Efficient Construction of Structured Argumentation Systems

Bruno Yun, Nir Oren, Madalina Croitoru

Computational Models of Argument
H. Prakken et al. (Eds.)

doi:10.3233/FAIA200525

Efficient Construction of Structured Argumentation Systems

Bruno YUN [a,1], Nir OREN [a] and Madalina CROITORU [b]
[a] *University of Aberdeen, Scotland*
[b] *University of Montpellier, France*

Abstract. We address the problem of efficient generation of structured argumentation systems. We consider a simplified variant of an ASPIC argumentation system and provide a backward chaining mechanism for the generation of structured argumentation graphs. We empirically compare the efficiency of this new approach with existing approaches (based on forward chaining) and characterise the benefits of using backward chaining for argumentation-based query answering.

Keywords. Argumentation, Backward chaining

1. Introduction

Logic-based argumentation is a powerful approach to reasoning with conflicting pieces of information. While argumentation traditionally generates arguments, most other approaches take a defeasible theory (DT) as input, together with a query. The output of the reasoning process is whether the query is, or is not, accepted by justified conclusions of the *saturated* DT, taking the interactions between conflicting facts into account. Most approaches to argumentation first compute all arguments, and detect conflicts between the arguments, thereby generating an argumentation graph [14, 19, 17]. Abstract argumentation semantics [8, 6] are then used to compute sets of justified arguments (called extensions), whose conclusions are compared to the query [7]. This follows the intuitions of what is known as *forward chaining* (FC) [15, 1]. The departure point of this work is the observation that FC is inappropriate for certain applications. Generating the entire set of arguments and identifying its extensions is computationally expensive [19, 18, 9], and one might want to answer a single query, in which case only arguments relevant to it should be generated. Here, we propose instead to use *backward chaining* (BC), focusing on an argumentation system (AS) based on a variant of ASPIC [4].

The focus of our approach is computational efficiency, though we recognise that in the worst case, the entire set of arguments may need to be generated to answer a query. However, we show in our empirical evaluation that this rarely occurs. We also analyse how the rules interact, proposing sufficient conditions to determine when, for a specific set of rules, fewer arguments will be generated by our approach.

[1] Corresponding Author: bruno.yun@abdn.ac.uk.

Our contributions are (1) the introduction of a new BC mechanism for ASPIC-like arguments; (2) a combinatorial structure that characterises the benefits of using BC over FC; and (3) an empirical evaluation demonstrating the potential impact of our approach.

The paper is structured as follows. In Section 2, we introduce the background notions. In Section 3, we introduce the *Graph of Rule Interaction*, a structure which captures how rules trigger each other. In Section 4, we describe how to generate arguments with BC. In Section 5, we show that the AS obtained via BC satisfies desirable properties and confirm them using an empirical evaluation. Section 6 summarises our approach.

2. Background

We begin by providing a brief overview of ASs built upon DTs. The notions presented here are used in the remainder of the paper. An argumentation formalism is usually built around an underlying logical language $\mathscr{L}$. We assume that $\mathscr{L}$ is made up of a set of literals and that we possess classical negation, i.e., we have a function "$\neg$" s.t. $\neg\psi = \phi$ iff $\psi = \neg\phi$ and $\neg\psi = \neg\phi$ iff $\psi = \phi$. A *strict rule* is then an expression $\mu : \phi_1, \ldots, \phi_n \rightarrow \psi$ and a *defeasible rule* is an expression $\mu : \phi_1, \ldots, \phi_n \Rightarrow \psi$ with $n \geq 0$ and $\{\phi_1, \ldots, \phi_n, \psi\} \subseteq \mathscr{L}$. For a rule $r = \phi_1, \ldots, \phi_n \rightsquigarrow \psi$, where $\rightsquigarrow \in \{\rightarrow, \Rightarrow\}$, we denote by $Body(r), Head(r), Name(r)$ and $Imp(r)$ the set $\{\phi_1, \ldots, \phi_n\}$, the literal ψ, the literal μ, and $\rightsquigarrow$ respectively. A rule $\phi_1, \ldots, \phi_n \rightsquigarrow \psi$ is said to be *applicable* on a set of literals $\mathscr{P} \subseteq \mathscr{L}$ iff $Body(r) \subseteq \mathscr{P}$. A set $\mathscr{P} \subseteq \mathscr{L}$ is said to be *consistent* iff there is no $\phi, \psi \in \mathscr{P}$ s.t. $\phi = \neg\psi$. The *closure* of $\mathscr{P}$ under a set of strict rules $\mathscr{S}$, denoted by $Cl_{\mathscr{S}}(\mathscr{P})$, is the minimal set s.t. (1) $\mathscr{P} \subseteq Cl_{\mathscr{S}}(\mathscr{P})$ and (2) for all strict rules $r = \phi_1, \ldots, \phi_n \rightarrow \psi$ in $\mathscr{S}$ s.t. r is applicable to $Cl_{\mathscr{S}}(\mathscr{P})$, it holds that $\psi \in Cl_{\mathscr{S}}(\mathscr{P})$. A DT is $\mathscr{T} = (\mathscr{S}, \mathscr{D})$ where $\mathscr{S}$ is a set of strict rules and $\mathscr{D}$ is a set of defeasible rules [2].

