1994).

2.2. *Post's lattice*

A *Boolean function* is an n-ary function $f : \{0,1\}^n \to \{0,1\}$. A *clone* is a set of Boolean functions that is closed under superposition, i.e. it contains all projections (that is, the functions $f(a_1,\ldots,a_n) = a_k$ for all $1 \leq k \leq n$ and $n \in \mathbb{N}$) and is closed under arbitrary composition (that is, application of one function to the results of other functions). Let B be a finite set of Boolean functions. We denote by $[B]$ the smallest clone containing B and call B a *base* for $[B]$. Post (1941) identified, the set of all clones of Boolean functions. He gave a finite base for each of the clones and showed that they form a lattice under the usual $\subseteq$-relation, hence the name *Post's lattice* (Figure 1). To define the clones, we introduce the following notions, where f is an n-ary Boolean function:

- f is *c-reproducing* if $f(c,\ldots,c) = c, c \in \{0,1\}$. The binary *and* ($\wedge$) and the binary *or* ($\vee$) are 0- and 1-reproducing, the binary exclusive or ($\oplus$) is 0-reproducing, but not 1-reproducing, whereas the unary negation ($\neg$) is neither 1- nor 0-reproducing.
- f is *monotonic* if $a_1 \leq b_1,\ldots,a_n \leq b_n$ implies $f(a_1,\ldots,a_n) \leq f(b_1,\ldots,b_n)$. Boolean functions built up on composition of only $\wedge, \vee, 0, 1$ are monotonic, like for instance $g(x,y,z) \equiv x \wedge (1 \wedge (y \vee z))$.
- f is *c-separating of degree k* if for all $A \subseteq f^{-1}(c)$ of size $|A| = k$ there exists an $i \in \{1,\ldots,n\}$ such that $(a_1,\ldots,a_n) \in A$ implies $a_i = c$, $c \in \{0,1\}$. The $(m+1)$-ary function $h_m(x_1,\ldots,x_{m+1}) \equiv \bigvee_{i=1}^{m+1} \bigwedge_{j=1,j\neq i}^{m+1} x_j$ being true if and only if 'at least m variables are true' is 1-separating of degree m. For instance $h_2(x,y,z) \equiv (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ is 1-separating of degree 2.
- f is *c-separating* if f is c-separating of degree $|f^{-1}(c)|$. The implication $g(x,y) \equiv \neg x \vee y$ is 0-separating.
- f is *self-dual* if $f \equiv \text{dual}(f)$, where $\text{dual}(f)(x_1,\ldots,x_n) := \neg f(\neg x_1,\ldots,\neg x_n)$. The function $g(x,y,z) \equiv (x \wedge \neg y) \vee (x \wedge \neg z) \vee (\neg y \wedge \neg z)$ is self-dual.
- f is *affine* if $f \equiv x_1 \oplus \cdots \oplus x_n \oplus c$ with $c \in \{0,1\}$. The function $g(x,y,z) \equiv x \oplus y \oplus z \oplus 1$ is affine and self-dual.

A list of all clones with definitions and finite bases is given in Table 1. In the naming of the clones, the semantic of single indexes is as follows. Index 2 indicates that the clone contains no constants at all. Index 0 (resp. 1) indicates that the clone contains only the constant 0 (resp. 1), but not 1 (resp. 0). Clones with no index contain both constants 0 and 1. The only exceptions to this convention are the clones D and D_1 which do not contain any constants at all. The index $*$ stands for all valid indexes. Clones of particular importance in this paper, since among other things they mark points in Post's lattice where the complexity of argumentation changes, are:

- the clone of all Boolean functions $\mathsf{BF} = [\wedge, \neg] = [\wedge, \vee, \neg, 0, 1]$
- the monotonic clones M_*, e.g. $\mathsf{M}_2 = [\wedge, \vee]$, $\mathsf{M} = [\wedge, \vee, 0, 1]$
- the affine clones L_*, e.g. $\mathsf{L}_2 = [x \oplus y \oplus z]$, $\mathsf{L} = [x \oplus y, 0, 1]$
- the disjunctive clones V_*, e.g. $\mathsf{V}_2 = [\vee]$, $\mathsf{V} = [\vee, 0, 1]$
- the conjunctive clones E_*, e.g. $\mathsf{E}_2 = [\wedge]$, $\mathsf{E} = [\wedge, 0, 1]$
- the c-reproducing clones R_*, R_1 1-repr., R_0 0-repr., R_2 1- and 0-repr.
- the implication clone $\mathsf{S}_0 = [\rightarrow]$
- the negated-implication clone $\mathsf{S}_1 = [x \wedge \neg y]$
- the dual clones D_*, D self-dual, $\mathsf{D}_1 = \mathsf{D} \cap \mathsf{R}_2$, $\mathsf{D}_2 = \mathsf{D} \cap \mathsf{M}$

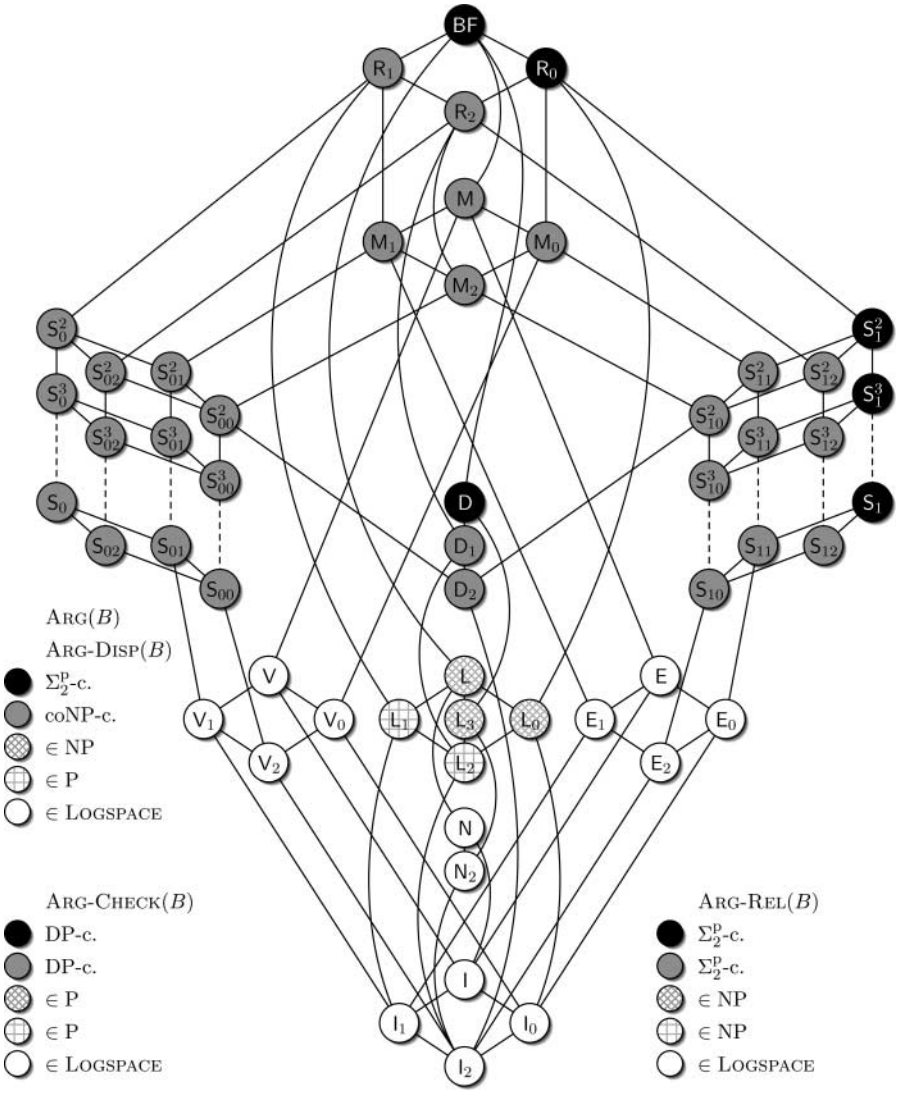

Figure 1. Post's lattice showing the complexity of the decision problems studied herein.

- the clones $\mathsf{S}_{00} = \mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M} = [x \vee (y \wedge z)]$, $\mathsf{S}_{10} = \mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M} = [x \wedge (y \vee z)]$ and $\mathsf{S}_{12} = \mathsf{S}_1 \cap \mathsf{R}_2 = [x \wedge (y \vee \neg z)]$

We will often add some function $f \notin C$ to a clone C and consider the clone $C' = [C \cup \{f\}]$ generated out of C and f. With Post's lattice, one can quite easily determine this C': it is the lowest clone above C that contains f. The following identities will be frequently used.

- $[\mathsf{S}_{10} \cup \{1\}] = \mathsf{M}_1$
- $[\mathsf{D}_2 \cup \{1\}] = \mathsf{S}^2_{01}$
- $[\mathsf{S}_1 \cup \{1\}] = [\mathsf{D} \cup \{1\}] = \mathsf{BF}$

For an example on how these identities will be useful, see Section 3.1, in particular at the end.

Table 1. The list of all Boolean clones with definitions and bases, where $h_n := \bigvee_{i=1}^{n+1} \bigwedge_{j=1, j\neq i}^{n+1} x_j$ and $\mathrm{dual}(f)(a_1, \ldots, a_n) = \neg f(\neg a_1 \ldots, \neg a_n)$.