Example 1. *We consider a DT containing the following information about John, a patient in a hospital: r_1 : "John has a prostate cancer" ($\rightarrow c$); r_2 : "John is following a treatment for his cancer" ($\rightarrow t$); r_3 : "John is a male patient" ($\rightarrow m$); r_4 : "Studies show that there is no correlation between treating prostate cancer and bowel disorders" ($\rightarrow \neg r_7$); r_5 : "John does not have abdominal pains" ($\Rightarrow a$); r_6 : "If John does not have abdominal pain then he may not suffer from bowel disorder" ($a \Rightarrow \neg b$); r_7 : "A patient with prostate cancer that is under treatment may suffer from bowel disorders" ($c, t \Rightarrow b$). The DT is modelled by $\mathscr{T} = (\mathscr{S}, \mathscr{D})$ s.t. $\mathscr{S} = \{r_1, r_2, r_3, r_4\}$ and $\mathscr{D} = \{r_5, r_6, r_7\}$.*

Our AS is based on a version of ASPIC [4]. Here, an argument are formed by applying deductive rules [3], built upon other arguments. Given $\mathscr{T} = (\mathscr{S}, \mathscr{D})$, an argument A is of the form $A_1, \ldots, A_n \rightsquigarrow \psi$, where $\{A_1, \ldots A_n\}$ is a minimal set of arguments s.t. there exists an $r \in \mathscr{S} \cup \mathscr{D}$, where r is applicable to $\{Conc(A_1), \ldots, Conc(A_n)\}$, $\rightsquigarrow = Imp(r)$ and $\psi = Head(r)$. Let $A = A_1, \ldots, A_n \rightsquigarrow \psi$. Then the conclusion of A, denoted by $Conc(A)$, is ψ and the set of sub-arguments of A is $Sub(A) = Sub(A_1) \cup \cdots \cup Sub(A_n) \cup \{A\}$. The top rule of A is $TR(A) = Conc(A_1), \ldots, Conc(A_n) \rightsquigarrow \psi$. Given $\mathscr{T} = (\mathscr{S}, \mathscr{D})$, $Name : \mathscr{S} \cup \mathscr{D} \rightarrow \mathscr{L}$ returns the name of each rule and provides a handle for rules to prevent other rule applications.

[2] Please note that in our formalism, axioms (resp. ordinary premises) [14] are represented using strict (resp. defeasible) rules with empty bodies (c.f., ASPIC- [5])

Example 2 (Cont'd). *There are 7 arguments:* $A_1 =\!\!\rightarrow c$, $A_2 =\!\!\rightarrow t$, $A_3 =\!\!\rightarrow m$, $A_4 =\!\!\Rightarrow h$, $A_5 =\!\!\rightarrow \neg r_7$, $A_6 = A_1, A_2 \Rightarrow b$ *and* $A_7 = A_4 \Rightarrow \neg b$. *Note that* $Name(c, d \Rightarrow b) = r_7$.

We consider that we are given a binary, total, reflexive and transitive preference relation $\preceq$ over arguments that reflects the quality of its underlying elements. One way of computing such a relation is to consider the type of rules (defeasible or strict) that are used in an argument and/or that defeasible rules have an associated strength, and *lifting* these preferences from rules to arguments [14], but we do not consider these aspects here. We write $A \prec B$ iff $A \preceq B$ and $B \not\preceq A$, and $A \sim B$ iff $A \preceq B$ and $B \preceq A$.