Name	Definition	Base
BF	All Boolean functions	$\{x \wedge y, \neg x\}$
R_0	$\{f \mid f$ is 0-reproducing$\}$	$\{x \wedge y, x \oplus y\}$
R_1	$\{f \mid f$ is 1-reproducing$\}$	$\{x \vee y, x \oplus y \oplus 1\}$
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{\vee, x \wedge (y \oplus z \oplus 1)\}$
M	$\{f \mid f$ is monotonic$\}$	$\{x \vee y, x \wedge y, 0, 1\}$
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \vee y, x \wedge y, 1\}$
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \vee y, x \wedge y, 0\}$
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \vee y, x \wedge y\}$
S_0^n	$\{f \mid f$ is 0-separating of degree $n\}$	$\{x \to y, \mathrm{dual}(h_n)\}$
S_0	$\{f \mid f$ is 0-separating$\}$	$\{x \to y\}$
S_1^n	$\{f \mid f$ is 1-separating of degree $n\}$	$\{x \wedge \neg y, h_n\}$
S_1	$\{f \mid f$ is 1-separating$\}$	$\{x \wedge \neg y\}$
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z), \mathrm{dual}(h_n)\}$
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z)\}$
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\{\mathrm{dual}(h_n), 1\}$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), 1\}$
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(h_n)\}$
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z), h_n\}$
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z)\}$
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\{h_n, 0\}$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), 0\}$
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z), h_n\}$
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$
D	$\{f \mid f$ is self-dual$\}$	$\{(x \wedge \neg y) \vee (x \wedge \neg z) \vee (\neg y \wedge \neg z)\}$
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{(x \wedge y) \vee (x \wedge \neg z) \vee (y \wedge \neg z)\}$
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)\}$
L	$\{f \mid f$ is affine$\}$	$\{x \oplus y, 1\}$
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \oplus y \oplus 1\}$
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus 1\}$
V	$\{f \mid f$ is a disjunction of variables or constants$\}$	$\{x \vee y, 0, 1\}$
V_0	$\mathsf{V} \cap \mathsf{R}_0$	$\{x \vee y, 0\}$
V_1	$\mathsf{V} \cap \mathsf{R}_1$	$\{x \vee y, 1\}$
V_2	$\mathsf{V} \cap \mathsf{R}_2$	$\{x \vee y\}$
E	$\{f \mid f$ is a conjunction of variables or constants$\}$	$\{x \wedge y, 0, 1\}$
E_0	$\mathsf{E} \cap \mathsf{R}_0$	$\{x \wedge y, 0\}$
E_1	$\mathsf{E} \cap \mathsf{R}_1$	$\{x \wedge y, 1\}$
E_2	$\mathsf{E} \cap \mathsf{R}_2$	$\{x \wedge y\}$
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg x, 0, 1\}$
N_2	$\mathsf{N} \cap \mathsf{R}_2$	$\{\neg x\}$
I	$\{f \mid f$ is a projection or a constant$\}$	$\{\mathrm{id}, 0, 1\}$
I_0	$\mathsf{I} \cap \mathsf{R}_0$	$\{\mathrm{id}, 0\}$
I_1	$\mathsf{I} \cap \mathsf{R}_1$	$\{\mathrm{id}, 1\}$
I_2	$\mathsf{I} \cap \mathsf{R}_2$	$\{\mathrm{id}\}$

A propositional formula using only functions from B as connectives is called a *B-formula*. The set of all B-formulæ is denoted by $\mathcal{L}(B)$.

Let f be an n-ary Boolean function. A B-formula φ is called *B-representation* of f if $f \equiv \varphi$. Such a B-representation exists for every $f \in [B]$. Yet, it may happen that the B-representation of some function uses some input variable more than once, see Example 2.1.

Example 2.1 (*Exponential blow up*). Let $h(x, y) \equiv \neg(x \wedge y)$. An $\{h\}$-representation of the binary *and*, $and(x, y) \equiv x \wedge y$, is $h(h(x, y), h(x, y))$. Observe that an $\{h\}$-representation of the n-ary *and*, $and_n(x_1, \ldots, x_n) \equiv x_1 \wedge \cdots \wedge x_n$, based on the recursive application of the $\{h\}$-representation $h(h(x, y), h(x, y))$ of the binary *and* to the formula

$$(\cdots(((x_1 \wedge x_2) \wedge x_3) \wedge x_4) \wedge \cdots) \wedge x_n$$

leads to an explosion of the formula size. This is because the parentheses-depth is linear and the variables x, y appear twice in the $\{h\}$-representation $h(h(x, y), h(x, y))$ of the binary *and*. We can avoid this exponential blow up by placing the parentheses in a way such that we get a formula of logarithmic parentheses-depth, i.e.

$$(\cdots((x_1 \wedge x_2) \wedge (x_3 \wedge x_4)) \wedge \cdots) \wedge (\cdots \wedge ((x_{n-3} \wedge x_{n-2}) \wedge (x_{n-1} \wedge x_n)) \cdots).$$

2.3. *Logic-based argumentation*

Throughout the paper, Δ is assumed to be a given finite set of formulæ (the knowledge base) representing a large depositary of information, from which arguments can be constructed for arbitrary claims.

Following Besnard and Hunter (2001), an argument is a pair (Φ, α), where Φ is a set of formulæ and α is a formula such that

(1) Φ is consistent,
(2) $\Phi \models \alpha$,
(3) Φ is minimal with this last property, i.e. no proper subset of Φ entails α.

We say that (Φ, α) is an *argument* for α. If $\Phi \subseteq \Delta$, then it is said to be an *argument in* Δ. We call α the *consequent* and Φ the *support* of the argument. Note that these three conditions are equivalent to the following:

(C1) Φ is satisfiable,
(C2) $\Phi \wedge \neg\alpha$ is unsatisfiable (i.e. $\Phi \models \alpha$), and
(C3) for all $\varphi \in \Phi$, $(\Phi \setminus \{\varphi\}) \cup \{\neg\alpha\}$ is satisfiable (i.e. Φ is minimal).

Let B be a finite set of Boolean functions. Then the *argument existence problem for B-formulæ* is defined as

Problem: ARG(B).
Instance: $\mathcal{A} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Question: Does there exist Φ such that (Φ, α) is an argument in Δ?

Besides the decision problem for the existence of an argument, we are interested in the decision problems for B-formulæ for validity, relevance and dispensability. They are defined as follows and deal with formulæ in $\mathcal{L}(B)$ only.

Problem: ARG-CHECK(B).
Instance: $\mathcal{A} = (\Phi, \alpha)$, where $\Phi \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Question: Is (Φ, α) an argument?

Problem: ARG-REL(B).
Instance: $\mathcal{A} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha, \varphi \in \mathcal{L}(B)$.
Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \in \Phi$?

Problem: ARG-DISP(B).

Instance: $\mathcal{A} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha, \varphi \in \mathcal{L}(B)$.
Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \notin \Phi$?

Observe that the minimality of the support is only relevant to the problems ARG-CHECK and ARG-REL. For ARG and ARG-DISP, the existence of a consistent subset Φ of the knowledge base Δ that entails the claim α (and does not contain some formula φ) implies a consistent $\Phi' \subseteq \Phi$ such that $\Phi' \models \alpha$ and $\Phi' \setminus \{\psi\} \not\models \alpha$ for all $\psi \in \Phi'$. To decide the existence of an argument, it therefore suffices to find any consistent subset of Δ that entails α. For ARG-REL, on the other hand, we have to decide whether there exists an argument for α that contains the formula φ. The existence of some consistent set $\Phi \subseteq \Delta$ with $\varphi \in \Phi$ and $\Phi \models \alpha$ does not help here, because φ might be excluded from the minimal subset $\Phi' \subseteq \Phi$ yielding an argument for α. Consequently, unlike in other non-monotonic reasoning formalisms, the complexity of deciding relevance and dispensability of a formula for some argument may differ. Indeed, we will show that there exist sets B such that ARG-REL(B) is harder to decide than ARG-DISP(B) unless the polynomial hierarchy collapses. Similarly, for ARG-CHECK, we have to verify that the set Φ in the given pair (Φ, α) is indeed minimal with respect to consistency and entailment of α. While this is supposedly easier to decide than ARG, we will see that owing to the verification of minimality there exist sets B such that ARG-CHECK(B) is harder to decide than ARG(B) unless the polynomial hierarchy collapses.

Other interesting tasks are to enumerate all solutions or to count them.

Problem: ENUM-ARG(B).
Instance: $\mathcal{A} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Output: Generate all arguments (Φ, α) in Δ without repetition.

Problem: #ARG(B).
Instance: $\mathcal{A} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Output: Number of arguments (Φ, α) in Δ.

Obviously, for both ENUM-ARG(B) and #ARG(B), minimality of the support plays a crucial role, as well.

3. Decision problems

3.1. *Tools and methods*

Observe that if B_1 and B_2 are two sets of Boolean functions such that $B_1 \subseteq [B_2]$, then every function of B_1 can be expressed by a B_2-formula, namely by its B_2-*representation*. This way there is a canonical reduction between ARG($B_1, \mathcal{M}$) and ARG($B_2, \mathcal{M}$) if $B_1 \subseteq [B_2]$: replace all B_1-connectives by their B_2-representation. This reduction is not necessarily polynomial: Since the B_2-representation of some function may use some input variable more than once, the formula size may grow exponentially, see Example 2.1. (The existence or not of a polynomial-size B_2-representation for any B_1-formula is a topic of independent interest, which has been addressed several times in the literature, see e.g. Spira 1971; Karchmer and Wigderson 1988). Nevertheless, we will use the idea of this canonical reduction very frequently, dealing only with cases where exponential blow up can be avoided by special structures of the B_1-formulæ, similar to Example 2.1.

When we show hardness-results for ARG(B) for some B such that $C \subseteq [B]$ (C a clone), we generally show hardness first only for ARG(C) and show then in a second step that the proof can indeed be extended to show hardness also for ARG(B), using the above-mentioned canonical reduction.

The following lemmas show that we can often suppose that our considered B contains the constant 1. This will reduce the number of cases to study.

LEMMA 3.1 Let ARG-$\mathfrak{P}$ denote any of the problems ARG, ARG-CHECK or ARG-REL. Let B be a finite set of Boolean functions such that $\wedge \in [B]$, i.e. $\mathsf{E}_2 \subseteq [B]$. Then ARG-$\mathfrak{P}(B \cup \{1\}) \leq_{\mathrm{m}}^{\log}$ ARG-$\mathfrak{P}(B)$.

Proof Let $\mathcal{I}$ be the given instance. We map $\mathcal{I}$ to the instance $\mathcal{I}'$ obtained by replacing each formula ψ occurring in $\mathcal{I}$ by $\psi[1/t] \wedge t$, where t is said to be a fresh variable. ■

In addition on this, one can also eliminate the constant 1 for the problems ARG(B) and ARG-REL(B) when $\mathsf{D}_2 \subseteq [B]$.

LEMMA 3.2 Let B be a finite set of Boolean functions such that $\mathsf{D}_2 \subseteq [B]$. Then ARG($B \cup \{1\}$) $\leq_{\mathrm{m}}^{\log}$ ARG(B) and ARG-REL($B \cup \{1\}$) $\leq_{\mathrm{m}}^{\log}$ ARG-REL(B).

Proof Let $g(x, y, z) := (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. The function g is a base of D_2 and evaluates to true if and only if at least two of the variables are set to true. Given an instance (Δ, α) of ARG($B \cup \{1\}$), we define an instance (Δ', α') of ARG(B) by $\Delta' := \{\psi[1/t] \mid \psi \in \Delta\} \cup \{t\}$ and $\alpha' = g(\alpha[1/t], t, q)$,where t and q are fresh variables. We claim that there is an argument for α in Δ if and only if there is an argument for α' in Δ'.

Let Φ be an argument for α in Δ. Consider $\Phi' := \{\psi[1/t] \mid \psi \in \Phi\} \cup \{t\}$. Observe that $\Phi' \equiv \Phi$. Thus Φ' is satisfiable and $\Phi' \models \alpha$, hence $\Phi' \models \alpha[1/t] \wedge t$, as t does not occur in α. Therefore, we obtain $\Phi' \models g(\alpha[1/t], t, q)$. Moreover, either Φ' or $\Phi' \setminus \{t\}$ is minimal with this property. Indeed, suppose that there exists a $\psi' \in \Phi'$ with $\psi' = \psi[1/t]$ for some $\psi \in \Phi$ such that $\Phi' \setminus \{\psi'\} \models g(\alpha[1/t], t, q)$. Then $\Phi' \setminus \{\psi'\} \models \alpha[1/t] \wedge t$ as q does not occur in Φ', and hence $\Phi \setminus \{\psi\} \models \alpha$, contradictory to the minimality of Φ.

Conversely, with similar arguments, it is easy to see that if Φ' is an argument for α' in Δ', then $\Phi := \{\psi[t/1] \mid \psi \in \Phi', \psi \neq t\}$ is an argument for α in Δ: as q does not occur in Φ', $\Phi' \models \alpha'$ implies that $\Phi' \models \alpha[1/t] \wedge t$.

This proves a correctness of the reduction from ARG($B \cup \{1\}$) to ARG(B). The analogous result for ARG-REL follows from the same arguments as above, mapping the additional component φ to $\varphi' := \varphi[1/t]$. ■

Remark 1 Observe that this reduction does not work for ARG − CHECK : one would have to decide whether to map Φ to Φ' or to $\Phi' \setminus \{t\}$ to ensure minimality, which requires the ability to decide whether $\Phi' \setminus \{t\} \models t$ in LOGSPACE. Further observe that an analogous statement of Lemma 3.1 for the constant 0 cannot be obtained in the obvious way. Replacing every formula ψ by $\psi[0/f] \vee f$ for a fresh variable f fails since such a reduction does not preserve consistency.

Let us show in an example how one can use these lemmas. Suppose, we want to show hardness results for ARG-REL(B) in the case where $\mathsf{S}_{00} \subseteq [B]$ or $\mathsf{D}_2 \subseteq [B]$ or $\mathsf{S}_{10} \subseteq [B]$ (see Theorem 3.9). Since in the latter two cases either $\mathsf{D}_2 \subseteq [B]$ or $\mathsf{E}_2 \subseteq [B]$, according to Lemmas 3.1 and 3.2 we have ARG-REL($B \cup \{1\}$) $\leq_{\mathrm{m}}^{\log}$ ARG-REL(B). Therefore, it is sufficient to prove hardness for ARG-REL($B \cup \{1\}$). Observe that if $\mathsf{D}_2 \subseteq [B]$ then $\mathsf{S}_{01}^2 = [\mathsf{D}_2 \cup \{1\}] \subseteq [[B] \cup \{1\}] = [B \cup \{1\}]$ and if $\mathsf{S}_{10} \subseteq [B]$ then $\mathsf{M}_1 = [\mathsf{S}_{10} \cup \{1\}] \subseteq [B \cup \{1\}]$. Since $\mathsf{S}_{00} \subseteq \mathsf{S}_{01}^2$ and $\mathsf{S}_{00} \subseteq \mathsf{M}_1$, we have in both cases $\mathsf{S}_{00} \subseteq [B \cup \{1\}]$. So, finally in order to prove that ARG-REL(B) is hard in the case where $\mathsf{S}_{00} \subseteq [B]$ or $\mathsf{D}_2 \subseteq [B]$ or $\mathsf{S}_{10} \subseteq [B]$, it is sufficient to prove that the hardness holds for B such that $\mathsf{S}_{00} \subseteq [B]$.

We will henceforth give less details when applying Lemmas 3.1 and 3.2.

3.2. *The complexity of verification*

We commence our study of the introduced argumentation problems with the argument verification problem. This problem is in DP. Indeed, it is readily observed, as there are languages A, B with $A \in \text{NP}$ and $B \in \text{coNP}$ such that $\text{ARG} - \text{CHECK} = A \cap B$, with

$$A = \{(\Delta, \Phi, \alpha) \mid \Phi \text{ is satisfiable}, \quad \forall \varphi \in \Phi : \Phi \setminus \{\varphi\} \not\models \alpha\};$$
$$B = \{(\Delta, \Phi, \alpha) \mid \Phi \models \alpha\}.$$

The next two results will cover all cases where we have a matching lower bound, i.e. we provide those clones for which DP-hardness holds.

PROPOSITION 3.3 *Let* $\mathsf{S}_{00} \subseteq [B]$. *Then* ARG-CHECK($B$) *is* DP-*complete.*

Proof To prove DP-hardness, we establish a reduction from $\text{CRITICAL} - \text{SAT}$. Let $\psi = \bigwedge_{j=1}^{m} C_j$ be an instance of $\text{CRITICAL} - \text{SAT}$, and $\text{Vars}(\psi) = \{x_1, \ldots, x_n\}$. Let $u, x'_1, \ldots, x'_n$ be fresh, pairwise distinct variables. Note that if $\psi \in \text{CRITICAL} - \text{SAT}$ then each variable of ψ appears both as positive and as negative literal. Let further $C'_j := C_j[\neg x_i / x'_i \mid 1 \leq i \leq n]$ for $1 \leq j \leq m$ and $\psi' := \bigwedge_{j=1}^{m} C'_j$. We map ψ to (Φ, α), where we define

$$\Phi = \{C'_j | 1 \leq j \leq m\} \quad \text{and}$$
$$\alpha = \bigvee_{i=1}^{n} u \vee (x_i \wedge x'_i).$$

Since $x \vee y$ and $x \vee (y \wedge z)$ are functions of S_{00}, α and all C'_j's are S_{00}-formulæ. These are by definition 1-reproducing. Therefore, Φ and α are satisfiable. For $1 \leq k \leq m$, let Φ_k, ψ_k, ψ'_k denote the respective set of clauses where we deleted the kth clause. Note that always $\Phi \equiv \psi'$ and $\Phi_k \equiv \psi'_k$.

Suppose now that $\psi \in \text{CRITICAL} - \text{SAT}$, i.e. ψ is unsatisfiable and ψ_k is satisfiable for all $k \in \{1, \ldots, m\}$. We show that Φ entails α. Since $\psi \equiv \psi' \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i)$ is unsatisfiable, and $\psi' \equiv \Phi$ is monotonic, all models of Φ have to set both x_i and x'_i to 1 for at least one $i \in \{1, \ldots, n\}$. Since $\alpha[u/0] \equiv \bigvee_{i=1}^{n} (x_i \wedge x'_i)$, we therefore have $\Phi \models \alpha[u/0]$. Obviously also $\Phi \models \alpha[u/1]$, thus we have $\Phi \models \alpha$.

It remains to prove that Φ is minimal. Since for each $k \in \{1, \ldots, m\}$ $\psi_k \equiv \psi'_k \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i)$ is satisfiable, no $\psi'_k \equiv \Phi_k$ entails $\alpha[u/0] \equiv \bigvee_{i=1}^{n} (x_i \wedge x'_i)$. *A fortiori* no Φ_k entails α.

Conversely suppose that $(\Phi, \alpha) \in \text{ARG} - \text{CHECK}$. Then, in particular, Φ entails $\alpha[u/0]$. Thus, we have $\psi' \models \bigvee_{i=1}^{n} (x_i \wedge x'_i)$, which implies that ψ is unsatisfiable. By the minimality of Φ we know that no Φ_k entails α. Since $\Phi_k \models \alpha[u/1]$, we conclude that $\Phi_k \not\models \alpha[u/0]$, which implies that $\psi'_k \wedge \bigwedge_{i=1}^{n} (\neg x_i \vee \neg x'_i)$ is satisfiable. As ψ'_k is monotonic, we also obtain that $\psi'_k \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i)$ and hence ψ_k itself is satisfiable.