A defeats B iff $B \preceq A$ and at least one of the following conditions is satisfied: (**Rebutting**) there exists $B' \in Sub(B)$ such that $Conc(B') = \neg Conc(A)$ and $Imp(TR(B'))$ is $\Rightarrow$[3] or (**Undercutting**) $Conc(A) = \neg Name(TR(B))$

An AS for a $\mathscr{T}$, denoted $\mathbb{AS}_{\mathscr{T}}$, is $(\mathscr{A}, \textsc{Def})$ where $\mathscr{A}$ is the set of arguments generated and $\textsc{Def} \subseteq \mathscr{A} \times \mathscr{A}$ is the defeat relation introduced above. Let $\mathbb{AS} = (\mathscr{A}, \textsc{Def})$, we say that $\mathbb{AS}' = (\mathscr{A}', \textsc{Def}')$ is a sub-system of $\mathbb{AS}$ iff $\mathscr{A}' \subseteq \mathscr{A}, \textsc{Def}' = \mathscr{A}' \times \mathscr{A}' \cap \textsc{Def}$.

Example 3 (Cont'd). *If we assume that* $A_7 \prec A_6$ *and* $A_6 \preceq A_5$ *then* A_6 *is defeated by* A_5 *(undercutting) and* A_6 *defeats (rebutting)* A_7 *but* A_7 *does not defeat* A_6. *The AS corresponding to* $\mathscr{T}$ *is* $\mathbb{AS}_{\mathscr{T}} = (\{A_1, \ldots, A_7\}, \{(A_5, A_6), (A_6, A_7\})$.

3. The Graph of Rule Interaction

We define a new combinatorial structure over a DT called the Graph of Rule Interaction (GRI) that generalises the Graph of Rule Dependency [2]. The GRI captures both how rules trigger each other and identifies the possible conflicts between them. There are three main elements to the GRI: (1) a set of nodes representing the rules, (2) a set of *support links* representing how rules can activate each other and (3) a set of *attack links* representing conflicts amongst rules. While attacks links are binary, support links are many-to-one relationships as multiple rules can be necessary to trigger a rule.

Definition 1 (Graph of Rule Interaction). *Let* $\mathscr{T} = (\mathscr{S}, \mathscr{D})$. *The GRI of* $\mathscr{T}$ *is* $GRI_{\mathscr{T}} = (\mathscr{N}, \mathscr{R}_s, \mathscr{R}_d)$, *where: (A)* $\mathscr{N} = \mathscr{S} \cup \mathscr{D} \cup \{\emptyset\}$ *represents the rules under consideration; (B)* $\mathscr{R}_s \subseteq 2^{\mathscr{N}} \times \mathscr{N}$ *s.t.* $\forall n_1 \in \mathscr{N}$ *and* $\forall N \subseteq \mathscr{N}, (N, n_1) \in \mathscr{R}_s$ *iff* $|N| = |Body(n_1)|$ *and* $\bigcup_{n \in N} Head(n) = Body(n_1)$. $\mathscr{R}_s$ *captures the support between rules and (C)* $\mathscr{R}_d \subseteq \mathscr{N} \times \mathscr{N}$ *s.t.* $\forall n_1, n_2 \in \mathscr{N}, (n_1, n_2) \in \mathscr{R}_d$ *iff at least one of the following conditions holds: (1)* $Head(n_1) = \neg Head(n_2)$ *and* $Imp(n_2) =\!\!\Rightarrow$ *or (2)* $Head(n_1) = \neg Name(n_2)$. $\mathscr{R}_d$ *captures potential defeats between rules. Note that* $Head(\emptyset) = Body(\emptyset) = Name(\emptyset) = \emptyset$.

Chains of rules can form where one rule is required for another to be applied, meaning that multiple rules can *support* others. We formalise this notion within the GRI.