We finally transform (Φ, α) into a B-instance for all B such that $\mathsf{S}_{00} \subseteq [B]$ by replacing every connective by its B-representation. This transformation works in logarithmic space for Φ since it consists in clauses of size 3. It also does for α if we first represent it as an $\vee$-tree of depth logarithmic in n. ■

PROPOSITION 3.4 *Let* B *be a finite set of Boolean functions such that* $\mathsf{D}_2 \subseteq [B]$. *Then* ARG-CHECK($B$) *is* DP-*complete.*

Proof We give a reduction from CRITICAL − SAT similar to Proposition 3.3. For $k \in \mathbb{N}$, we define g_k as the $(k+1)$-ary function verifying

(a) $g_k(z_1, \ldots, z_k, 0) \equiv \bigwedge_{i=1}^{k} z_i$ and
(b) $g_k(z_1, \ldots, z_k, 1) \equiv \bigvee_{i=1}^{k} z_i$.

Note that for every $k \in \mathbb{N}$, g_k is monotonic and self-dual, and thus contained in D_2. By abuse of notation, given a clause $C = (l^1 \vee l^2 \vee l^3)$ and a variable x, $g_3(C, x)$ stands for $g_3(l^1, l^2, l^3, x)$. Let $\psi = \bigwedge_{j=1}^{m} C_j$ be an instance of CRITICAL − SAT with $C_j = (l_j^1 \vee l_j^2 \vee l_j^3)$ and $\mathrm{Vars}(\psi) = \{x_1, \ldots, x_n\}$. Let further $u, v, x_1', \ldots, x_n'$ be fresh, pairwise distinct variables and $C_j' := C_j[\neg x_i / x_i' \mid 1 \leq i \leq n]$ for $1 \leq j \leq m$.

We map ψ to (Φ, α), where we define

$$\Phi = \{g_3(C_j', u) \mid 1 \leq j \leq m\}, \quad \text{and}$$

$$\alpha = g_n((g_2(x_i, x_i', v))_{1 \leq i \leq n}, u).$$

Obviously α and the formulæ in Φ are D_2-formulæ and thus satisfiable. Let $\psi' := \bigwedge_{j=1}^{m} C_j'$ and for $1 \leq k \leq m$, let Φ_k, ψ_k, ψ_k' denote the respective set of formulæ (clauses) where we deleted the kth formula (clause).

Note that if $\psi \in$ CRITICAL − SAT then each variable of ψ appears both as positive and as negative literal. Without loss of generality we may further assume that each literal has at least two occurrences in two different clauses. These two properties will assure that we have for all $k \in \{1, \ldots, m\}$

$$\mathrm{Vars}(\psi') = \mathrm{Vars}(\psi_k') = \{x_i, x_i' | 1 \leq i \leq n\}. \tag{1}$$

$$\mathrm{Vars}(\Phi) = \mathrm{Vars}(\Phi_k) = \{x_i, x_i' | 1 \leq i \leq n\} \cup \{u\}. \tag{2}$$

Suppose now that $\psi \in$ CRITICAL − SAT, i.e. ψ is unsatisfiable and ψ_k is satisfiable for all $k \in \{1, \ldots, m\}$. We show that Φ entails α for all possible values of u, v, where we consider in detail only the case where $v = 0$. The case $v = 1$ is analogous.

(i) $(u, v) = (1, 0)$: $\Phi[u/1] \equiv \psi'$ and $\alpha[u/1, v/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x_i')$. Since $\psi \equiv \psi' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ is unsatisfiable and ψ' is monotonic, all models of ψ' have to set both x_i and x_i' to 1 for at least one $i \in \{1, \ldots, n\}$. Therefore, $\Phi[u/1] \models \alpha[u/1, v/0]$.
(ii) $(u, v) = (0, 0)$: $\Phi[u/0] \equiv \bigwedge_{i=1}^{n}(x_i \wedge x_i')$ and $\alpha[u/0, v/0] \equiv \bigwedge_{i=1}^{n}(x_i \wedge x_i')$. Obviously $\Phi[u/0] \models \alpha[u/0, v/0]$.

It remains to prove that Φ is minimal. Since $\psi \in$ CRITICAL − SAT, $\psi_k \equiv \psi_k' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ is satisfiable and hence $\Phi_k[u/1] \equiv \psi_k'$ does not entail $\alpha[u/1, v/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x_i')$, i.e. $\Phi_k \not\models \alpha$.

Conversely suppose that $(\Phi, \alpha) \in$ ARG − CHECK. Then, in particular, $\Phi[u/1]$ entails $\alpha[u/1, v/0]$. Thus, we have $\psi' \models \bigvee_{i=1}^{n}(x_i \wedge x_i')$, which implies that ψ is unsatisfiable. By the minimality of Φ, we obtain that no Φ_k entails α. One easily verifies that in the cases $(u, v) \in \{(0, 0), (0, 1), (1, 1)\}$ Φ_k still entails α (use Equation (2) for $(u, v) = (0, 0)$). Thus, we have that $\Phi_k[u/1] \not\models \alpha[u/1, v/0]$, which implies that $\psi_k' \wedge \bigwedge_{i=1}^{n}(\neg x_i \vee \neg x_i')$ is satisfiable. As ψ_k' is monotonic, we obtain that $\psi_k' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ and hence ψ_k itself is satisfiable, too.

Finally, we transform (Φ, α) into a B-instance for all B such that $\mathsf{D}_2 \subseteq [B]$ in replacing all occurrences of g_k by its B-representation. This transformation works in logarithmic space, because we may assume the function g_n to be a g_2-tree of depth logarithmic in n. ■

The full picture for ARG-CHECK(B) is given in the forthcoming theorem, where we also provide the results for the 'easier' clones.

THEOREM 3.5 *Let B be a finite set of Boolean functions. Then the argument validity problem for propositional B-formulæ,* ARG − CHECK(B), *is*

(1) DP-*complete if* $\mathsf{S}_{00} \subseteq [B]$ *or* $\mathsf{S}_{10} \subseteq [B]$ *or* $\mathsf{D}_2 \subseteq [B]$,
(2) *in* P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$,
(3) *in* LOGSPACE *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$.

Proof For DP-completeness, according to Propositions 3.3 and 3.4, it remains only to deal with the case $\mathsf{S}_{10} \subseteq [B]$. Since $\mathsf{D}_2 \subseteq \mathsf{M}_1 = [\mathsf{S}_{10} \cup \{1\}] \subseteq [B \cup \{1\}]$, we obtain that ARG − CHECK($B \cup \{1\}$) is DP-hard by Proposition 3.4. As $\wedge \in [B]$, we may apply Lemma 3.1 and obtain the DP-hardness of ARG-CHECK(B).

In the case $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$, the sets Φ, $\Phi \cup \{\neg\alpha\}$, and $(\Phi \setminus \{\varphi\}) \cup \{\neg\alpha\}$ for all $\varphi \in \Phi$ can be easily transformed into systems of linear equations. Thus, checking the three conditions as mentioned in Section 2.3 comes down to solving a polynomial number of systems of linear equations. This can be done in polynomial time, using Gaussian elimination. For $[B] \subseteq \mathsf{V}$, for $[B] \subseteq \mathsf{E}$, and for $[B] \subseteq \mathsf{N}$ this check can be done in logarithmic space, as in this case the satisfiability of sets of B-formulæ can be determined in logarithmic space (Schnoor 2005). ■

3.3. *The complexity of existence and dispensability*

We next consider the problems ARG(B) and ARG-DISP(B). For the membership part of the forthcoming results, we can make use of the results from the previous subsection.

THEOREM 3.6 *Let B be a finite set of Boolean functions. Then the argument existence problem for propositional B-formulæ,* ARG(B), *is*

(1) Σ_2^{p}-*complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,
(2) coNP-*complete if* $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ *with* $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ *and* $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$,
(3) *in* NP *if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$,
(4) *in* P *if* $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$, *and*
(5) *in* LOGSPACE *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$.

The same classification holds for ARG-DISP(B).

Proof The general argumentation problem has been shown to be Σ_2^{p}-complete by Parsons et al. (2003) via a reduction from $\mathrm{Qsat}_{2,\exists}$. An instance of this problem is a quantified formula $\exists X \forall Y \beta$, and one may assume that the formula β is in 3DNF, i.e. $\beta = \bigvee_{j=1}^{p} t_j$ with exactly three literals by term. The authors map such an instance $\exists X \forall Y \beta$ to (Δ, α), where $\Delta := \{x \leftrightarrow 0, x \leftrightarrow 1 | x \in X\}$, and $\alpha := \beta$. We use this reduction to obtain Σ_2^{p}-completeness for ARG(B) if $[B] = \mathsf{BF}$. We insert parentheses in β and obtain a formula of logarithmic parentheses-depth only. We can now substitute all occurring connectives with their B-representations and obtain this way in logarithmic space an instance of ARG(B) which is equivalent to the original (Δ, α).

As $\wedge \in \mathsf{S}_1$ and $[\mathsf{S}_1 \cup \{1\}] = \mathsf{BF}$, we obtain Σ_2^{p}-completeness for the case $\mathsf{S}_1 \subseteq [B]$ according to Lemma 3.1. For the case $\mathsf{D} \subseteq [B]$, we obtain Σ_2^{p}-completeness by Lemma 3.2, since $\mathsf{D}_2 \subseteq \mathsf{D}$ and $[\mathsf{D} \cup \{1\}] = \mathsf{BF}$.

For $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ with $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ and $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$, membership in NP follows from the facts that satisfiability is in LOGSPACE (Lewis 1979), while entailment is in coNP (Beyersdorff et al. 2009a). To prove the coNP-hardness of ARG(B), we give a reduction from the implication problem for B-formulæ, which is coNP-hard if $[B]$ contains one of the clones S_{00}, S_{10}, D_2. Let (ψ, α) be a pair of B-formulæ. We map this instance to $(\{\psi\}, \alpha)$ if ψ is satisfiable (which is easy to decide) and to a trivial positive instance otherwise.

For $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$, membership in NP follows from the fact that in this case ARG-CHECK is in P. Owing to the trivial satisfiability of B-formulæ for $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$, we can improve the upper bound for ARG(B) with $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$ to membership in P.

In all other cases, LOGSPACE-membership follows from the fact that both problems, satisfiability and entailment, are contained in LOGSPACE (see Beyersdorff et al. 2009a) for B-formulæ.

Finally, observe that we have ARG-DISP(B) $\equiv_{\mathrm{m}}^{\log}$ ARG(B). To prove that ARG(B) $\leq_{\mathrm{m}}^{\log}$ ARG-DISP(B), map $\mathcal{A} = (\Delta, \alpha)$ to $\mathcal{D} := (\Delta \cup \{t\}, \alpha, t)$. For the converse direction, map $\mathcal{D} = (\Delta, \alpha, \varphi)$ to $\mathcal{A} := (\Delta \setminus \{\varphi\}, \alpha)$. ■

3.4. *The complexity of relevance*

The remaining decision problem to analyse is ARG-REL(B) which turns out to be the most difficult in terms of complexity.

PROPOSITION 3.7 *Let B be a finite set of Boolean functions such that $\mathsf{S}_{00} \subseteq [B]$. Then* ARG-REL($B$) *is Σ_2^{p}-complete.*

Proof To see that ARG-REL(B) is contained in Σ_2^{p}, observe that, given an instance $(\Delta, \alpha, \varphi)$, we can guess a set $\Phi \subseteq \Delta$ such that $\varphi \in \Phi$ and verify conditions (C1)–(C3) in polynomial time using an NP-oracle.

To prove Σ_2^{p}-hardness, we provide a reduction from the problem $\mathrm{Qsat}_{2,\exists}$. An instance of this problem is a quantified formula $\exists X \forall Y \beta$, where $\beta = \bigvee_{j=1}^{p} t_j$ with exactly three literals by term. Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. Let u, v, $x'_1, \ldots, x'_n, y'_1, \ldots, y'_m$ be fresh variables. We transform $\exists X \forall Y \beta$ to $(\Delta, \alpha, \varphi)$, where

$$\Delta := \{x_i, x'_i \mid 1 \leq i \leq n\} \cup \left\{ v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i) \right\} \cup \{u\},$$

$$\alpha := \beta' \wedge v \wedge \left(\bigvee_{i=1}^{n} (x_i \wedge x'_i) \vee u \right),$$

$$\varphi := u,$$

and where $\beta' = \bigvee_{j=1}^{p} t'_j$ and $t'_j := t_j[\neg x_1/x'_1, \ldots, \neg x_n/x'_n, \neg y_1/y'_1, \ldots, \neg y_m/y'_m]$ for all $1 \leq j \leq p$.

We show that $\exists X \forall Y \beta$ is valid if and only if $(\Delta, \alpha, \varphi) \in$ ARG-REL($\{\wedge, \vee\}$). If $\exists X \forall Y \beta$ is valid, then there exists an assignment $\sigma : X \to \{0, 1\}$ such that $\sigma \models \beta$. Consequently, for $\Phi := \{x_i \mid \sigma(x_i) = 1\} \cup \{x'_i \mid \sigma(x_i) = 0\} \cup \{u, v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)\}$, we obtain $\Phi \models \alpha$. As Φ is consistent, it thus remains to show that u is relevant, i.e. that $\Phi \setminus \{u\} \not\models \alpha$. This follows from the fact that $\Phi \setminus \{u\}$ is satisfied by any assignment σ' extending σ with $\sigma'(u) := 0$ and $\sigma'(x'_i) := 1 - \sigma(x_i)$, while such a σ' does not entail $\bigvee_{i=1}^{n} (x_i \wedge x'_i) \vee u$ and hence $\sigma' \not\models \alpha$.

For the converse direction, let Φ be a support for α such that $u \in \Phi$. Since $\Phi \models \alpha$ we conclude that $v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i) \in \Phi$ and hence $\Phi = \mathcal{X} \cup \{v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)\} \cup \{u\}$, for some $\mathcal{X} \subseteq \{x_i, x'_i \mid 1 \leq i \leq n\}$. From $\Phi \models \alpha$ also follows that $\Phi \models \beta'$. From the minimality of Φ, we conclude that in particular $\Phi \setminus \{u\} \not\models \alpha$. And therefore $\Phi \not\models \bigvee_{i=1}^{n} (x_i \wedge x'_i)$. That is $\Phi \wedge \bigwedge_{i=1}^{n} (\neg x_i \vee \neg x'_i)$ is satisfiable and since Φ is monotonic, consequently also $\Phi \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i)$ is satisfiable. Summed up, we know that $\gamma := \mathcal{X} \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i) \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)$ is satisfiable and $\gamma \models \beta'$. Hence, *a fortiori*, $\gamma' := \mathcal{X} \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i) \wedge \bigwedge_{i=1}^{m} (y_i \oplus y'_i)$ is satisfiable and $\gamma' \models \beta'$. Define now $\sigma_X(x_i) = 1$ if $x_i \in \mathcal{X}$, $\sigma_X(x_i) = 0$ otherwise. Obviously any extension of σ_X to Y satisfies β and therefore $\exists X \forall Y \beta$ is valid.

It remains to transform $(\Delta, \alpha, \varphi)$ into an ARG-REL(B)-instance for all B such that $\mathsf{S}_{00} \subseteq [B]$. As both $\wedge$ and $\vee$ are associative, we can insert parentheses into $(\Delta, \alpha, \varphi)$, such that we can represent each formula as binary $\{\wedge, \vee\}$-tree of logarithmic depth. Let f be a fresh variable and let h be the Boolean function in S_{00} defined by $h(f, x, y) \equiv f \vee (x \wedge y)$. We further transform our instance into $(\Delta', \alpha' \vee f, \varphi')$, where $\Delta', \alpha', \varphi'$ are obtained by replacing each occurrence of $x \wedge y$ by $h(f, x, y)$. One easily verifies that $(\Delta', \alpha' \vee f, \varphi')$ is in ARG-REL($\{\vee, h\}$) if and only if $(\Delta, \alpha, \varphi) \in$ ARG-REL($\{\wedge, \vee\}$). We finally replace $\vee$ and h by their B-representation. ■

PROPOSITION 3.8 *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathsf{V}$ or $[B] \subseteq \mathsf{E}$ or $[B] \subseteq \mathsf{N}$. Then* ARG-REL(B) *is in* LOGSPACE.

Proof We assume the representation of V-, E-, or N-formulæ as respectively positive clauses, positive terms, or literals. Let us first consider ARG-REL(B) for $[B] \subseteq \mathsf{E}$. It is easy to observe that a set of positive terms Δ entails a positive term α if and only if $\mathrm{Vars}(\alpha) \subseteq \mathrm{Vars}(\Delta)$. We claim that Algorithm 1 decides ARG-REL(B).

Algorithm 1 Algorithm for ARG-REL(B) with $[B] \subseteq \mathsf{E}$.

Require: a set Δ of positive terms and positive terms α, φ with $\varphi \in \Delta$.
1: **for all** $x \in \mathrm{Vars}(\varphi)$ **do**
2: $\quad\Delta_x := \{\varphi\} \cup \{\tau \in \Delta \mid x \notin \mathrm{Vars}(\tau)\}$
3: $\quad$**if** $\Delta_x \models \alpha$ **then**
4: $\quad\quad$accept
5: $\quad$**end if**
6: **end for**
7: reject

Algorithm 1 can be implemented using only a logarithmic amount of space if we do not construct Δ_x entirely, but rather check the condition in line 3 directly: $\Delta_x \models \alpha$ holds if and only if $\mathrm{Vars}(\alpha) \subseteq \mathrm{Vars}(\varphi) \cup \mathrm{Vars}(\{\tau \in \Delta | x \notin \mathrm{Vars}(\tau)\})$.

To prove correctness, notice that Algorithm 1 accepts only if there exists a $\Delta_x \subseteq \Delta$ such that $\Delta_x \models \alpha$ and $\Delta_x \setminus \{\varphi\} \not\models \alpha$. Thus Δ_x contains a support Φ such that $\varphi \in \Phi$. Conversely, let Φ be a support such that $\varphi \in \Phi$. Since $\Phi \models \alpha$ and $\Phi \setminus \{\varphi\} \not\models \alpha$, there is at least one $x_i \in (\mathrm{Vars}(\varphi) \cap \mathrm{Vars}(\alpha)) \setminus \mathrm{Vars}(\Phi)$. For this x_i, the algorithm constructs $\Delta_{x_i} := \{\varphi\} \cup \{\tau \in \Delta \mid x_i \notin \mathrm{Vars}(\tau)\}$. Obviously $\Phi \subseteq \Delta_{x_i}$ and therefore $\Delta_{x_i} \models \alpha$ which causes the algorithm to accept.

Next, consider ARG-REL(B) for $[B] \subseteq \mathsf{V}$. Observe that a set of positive clauses C entails a positive clause α if and only if there is a clause $c \in C$ such that $\mathrm{Vars}(c) \subseteq \mathrm{Vars}(\alpha)$. Thus if there is a support Φ with $\varphi \in \Phi$ then it is the singleton $\{\varphi\}$. Given $(\Delta, \alpha, \varphi)$ as an instance of ARG-REL(V), it hence suffices to check whether $\mathrm{Vars}(\varphi) \subseteq \mathrm{Vars}(\alpha)$, which can be done in LOGSPACE.

Finally ARG-REL(B) for $[B] \subseteq \mathsf{N}$ is in LOGSPACE, since each B-formula can be transformed into a single literal. ■

We obtain the following complexity classification for ARG-REL.

THEOREM 3.9 *Let B be a finite set of Boolean functions. Then the argument relevance problem for propositional B-formulæ,* ARG-REL(B), *is*

(1) Σ_2^{p}*-complete if* $\mathsf{S}_{00} \subseteq [B]$ *or* $\mathsf{D}_2 \subseteq [B]$ *or* $\mathsf{S}_{10} \subseteq [B]$,

(2) *in* NP *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$,
(3) *in* LOGSPACE *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$.