Definition 2 (Support path). *Let* $\mathscr{T} = (\mathscr{S}, \mathscr{D})$ *and* $GRI_{\mathscr{T}} = (\mathscr{N}, \mathscr{R}_s, \mathscr{R}_d)$. *The sequence* $(\mathscr{N}_1, \mathscr{N}_2, \ldots, \mathscr{N}_k)$ *is a support path to* $n \in \mathscr{S} \cup \mathscr{D}$ *in* $GRI_{\mathscr{T}}$ *iff all the following conditions are satisfied: (1)* $\forall i \in \{1, \ldots, k\}$, $\mathscr{N}_i \subseteq 2^{\mathscr{N}}$, *(2)* $\mathscr{N}_1 = \{\emptyset\}$ *and* $\mathscr{N}_k = \{\{n\}\}$ *and (3)* $\forall i \in \{2, \ldots, k\}$, $\forall N_j \in \mathscr{N}_i$ *and* $\forall n' \in N_j$, *there exists* $N' \in \mathscr{N}_{i-1}$ *s.t.* $(N', n') \in \mathscr{R}_s$.

[3] Note that we use *restricted rebut* and we did not evaluate unrestricted rebut due to space limitations.

An *activated* rule is a rule with a support path to it. The underlying idea is that such a rule's body can be obtained from other rules via the support path to it.

Definition 3 (Activated & Connected rule)**.** *Let $\mathscr{T} = (\mathscr{S}, \mathscr{D})$, $GRI_{\mathscr{T}} = (\mathscr{N}, \mathscr{R}_s, \mathscr{R}_d)$ and $n, n' \in \mathscr{N}$. $r \in \mathscr{S} \cup \mathscr{D}$ is activated iff there exists a support path to r in $GRI_{\mathscr{T}}$. n is connected to n' iff there exists a sequence $(n_1, \ldots, n_k)$ s.t. both of the following are satisfied: (1) for every $1 \leq i \leq k$, $n_i \in \mathscr{N}$ and n_i is activated and (2) for every $1 \leq i \leq k-1$, it holds that either $(n_i, n_{i+1}) \in \mathscr{R}_d$ or there exists $N \subseteq 2^{\mathscr{N}}$ s.t. $(N, n_{i+1}) \in \mathscr{R}_s$ with $n_i \in N$.*

Reasoning with BC requires a query that will be used to select the necessary rules in the original DT. We thus need to describe whether a rule is important for a given query. We refer to such rules as *potentially necessary* rules.

Definition 4 (Potentially necessary rule)**.** *Let $l \in \mathscr{L}$ and $\mathscr{T} = (\mathscr{S}, \mathscr{D})$. $r \in \mathscr{S} \cup \mathscr{D}$ is potentially necessary for l iff $\exists r' \in \mathscr{S} \cup \mathscr{D}$ s.t. $Head(r') = l$ and r is connected to r'.*

Example 4 (Cont'd)**.** *$GRI_{\mathscr{T}} = (\mathscr{N}, \mathscr{R}_s, \mathscr{R}_d)$ where $\mathscr{N} = \{\emptyset, r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$, $\mathscr{R}_s = \{(\{\emptyset\}, r_1), (\{\emptyset\}, r_2), (\{\emptyset\}, r_3), (\{\emptyset\}, r_4), (\{\emptyset\}, r_5), (\{r_1, r_2\}, r_7), (\{r_5\}, r_6)\}$ and $\mathscr{R}_d = \{(r_4, r_7), (r_6, r_7), (r_7, r_6)\}$. The sequence $(\{\emptyset\}, \{\{r_1\}, \{r_2\}\}, \{\{r_7\}\})$ is a support path to r_7. The rule r_5 is potentially necessary for b but r_3 is not.*

4. Backward Chaining for Argumentation

An argument for $l \in \mathscr{L}$ is an argument that concludes l. This notion is needed to define what an AS for a literal is.

Definition 5 (Argument for a literal)**.** *Let $l \in \mathscr{L}$ and $\mathbb{AS}_{\mathscr{T}} = (\mathscr{A}, \text{DEF})$. $A \in \mathscr{A}$ is an argument for l iff $Conc(A) = l$. $\mathscr{A}_l \subseteq \mathscr{A}$ is the set of arguments for l in $\mathscr{A}$.*

Definition 6 (AS for a literal)**.** *Let $l \in \mathscr{L}$ and $\mathbb{AS}_{\mathscr{T}} = (\mathscr{A}, \text{DEF})$. We say that the sub-system $\mathbb{AS}^l_{\mathscr{T}} = (\mathscr{A}', \text{DEF}')$ of $\mathbb{AS}_{\mathscr{T}}$ is the AS for l (w.r.t. $\mathscr{T}$) iff $\mathscr{A}'$ is minimal (w.r.t. $\subseteq$) s.t. all the following are satisfied: **(1)** $\mathscr{A}_l \subseteq \mathscr{A}'$, **(2)** If $A \in \mathscr{A}'$ and $B \in \mathscr{A}$ s.t. $(B, A) \in \text{DEF}$ then $B \in \mathscr{A}'$, **(3)** If $A \in \mathscr{A}'$ then $Sub(A) \subseteq \mathscr{A}'$ and (4) $\text{DEF}' = \{(A, B) \in \text{DEF} \mid A, B \in \mathscr{A}'\}$.*