Proof Applying Lemma 3.1 and Lemma 3.2 as shown in Section 3.1, for (1) it suffices to show hardness for B such that $\mathsf{S}_{00} \subseteq [B]$. This has been done in Proposition 3.7. Item (2) follows from the fact that for $[B] \subseteq \mathsf{L}$, ARG-CHECK(B) is in P (Proposition 3.5) and (3) has been shown in Proposition 3.8. ■

4. Enumeration and counting problems

4.1. *Enumeration*

A general procedure to enumerate all supports in Δ for a given claim α is given in Algorithm 2. The initial call has to be done by ENUM($\Delta, \alpha, \emptyset$). The idea is to build supports recursively, picking up at each step a formula φ from Δ and deciding whether φ will be present in the currently constructed support or not. The decision procedures for ARG-REL and ARG-DISP are used in order to avoid backtracking.

Algorithm 2 General enumeration algorithm for ARG

ENUM(Δ, α, Φ)
if ARG − CHECK(Φ, α) **then**
  output Φ
  return
end if
choose a $\varphi \in \Delta$
if ARG-REL(Δ, α, φ) **then**
  ENUM($\Delta \backslash \{\varphi\}, \varphi \rightarrow \alpha, \Phi \cup \{\varphi\}$)
end if
if ARG-DISP(Δ, α, φ) **then**
  ENUM($\Delta \backslash \{\varphi\}, \alpha, \Phi$)
end if
return

The correctness of the procedure is based on the deduction theorem, which says that $\{\varphi\} \cup \Phi \models \alpha$ if and only if $\Phi \models (\varphi \rightarrow \alpha)$. Observe that, in general, when initially called on B-formulæ, this procedure deals in its recursive calls with some formulæ which are not B-formulæ anymore (because of the formula $\varphi \rightarrow \alpha$). There is an exception to the sets B such that $\mathsf{S}_0 \subseteq [B]$, since in this case $\rightarrow \in [B]$ and thus $\varphi \rightarrow \alpha$ can be represented as a B-formula.

Note that the only cases in which we can hope to obtain polynomial delay enumeration algorithms are those for which the decision problem ARG(B) is tractable, i.e. the cases $[B] \subseteq \mathsf{V}$ or $[B] \subseteq \mathsf{N}$ or $[B] \subseteq \mathsf{E}$ or $[B] \subseteq \mathsf{L}_1$. Indeed the presented procedure can be slightly modified in these cases, except for the case $[B] \subseteq \mathsf{L}_1$, in order to either avoid recursive calls or manage recursive calls on B-formulæ only. As a consequence and since in these cases the three decision problems ARG-CHECK(B), ARG-REL(B) and ARG-DISP(B) are in P, the procedure will generate efficiently all possible arguments. This is made precise in the following proposition.

PROPOSITION 4.1 *Let B be a finite set of Boolean functions. If* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{N}$ *then* ENUM-ARG(B) *is in* FP. *If* $[B] \subseteq \mathsf{E}$ *then* ENUM-ARG(B) *is in* DelayP.

Proof If $[B] \subseteq \mathsf{V}$ or $[B] \subseteq \mathsf{N}$ then we have seen (in the proof of Proposition 3.8) that every support for α consists of a single formula. For this reason, no recursive call in the procedure is necessary. It is enough to check whether every single formula from Δ is a support for α. Since ARG-CHECK(B) is in P in this case (see Theorem 3.5), we are done.

If $[B] \subseteq \mathsf{E}$ then every B-formula is a positive term. Recall that if Φ is a set of positive terms, and φ is a positive term, then we have $\Phi \models \alpha$ if and only if $\mathrm{Vars}(\alpha) \subseteq \mathrm{Vars}(\Phi)$. As a consequence, in the case of positive terms $\{\varphi\} \cup \Phi \models \alpha$ if and only if $\Phi \models \alpha \setminus \varphi$, where $\alpha \setminus \varphi$ denotes the term obtained from α by removing those variables that occur in φ. Therefore, in the general enumeration procedure described above, we can make the recursive call on $\alpha \setminus \varphi$ instead of $\alpha \to \varphi$. In this way, starting from B-formulæ, all recursive calls will be made on B-formulæ as well. Since ARG-CHECK(B), ARG-REL(B) and ARG-DISP(B) are all in P for any B such that $[B] \subseteq \mathsf{E}$ (see Theorems 3.9 and 3.6), it is then clear that the procedure runs with polynomial delay and uses only polynomial space. ■

4.2. *Counting*

The argument counting problem #ARG(B) belongs to #·NP. This follows from the facts that checking an argument is in $\mathrm{DP} \subseteq \mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{p}}$ (see Section 3.2) and from the equality $\#\cdot\Delta_2^{\mathrm{p}} = \#\cdot\mathrm{NP}$ (see Hemaspaandra and Vollmer 1995).

PROPOSITION 4.2 *Let B be a finite set of Boolean functions such that either $\mathsf{D} \subseteq [B]$ or $\mathsf{S}_1 \subseteq [B]$. Then* #ARG($B$) *is* #·NP*-complete.*

Proof We show #·NP-hardness by giving a parsimonious reduction from the following #·NP-hard problem: Count the number of satisfying assignments of $\psi(x_1, \ldots, x_n) = \forall y_1 \cdots \forall y_m \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$, where φ is a DNF-formula (Durand, Hermann, and Kolaitis 2005).

Let $t, r_1, \ldots r_n$ be fresh variables. We map ψ to (Δ, α) which we define as follows:

$$\Delta = \{x_i, \neg x_i, x_i \longrightarrow r_i, \neg x_i \longrightarrow r_i | 1 \leq i \leq n\} \cup \{\varphi \longrightarrow t\}$$

$$\alpha = t \wedge \bigwedge_{i=1}^{n} r_i.$$

By minimality, every support $\Phi \subseteq \Delta$ for α contains for every i either x_i or $\neg x_i$, but not both. One easily verifies that by $m \mapsto \{x_i | m(x_i) = 1\} \cup \{\neg x_i | m(x_i) = 0\}$ we define a bijection between the models of ψ and the supports for α in Δ.

Since $\mathsf{D}_2 \subseteq \mathsf{D}$ and $\mathsf{E}_2 \subseteq \mathsf{S}_1$, we can use Lemmas 3.1 and 3.2. Observe that the reductions used in the proofs of these lemmas are parsimonious. Therefore we have $\#\mathrm{ARG}(B \cup \{1\}) \leq_! \#\mathrm{ARG}(B)$. Thus, it is sufficient to prove that hardness holds for $\#\mathrm{ARG}(B \cup \{1\})$. Since $\mathsf{BF} = [\mathsf{D} \cup \{1\}] \subseteq [B \cup \{1\}]$ and $\mathsf{BF} = [\mathsf{S}_1 \cup \{1\}] \subseteq [B \cup \{1\}]$, it suffices finally to prove that hardness of #ARG(B) holds for any B such that $\mathsf{BF} = [B]$. Suppose that $\varphi = \bigvee_{i \in I} t_i$ and let Γ' be the set of formulæ obtained from Γ by replacing $\varphi \to t \equiv \neg(\bigvee_{i \in I} t_i) \vee t \equiv (\bigwedge_{i \in I} \neg t_i) \vee t$ by the set of clauses $\{\neg t_i \vee t | i \in I\}$. Then Γ' is a set of disjunctions of literals, whose size is polynomially bounded by $|\Gamma|$. Hence, by the associativity of $\vee$, Γ' can be transformed in logarithmic space into an equivalent set of B-formulæ. This provides a parsimonious reduction from the above #·NP-complete problem to #ARG(B). ■

PROPOSITION 4.3 *Let B be a finite set of Boolean functions such that $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ with $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ and $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$. Then* #ARG($B$) *is in* #·NP *and* #P*-hard.*

Proof Since the reductions in Lemmas 3.1 and 3.2 are parsimonious, analogously to Theorem 3.9, it suffices to show hardness for the case $\mathsf{S}_{00} \subseteq [B]$.

We prove #P-hardness by giving a parsimonious reduction from the following #P-hard problem: count the number of models of a positive 2-CNF-formula. Let $\varphi = \bigwedge_{j=1}^{p} C_j$ with $\mathrm{Vars}(\varphi) = \{x_1, \ldots, x_n\}$ be such a formula. Let $x'_1, \ldots x'_n$ be fresh variables. We map φ to (Δ, α) which we define as follows:

$$\Delta = \{x_i, x'_i | 1 \leq i \leq n\}$$
$$\alpha = \varphi \wedge \bigwedge_{i=1}^{n} (x_i \vee x'_i)$$

By minimality every support $\Phi \subseteq \Delta$ for α contains for every i either x_i or x'_i, but not both. One easily verifies that by $m \mapsto \{x_i | m(x_i) = 1\} \cup \{x'_i | m(x_i) = 0\}$, we define a bijection between the models of φ and the supports for α in Δ.

It remains to show that α can be transformed in polynomial time into a B-formula for all B such that $\mathsf{S}_{00} \subseteq [B]$. As $\wedge$ and $\vee$ are monotonic, we can insert parentheses such that α is represented as a binary $(\wedge, \vee)$-tree of logarithmic depth. Let f be a fresh variable and let h be the Boolean functions in S_{00} defined by $h(f, x, y) \equiv f \vee (x \wedge y)$. We further transform α into α' by replacing every occurrence of $x \wedge y$ by $h(f, x, y)$. Since f does not occur in Δ it is clear that the arguments for α in Δ are the ones for α'. Since $\vee, h \in \mathsf{S}_{00} \subseteq [B]$, we finally replace $\vee$ and h by their B-representation, thus concluding the proof. ■

PROPOSITION 4.4 *Let B be a finite set of Boolean functions such that $\mathsf{E}_2 \subseteq [B] \subseteq \mathsf{E}$. Then* #ARG($B$) *is* #P*-complete.*

Proof Membership in #P follows from the fact that $\mathrm{ARG}(B) \in \mathrm{P}$ for every B such that $[B] \subseteq \mathsf{E}$.