Note that for a given DT $\mathscr{T}$, there is a *unique* AS for $l \in \mathscr{L}$ w.r.t. $\mathscr{T}$. Moreover, since $\mathbb{AS}^l_{\mathscr{T}} = (\mathscr{A}', \text{DEF}')$ is a sub-system of $\mathbb{AS}_{\mathscr{T}} = (\mathscr{A}, \text{DEF})$, it has at most the same number of arguments and attacks, i.e. $|\mathscr{A}| \geq |\mathscr{A}'|$ and $|\text{DEF}| \geq |\text{DEF}'|$.

Example 5 (Cont'd)**.** *$\mathbb{AS}^{\neg b}_{\mathscr{T}} = (\mathscr{A}', \text{DEF}')$ where $\mathscr{A}' = \{A_1, A_2, A_4, A_5, A_6, A_7\}$ and $\text{DEF}' = \{(A_5, A_6), (A_6, A_7)\}$. $\mathbb{AS}^{\neg b}_{\mathscr{T}}$ is the AS for $\neg b$ w.r.t. $\mathscr{T}$.*

We now focus on the generation of arguments from a DT, describing a two-step procedure for generating the AS using BC, i.e. from $\mathscr{T}$, we use BC to generate $\mathbb{AS}^l_{\mathscr{T}}$, for any $l \in L$. Our approach makes use of two algorithms: AL and ASG (see Figure 1).

AL: This algorithm generates $\mathscr{A}_l$. This procedure gathers all the rules that conclude l in R (line 5). It then constructs all the possible arguments that have a rule of R as top rule. (lines 6–9). Note that the parameter *seen* is used to avoid the generation of an infinite number of sub-arguments in case of rule cycles (for instance $p \to q$ and $q \to p$).

Require: $\mathscr{T} = (\mathscr{S}, \mathscr{D})$, $l \in \mathscr{L}$ and $seen \subseteq \mathscr{L}$
1: **function** AL($\mathscr{T}, l, seen$)
2: $A \leftarrow \emptyset$
3: **if** $l \in seen$ **then**
4: **return** $\emptyset$
5: $R \leftarrow \{r \in \mathscr{S} \cup \mathscr{D} \mid Head(r) = l\}$
6: **for** $r \in R$ **do**
7: $\mathscr{A}_B \leftarrow \emptyset$
8: **for** $\phi \in Body(r)$ **do**
9: $\mathscr{A}_B[\phi] = \text{AL}(\mathscr{T}, \phi, seen \cup \{l\})$
10: $Prod \leftarrow \bigtimes_{\phi \in Body(r)} \mathscr{A}_B[\phi]$
11: **for** $\{A_1, \ldots, A_n\} \in Prod$ **do**
12: $\rightsquigarrow_r \leftarrow Imp(r)$
13: $A \leftarrow A_1, \ldots A_n \rightsquigarrow_r Head(r)$
14: $\mathscr{A} \leftarrow \mathscr{A} \cup \{A\}$
15: **return** $\mathscr{A}$

Require: $\mathscr{T} = (\mathscr{S}, \mathscr{D})$ and $l \in \mathscr{L}$
1: **function** ASG($\mathscr{T}, l$)
2: $finished \leftarrow false$
3: $\mathscr{A}_{old} \leftarrow \text{AL}(\mathscr{T}, l, \emptyset)$
4: **while** not $finished$ **do**
5: $temp \leftarrow \mathscr{A}_{old}$
6: $\mathscr{A} \leftarrow temp$
7: **for all** $A \in temp$ **do**
8: $\mathscr{A} \leftarrow \mathscr{A} \cup Sub(A)$
9: $temp \leftarrow \mathscr{A}$
10: **for all** $A \in temp$ **do**
11: $\mathscr{A} \leftarrow \mathscr{A} \cup \text{AL}(\mathscr{T}, \neg Head(TR(A)), \emptyset)$
12: $\mathscr{A} \leftarrow \mathscr{A} \cup \text{AL}(\mathscr{T}, \neg Name(TR(A)), \emptyset)$
13: **if** $\mathscr{A} = \mathscr{A}_{old}$ **then**
14: $finished \leftarrow true$
15: **else**
16: $\mathscr{A}_{old} \leftarrow \mathscr{A}$
17: DEF $\leftarrow$ DEF-GENERATE($\mathscr{T}, \mathscr{A}, \preceq$)
18: $(\mathscr{A}, \text{DEF}) \leftarrow$ AS-FILTER($\mathscr{A}$, DEF)
19: **return** $(\mathscr{A}, \text{DEF})$

Figure 1. Algorithms to generate $\mathscr{A}_l$ (left) and $\mathbb{AS}^l_{\mathscr{T}}$ (right)

ASG: This algorithm generates $\mathbb{AS}^l_{\mathscr{T}}$. It starts by generating $\mathscr{A}_l$, after which it successively adds sub-arguments and attacking arguments to the existing ones (lines 10-12). Once the arguments have been generated, `DEF-generate` computes the defeat relation — as described in Section 2 — by comparing pairs of arguments. As attacking arguments are not always defeaters (depending on the $\preceq$ relation chosen), we have to filter unnecessary arguments that do not fit the conditions of Definition 6. This is done by using calling the `AS-filter` (line 18) to perform the filtering.

5. Evaluation

Following Dung [8], let $\mathbb{AS} = (\mathscr{A}, \text{DEF})$ be an AF and $\varepsilon \subseteq \mathscr{A}$. We say that ε is *conflict-free* iff there is no $a, b \in \varepsilon$ s.t. $(a,b) \in \text{DEF}$. ε *defends* a iff for every $b \in \mathscr{A}$ s.t. $(b,a) \in \text{DEF}$, there exists $c \in \varepsilon$ s.t. $(c,b) \in \text{DEF}$. ε is *admissible* iff it is conflict-free and defends all its arguments. ε is a *complete extension* iff ε is admissible and contains all the arguments it defends. ε is a *preferred extension* iff it is a maximal (w.r.t. $\subseteq$) admissible set. ε is the *grounded extension* iff ε is a minimal (w.r.t. $\subseteq$) complete extension. We denote by $Ext(x,y)$ the function that returns the set of extensions of the AS x w.r.t. y, where $y \in \{pr, gr\}$ and pr (resp. gr) stands for the preferred (resp. grounded) semantics. Likewise, $Acc(x,y)$ is returns the accepted arguments of the AS x w.r.t. the semantics y.

Definition 7 (Status)**.** *Let $\mathbb{AS} = (\mathscr{A}, \text{DEF})$ and $a \in \mathscr{A}$. a is* accepted *w.r.t. the preferred semantics (resp. grounded semantics) if $\forall E \in Ext(\mathbb{AS}, pr)$ (resp. $Ext(\mathbb{AS}, gr)$), $a \in E$. a is* rejected *w.r.t. the preferred semantics (resp. grounded semantics) if $\forall E \in Ext(\mathbb{AS}, pr)$ (resp. $Ext(\mathbb{AS}, gr)$), $a \notin E$ and a is* undecided *if it is neither accepted nor rejected.*

Definition 8 (Acceptability of a literal)**.** *Let $l \in \mathscr{L}$, $\mathscr{T}$ and $\mathbb{AS} = (\mathscr{A}, \text{DEF})$. l is accepted w.r.t. the preferred (resp. grounded) semantics and $\mathbb{AS}$ iff there exists an $a \in \mathscr{A}$ s.t. $Conc(a) = l$ and $a \in Acc(\mathbb{AS}, pr)$ (resp. $a \in Acc(\mathbb{AS}, gr)$). Otherwise, l is rejected.*

Example 6 (Cont'd). *$Ext(\mathbb{AS}_{\mathscr{T}}, pr) = Ext(\mathbb{AS}_{\mathscr{T}}, gr) = \{\{A_1, A_2, A_3, A_4, A_5, A_7\}\}$ and $Ext(\mathbb{AS}^{\neg b}_{\mathscr{T}}, pr) = \{\{A_1, A_2, A_4, A_5, A_7\}\}$. Since $A_7 \in Acc(\mathbb{AS}^{\neg b}_{\mathscr{T}}, pr)$, $\neg b \in Acc_{\mathscr{L}}(\mathbb{AS}^{\neg b}_{\mathscr{T}}, pr)$. Note that m is accepted w.r.t. preferred/grounded in $\mathbb{AS}_{\mathscr{T}}$ but rejected in $\mathbb{AS}^{\neg b}_{\mathscr{T}}$.*