To prove #P-hardness, we establish a parsimonious reduction from the same #P-hard problem as in the previous proposition. So let $\varphi = \bigwedge_{j=1}^{p} C_j$ with $\mathrm{Vars}(\varphi) = \{x_1, \ldots, x_n\}$ be a positive 2-CNF-formula. We introduce p new variables $\{c_1, \ldots, c_p\}$ that will represent the clauses. For every $1 \leq i \leq n$ let $\mathrm{occ}(x_i) = \{j | 1 \leq j \leq p, x_i \text{ occurs } C_j\}$. We map φ to (Δ, α) which we define as follows:

$$\Delta = \left\{ x_i, x_i \wedge \bigwedge_{j \in \mathrm{occ}(x_i)} c_j | 1 \leq i \leq n \right\}$$
$$\alpha = \bigwedge_{j=1}^{p} c_j \wedge \bigwedge_{i=1}^{n} x_i$$

By minimality every support $\Phi \subseteq \Delta$ for α contains for every i either x_i or $x_i \wedge \bigwedge_{j \in \mathrm{occ}(x_i)} c_j$, but not both. One easily verifies that by $m \mapsto \{x_i \wedge \bigwedge_{j \in \mathrm{occ}(x_i)} c_j | m(x_i) = 1\} \cup \{x_i | m(x_i) = 0\}$, we define a bijection between the models of φ and the supports for α in Δ.

It remains to transform α and the formulæ in Δ into B-formulæ. We can insert parentheses in each term in order to represent it as a binary $\wedge$-tree of logarithmic depth. Then it suffices to replace the $\wedge$-connective by its B-representation which exists since $\wedge \in \mathsf{E}_2 \subseteq [B]$. ■

THEOREM 4.5 *Let B be a finite set of Boolean functions. Then the argument counting problem for propositional B-formulæ,* #ARG(B)*, is*

(1) #·NP*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,

(2) *in* #·NP *and* #P*-hard if* $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ *with* $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ *and* $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$,
(3) *in* #P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$,
(4) #P*-complete if* $\mathsf{E}_2 \subseteq [B] \subseteq \mathsf{E}$, *and*
(5) *in* FP *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{N}$.

Proof Items (1), (2) and (4) follow directly from the above three propositions. Item (3) follows from the fact that $\textsc{Arg}(B) \in$ NP for B such that $[B] \subseteq \mathsf{L}$ (see Theorem 3.6). In the last case, all supports are singletons (see e.g. Theorem 3.8), thus there is at most a polynomial number of supports that can hence be parsed in polynomial time in order to determine the exact number of solutions. ■

Let us conclude this section by pointing out that for the case where $\mathsf{E}_2 \subseteq [B] \subseteq \mathsf{E}$, i.e. when formulæ may depend on more than one variable and are representable as positive terms, the counting problem is intractable (#P-hard, Proposition 4.4), whereas enumeration is tractable ($\in$ DelayP, Proposition 4.1).

5. Discussion of results

Complexity classifications along the lines of Boolean clones have already been carried out for AI formalisms as circumscription (Thomas 2009) and abduction (Creignou et al. 2010). In particular, the latter work is closely related to the contents of this paper. To make this more precise, let us consider the positive abduction problem P-Abd(B). It takes as an instance a triple (Γ, H, m), where $\Gamma \subseteq \mathcal{L}(B)$, $m \in \mathcal{L}(B)$, H is a set of variables, and asks whether there exists an explanation $E \subseteq H$ such that $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \models m$. Hence, the main differences to argumentation are as follows: (a) the knowledge base Γ is always entirely used (together with a selected subset of hypotheses) in the tests for consistency and entailment, while for argumentation these tests are performed on a chosen subset of the knowledge base Δ; (b) the parameterisation via B-formula affects only 'fixed' parts Γ and m in abduction (the hypotheses are by definition restricted); while in argumentation, the parameterisation is applied to the knowledge base Δ from which one has to select a subset. Nonetheless, the following relations between these two formalisms hold:

(1) if $\wedge \in [B]$, i.e. if $\mathsf{E}_2 \subseteq [B]$, then P-Abd($B$) $\leq_{\mathrm{m}}^{\log}$ $\textsc{Arg}(B)$
(2) if $\rightarrow \in [B]$, i.e. if $\mathsf{S}_0 \subseteq [B]$, then $\textsc{Arg}(B) \leq_{\mathrm{m}}^{\log}$ P-Abd(B)

For (1), one can give the reduction $(\Gamma, A, \psi) \mapsto (\Delta, \alpha)$, where $\Delta := \{\bigwedge_{\varphi \in \Gamma} \varphi\} \cup A$ and $\alpha := \psi \wedge \bigwedge_{\varphi \in \Gamma} \varphi$; likewise the mapping $(\Delta, \alpha) \mapsto (\Gamma, A, \alpha)$, where $A := \{x_\varphi \mid \varphi \in \Delta\}$ and $\Gamma := \{x_\varphi \leftrightarrow \varphi; |\varphi \in \Delta\}$ underlies (2).

It turns out that Arg and Arg-Disp have the same complexity classification as positive abduction. This is due to the fact that minimality of the argument plays no role in Arg and Arg-Disp. However, for Arg-Rel the situation is different but we expect similarly harder complexity for the relevance problem in abduction with respect to subset-minimal explanations (see Eiter and Gottlob 1995 for the definitions) which has not been analysed by Creignou et al. (2010). In other words, the results provided in the present paper can be used to obtain novel results for certain variants of abduction, which have not been classified yet.

Table 2 gives an overview of the results for the studied argumentation problems. The small numbers on the right side in the table cells refer to the corresponding theorem/proposition/remark.

Our results show, for instance, that when the knowledge base's formulæ are restricted to be represented as positive clauses or single literals (column $\mathsf{V}_*, \mathsf{N}_*$) then all considered decision problems as well as counting and enumeration are very easy. When allowing for positive terms

Table 2. The complexity of the argumentation problems.

	$\mathsf{N}_*, \mathsf{V}_*$	E_*	$\mathsf{L}_1, \mathsf{L}_2$	$\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3$	$\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}, \mathsf{R}_1$	$\mathsf{D}, \mathsf{S}_1 \subseteq [B] \subseteq \mathsf{BF}$
ARG-CHECK	$\in$ L, 3.5	$\in$ L, 3.5	$\in$ P, 3.5 $\in$ P, 3.5	DP-c, 3.3, 3.4	DP-c, 3.3, 3.4	
ARG, ARG-DISP	$\in$ L, 3.6	$\in$ L, 3.6	$\in$ P, 3.6	$\in$ NP, 3.6	coNP-c, 3.6	Σ_2^{p}-c, 3.6
ARG-REL	$\in$ L, 3.8	$\in$ L, 3.8	$\in$ NP, 3.9	$\in$ NP, 3.9	Σ_2^{p}-c, 3.7	Σ_2^{p}-c, 3.7
ENUM-ARG	$\in$ FP, 4.1	$\in$ DelayP, 4.1	Unknown	Unknown	coNP-h, 3.6	Σ_2^{p}-h, 3.6
#ARG	$\in$ FP, 4.5	#P-c, 4.4	$\in$ #P, 4.5	$\in$ #P, 4.5	$\in$ #·NP, #P-h, 4.3	#·NP-c, 4.2

Note: The $*$-subscripts on clones denote all valid completions, L abbreviates LOGSPACE, and '-c' and '-h' indicate, respectively, completeness and hardness.

(column E_*), the decision problems remain very easy, the enumeration problem becomes less trivial but still tractable (DelayP), whereas the counting problem becomes even intractable, namely #P-complete.

Notably are the sets B of Boolean connectives where $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ with $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ and $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$. This case typically applies to monotonic formulæ in which no negation is involved. Such sets of B give coNP-completeness for ARG(B), while ARG-REL(B) remains complete for Σ_2^{p}. It may come as a surprise that in these cases verifying an argument is potentially harder than deciding the existence of an argument (ARG-CHECK is DP-complete, ARG is only coNP-complete). This is due to the fact that when verifying an argument, the minimality condition of a support has to be checked, whereas this condition is of no importance to the argument existence problem.

The results obtained for the L-cases are partial. This corresponds to the case where individual formulæ are linear equations over the two-element field. The fact that ARG(B) with $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}_1$ is in P relies on Gaussian elimination, knowing that in this case we only have to check whether $\Phi \models \alpha$, that is whether the linear system $\Phi \wedge \neg\alpha$ is satisfiable. For the corresponding problems ARG-REL(B), in which minimality plays a role, we only have an NP upper-bound, so far. In fact, the exact classification of the problems into tractable and intractable cases remains open for affine sets of Boolean connectives in the following cases: ARG(B) with $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$ and ARG-REL(B) with $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$.[1]

6. Conclusion and future work

To summarise, we took in this paper first steps to understanding the complexity of logic-based argumentation. We provided a classification of the complexity of four important decision tasks (Figure 1), as well as a classification of the complexity of enumeration and counting, for all possible restrictions on the set of allowed connectives.

The interpretation of our results is twofold: first, we have shown that the high complexity (i.e. Σ_2^{p}) only shows up for small parts of the lattice. In other words, only a few Boolean clones provide the inherent full complexity, while others allow for alternative, less involved, algorithms. Nonetheless, for the important problem of ARG-REL, Σ_2^{p} covers the most interesting areas of the lattice. The bad news is that all tractable fragments are of little practical relevance. Recall that these fragments were literals, positive terms, positive clauses, and with each of the latter two restrictions we cannot even construct an inconsistent Δ. This also hints that for future work, it might be important to restrict the connectives in Δ, and respectively, α independently.