Proposition 1. *Let $l \in \mathscr{L}$, $\mathbb{AS}_{\mathscr{T}}$ be the AS for $\mathscr{T}$ and $\mathbb{AS}^{l}_{\mathscr{T}}$ be the AS for l. It holds that $l \in Acc_{\mathscr{L}}(\mathbb{AS}^{l}_{\mathscr{T}}, y)$ iff $l \in Acc_{\mathscr{L}}(\mathbb{AS}_{\mathscr{T}}, y)$, where $y \in \{pr, gr\}$.*

We now show that (1) in specific DTs (characterised via a sufficient condition) the AS for a literal has strictly fewer arguments than the corresponding original AS, (2) the rules that are not activated are not taken into account when constructing the arguments of an AS and (3) the GRI can be used to filter a DT $\mathscr{T}$ prior to the generation of $\mathbb{AS}^{l}_{\mathscr{T}}$.

Proposition 2. *Let $\mathscr{T} = (\mathscr{S}, \mathscr{D})$, $l \in \mathscr{L}$, $\mathbb{AS}_{\mathscr{T}} = (\mathscr{A}, \text{DEF})$, $\mathbb{AS}^{l}_{\mathscr{T}} = (\mathscr{A}', \text{DEF}')$ be the AS for l. If there exists $r \in \mathscr{S} \cup \mathscr{D}$ such that:*

- *r is activated and not potentially necessary for l then $|\mathscr{A}'| < |\mathscr{A}|$.*
- *r is not activated then $\mathbb{AS}_{\mathscr{T}} = \mathbb{AS}_{\mathscr{T}'}$, where $\mathscr{T}' = (\mathscr{S} \setminus \{r\}, \mathscr{D} \setminus \{r\})$.*
- *r is not potentially necessary for l then $\mathbb{AS}^{l}_{\mathscr{T}} = \mathbb{AS}^{l}_{\mathscr{T}'}$ where $\mathbb{AS}^{l}_{\mathscr{T}'}$ is the AS for l w.r.t. $\mathscr{T}' = (\mathscr{S} \setminus \{r\}, \mathscr{D} \setminus \{r\})$.*

The third item of Proposition 2 shows that filtering DTs to only keep potentially necessary rules is possible when generating the AS for a literal. If $\mathscr{T}$ contains rules that are not potentially necessary for a literal, this filtering reduces the time taken to answer a query. Moreover, the GRI only has to be computed once, and can then be stored in memory and reused for multiple queries. The proposed framework is inspired from previous approaches based on DT pre-processing [18].

Empirical Evaluation

To test our approach, we use existing benchmarks and DTs to compare the effectiveness of reasoning using BC and FC in the context of argumentation. To this end, we use existing DTs [12] as we are not aware of other standard benchmarks for instantiated argumentation. Due to space constraints, we only considered four theories from that work (tree, level, levels and teams). In **tree(n,k)**, the rules form a k-branching tree of depth n where the literal p_0 is the root. In these theories, every literal occurs only once. In **level(n)**, there is a cascade of n disputed conclusions, i.e. there are rules $\Rightarrow p_i$ and $p_{i+1} \Rightarrow \neg p_i$, for $0 \leq i \leq n$. In **levels(n)**, for odd i, the latter rule has a superior strength when compared to even rules. Finally, in **teams(n)**, every literal is disputed with two rules for p_i and two rules for $\neg p_i$, and the rules for p_i are superior to the rules for $\neg p_i$. To obtain the $\preceq$, we use the last-link principle described in [14].

For the FC procedure, we generated all arguments (5 times) using a breadth-first naive approach. For the BC procedure, for all theories, we randomly selected ten literals and generated the ASs for those literals. An upper limit of 200 minutes was set for all runs. Table 1 is split in three parts: FC, BC and DT filtration (by removing non-potentially necessary rules). In the *Forward* columns, we depict the mean time, the number of arguments generated, and the number of defeats of the graph[4]. In the *Backward* columns, we show, across all literals on non-timed out instances, the mean time for the generation

[4]Time does not include defeat generation, their number is calculated based on the structure of the theory.