The complexity of the problems studied in this paper is also a computational core for evaluating more complex argumentation problems, for instance, the warranted formula problem (WFP) on argument trees, which has recently been shown to be PSPACE-complete by Hirsch and Gorogiannis (2011). We expect that fragments studied here also lower the complexity of WFP, but leave details

for future work. Another problem which is amenable to the kind of complexity analysis we did in this paper is the identification of conflicts between two arguments (different notions for conflicts between arguments based on classical logic exist, see e.g. Gorogiannis and Hunter 2011) which is an intractable problem itself; however, since most conflicts are identified via the implication problem, we expect results similar to the one derived by Beyersdorff et al. (2009a). A complexity analysis in that direction has already been undertaken by Wooldridge, Dunne, and Parsons (2006). In that paper, the authors studied problems as, for instance, equivalence of arguments. Further future work also concerns studying the complexity of all the problems investigated here in the popular Schaefer's framework in which formulas are in generalised conjunctive normal form (see Creignou and Zanuttini (2006), Nordh and Zanuttini (2008) for complexity classifications of abduction in this framework, and Creignou and Vollmer (2008) for a survey of results obtained for various non-monotonic logics).

Acknowledgements

Supported by ANR *ENUM* 07-BLAN-0327-04, WWTF grant ICT 08-028, FWF grant P20704-N18, ÖAD grant Amadée 17/2011/EGIDE PHC Amadeus project 24908NM, and DFG grant VO 630/6-2.

Notes

1. We note that the complexity of the corresponding fragments remained unclassified also for circumscription and positive abduction.

References

Amgoud, L., and Besnard, P. (2010), 'A Formal Analysis of Logic-based Argumentation Systems', in *Proceedings of the 4th International Conference on Scalable Uncertainty Management (SUM'10, Toulouse, France)*, eds. A. Deshpande and A. Hunter, Vol. 6379 of Lecture Notes in Computer Science, Springer Verlag, pp. 42–55.

Amgoud, L., and Cayrol, C. (2002), 'A Reasoning Model Based on the Production of Acceptable Arguments', *Annals of Mathematics and Artificial Intelligence*, 34, 197–215.

Baroni, P., Dunne, P., and Giacomin, M. (2010), 'On Extension Counting Problems in Argumentation Frameworks', in *Proceedings of the 3rd International Conference on Computational Models of Argument* (*COMMA 2010, Desenzano del Garda, Italy*), eds. P. Baroni, F. Cerutti, M. Giacomin, and G. Simari, Vol. 216 of Frontiers in Artificial Intelligence and Applications, IOS Press, pp. 63–74.

Bauland, M., Hemaspaandra, E., Schnoor, H., and Schnoor, I. (2006), 'Generalized Modal Satisfiability', in *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science* (*STACS 2006, Marseille, France*), eds. B. Durand and W. Thomas, Vol. 3884 of Lecture Notes in Computer Science, Springer Verlag, pp. 500–511.

Bauland, M., Schneider, T., Schnoor, H., Schnoor, I., and Vollmer, H. (2008), 'The Complexity of Generalized Satisfiability for Linear Temporal Logic', *Logical Methods in Computer Science*, 5.

Bench-Capon, T., and Dunne, P. (2007), 'Argumentation in Artificial Intelligence', *Artificial Intelligence*, 171, 619–641.

Besnard, P., and Hunter, A. (2001), 'A Logic-based Theory of Deductive Arguments', *Artificial Intelligence*, 128, 203–235.

Besnard, P., and Hunter, A. (2008), *Elements of Argumentation*, MIT Press.

Beyersdorff, O., Meier, A., Thomas, M., and Vollmer, H. (2009a), 'The Complexity of Propositional Implication', *Information Processing Letters*, 109, 1071–1077.

Beyersdorff, O., Meier, A., Thomas, M., and Vollmer, H. (2009b), 'The Complexity of Reasoning for Fragments of Default Logic', in *Proceedings of the 12th International Conference on Theory and Applications*

of Satisfiability Testing (SAT 2009, Swansea, UK), ed. O. Kullmann, Vol. 5584 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 51–64.

Caminada, M., and Amgoud, L. (2007), 'On the Evaluation of Argumentation Formalisms', *Artificial Intelligence*, 171, 286–310.

Cayrol, C. (1995), 'On the Relation between Argumentation and Non-monotonic Coherence-based Entailment', in *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (*IJCAI 1995, Montréal, Canada*), Morgan Kaufmann, pp. 1443–1448.

Chesñevar, C., Maguitman, A., and Loui, R. (2000), 'Logical Models of Argument', *ACM Compututing Surveys*, 32, 337–383.

Creignou, N., Schmidt, J., and Thomas, M. (2010), 'Complexity of Propositional Abduction for Restricted Sets of Boolean Functions', in *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2010, Toronto, Canada*), eds. F. Lin, U. Sattler, and M. Truszczynski, AAAI Press (full review to appear in *Journal of Logic and Computation* 2011), pp. 8–16.

Creignou, N., and Vollmer, H. (2008), 'Boolean Constraint Satisfaction Problems: When Does Post's Lattice Help?', in *Complexity of Constraints – An Overview of Current Research Themes*, Vol. 5250 of *Lecture Notes in Computer Science*, eds. N. Creignou, P.G. Kolaitis, and H. Vollmer, Springer, pp. 3–37.

Creignou, N., and Zanuttini, B. (2006), 'A Complete Classification of the Complexity of Propositional Abduction', *SIAM Journal on Computing*, 36, 207–229.

Dung, P., Kowalski, R., and Toni, F. (2006), 'Dialectical Proof Procedures for Assumption-based Admissible Argumentation', *Artificial Intelligence*, 170, 114–159.

Dung, P.M. (1995), 'On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *n*-Person Games', *Artificial Intelligence*, 77, 321–358.

Durand, A., Hermann, M., and Kolaitis, P.G. (2005), 'Subtractive Deductions and Complete Problems for Counting Complexity Classes', *Theoretical Computer Science*, 340, 496–513.

Durand, A., and Hermann, M. (2008), 'On the Counting Complexity of Propositional Circumscription', *Information Processing Letters*, 106, 164–170.

Eiter, T., and Gottlob, G. (1995), 'The Complexity of Logic-based Abduction', *Journal of the ACM*, 42, 3–42.

García, A., and Simari, G. (2004), 'Defeasible Logic Programming: An Argumentative Approach', *Theory and Practice of Logic Programming*, 4, 95–138.

Gorogiannis, N., and Hunter, A. (2011), 'Instantiating Abstract Argumentation with Classical Logic Arguments: Postulates and Properties', *Artificial Intelligence*, 175, 1479–1497.

Hemaspaandra, L., and Vollmer, H. (1995), 'The Satanic Notations: Counting Classes Beyond #P and Other Definitional Adventures', *Complexity Theory Column 8, ACM-SIGACT News*, 26, 2–13.

Hermann, M., and Pichler, R. (2010), 'Counting Complexity of Propositional Abduction', *Journal of Computer and System Sciences*, 76, 634–649.

Hirsch, R., and Gorogiannis, N. (2010), 'The Complexity of the Warranted Formula Problem in Propositional Argumentation', *Journal of Logic and Computation*, 20, 481–499.

Johnson, D., Papadimitriou, C., and Yannakakis, M. (1988), 'On Generating All Maximal Independent Sets', *Information Processing Letters*, 27, 119–123.

Karchmer, M., and Wigderson, A. (1988), 'Monotone Circuits for Connectivity Require Super-logarithmic Depth', in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988, Chicago, Illinois, USA)*, ed. J. Simon, ACM, pp. 539–550.

Lewis, H. (1979), 'Satisfiability Problems for Propositional Calculi', *Mathematical Systems Theory*, 13, 45–53.

Nordh, G., and Zanuttini, B. (2008), 'What Makes Propositional Abduction Tractable', *Artificial Intelligence*, 172, 1245–1284.

Papadimitriou, C., (1994), *Computational Complexity*, Addison-Wesley.

Papadimitriou, C., and Wolfe, D. (1988), 'The Complexity of Facets Resolved', *Journal of Computer and System Sciences*, 37, 2–13.

Parsons, S., Wooldridge, M., and Amgoud, L. (2003), 'Properties and Complexity of Some Formal Inter-agent Dialogues', *Journal of Logic and Computation*, 13, 347–376.

Post, E. (1941), 'The Two-valued Iterative Systems of Mathematical Logic', *Annals of Mathematics Studies*, 5, 1–122.

Prakken, H., and Vreeswijk, G. (2002), 'Logical Systems for Defeasible Argumentation', in *Handbook of Philosophical Logic* ed. D. Gabbay, Kluwer.

Rahwan, I., and Simari, G. (eds.) (2009), *Argumentation in Artificial Intelligence*, Springer Verlag.

Reith, S. (2003), 'On the Complexity of some Equivalence Problems for Propositional Calculi', in *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS 2003, Bratislava, Slovakia)*, eds. B. Rovan and P. Vojtás, Vol. 2747 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 632–641.

Schnoor, H. (2005), 'The Complexity of the Boolean Formula Value Problem', Technical Report, Theoretical Computer Science, University of Hannover.

Spira, P.M. (1971), 'On Time-hardware Complexity Tradeoffs for Boolean Functions', in *Proceedings of the 4th Hawaii International Symposium on System Sciences*, pp. 525–527.

Thomas, M. (2009), 'The Complexity of Circumscriptive Inference in Post's Lattice', in *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009, Potsdam, Germany)*, eds. E. Erdem, F. Lin, and T. Schaub, Vol. 5753 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 290–302.

Wooldridge, M., Dunne, P., and Parsons, S. (2006), 'On the Complexity of Linking Deductive and Abstract Argument Systems', in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006, Boston, Massachusetts, USA)*, MIT Press, pp. 299–304.