Theory	Forward			Backward			Filter		
	Mean time (s)	# args.	# defeats	Mean time (s)	Mean # arguments	# Succ. instances	Mean # rules	% Filtration	Mean time (ms)
tree(n,k)									
$n = 8, k = 3$	5712.7	59049	0	19.6	15.8	10	11	99.93%	484.6
$n = 9, k = 3$	Timeout	196830	0	53.5	3.5	10	3.5	99.99%	810.5
level(n)									
$n = 10$	0.1	19	26	0.03	10.8	10	10.8	43.16%	41
$n = 1000$	6.7	1999	2996	830.5	716.6	10	716.6	64.15%	132.60
$n = 5000$	181.7	9999	14996	3927.2	602.3	3	3386	66.14%	302.30
$n = 10000$	678.9	19999	29996	Timeout	-	0	8423	57.88%	490.40
levels(n)									
$n = 10$	0.1	19	18	0.04	14	10	14	26.32%	46.5
$n = 1000$	6.7	1999	1998	1245.3	841.2	10	841.2	57.92%	160.9
$n = 5000$	155.9	9999	9998	96542.02	3702	2	5804.4	41.95%	451.6
$n = 10000$	696.8	19999	19998	428.5	163	1	10798.2	46.0%	555
teams(n)									
$n = 3$	0.4	176	272	0.26	3.1	10	3.1	97.89%	88
$n = 4$	1.6	736	1568	1.62	19.9	10	17.1	97.1%	131.2
$n = 5$	26.8	3008	8256	5.28	23.8	10	20.6	99.14%	198.3
$n = 6$	539.7	12160	254335	18.35	3.8	10	3.8	99.96%	369.2
$n = 7$	11613.2	48896	1401159	84.07	14.1	10	12.9	99.97%	866.5

Table 1. Summary of the empirical evaluation

of arguments, the mean number of arguments in the AS for the literal and the number of successful (non -timeout) instances. In the *Filter* columns, we show the number of rules after the filtration, the percentage of the number of rules filtered, and the mean time used for obtaining the filtered DT. We make three important observations: **(1)** For the tree and teams DTs, all the runs were successful and the BC was significantly faster in generating the arguments than the FC. It is worth noting that while the FC procedure times out after $n = 9$, the BC procedure is able to provide an answer in less than 5 minutes. (2) For the level DTs, the BC takes longer than the FC (even if it generates fewer arguments). From $n = 5000$ onward, most instances timeout. Note that the BC takes longer than the FC for these instances because it checks and generates all the arguments that can potentially attack the existing arguments. (3) In the tree and teams DTs, we obtain fewer arguments with the BC compared to the FC. The gap in the number of arguments means that the process of verification does not cause a serious overhead in the computation time.

6. Discussion and Future Work

We introduced the notion of BC argumentation and illustrated our approach with an ASPIC-style structured AS. We analysed the links between the AS generated using the BC procedure and the FC procedure w.r.t. argumentation semantics and showed an empirical comparison of the time needed to generate the arguments for both procedures.

Our work is motivated by the need for efficient query answering frameworks that do not need to generate the whole set of arguments [19, 18]. Our work relates with existing BC-based works such as DeLP [10] or ABA [16]. However, our focus is explicitly on ASPIC-like systems. There are also similarities between BC and proof dialogues [13], though most such dialogues operate on abstract ASs.

We have identified several potential avenues of future work. First, we intend to create additional benchmarks for instantiated ASs by replicating the properties of existing DTs [11]. Second, we recognise that there are similarities between the process we use, and different search algorithms. We intend to evaluate these different strategies, as well as

heuristics for guiding the expansion process, and their effects on performance. Finally, integrating lifting rules for preferences (e.g. weakest link, elitist or democratic orderings [14]) could provide optimisations regarding argument expansion.

References

[1] O. Arieli, A. Borg, and C. Straßer. Prioritized Sequent-Based Argumentation. In *AAMAS 2018*, pages 1105–1113, 2018.

[2] J.-F. Baget, F. Garreau, M.-L. Mugnier, and S. Rocher. Extending Acyclicity Notions for Existential Rules. In *ECAI-14*, pages 39–44, 2014.

[3] P. Besnard and A. Hunter. *Elements of Argumentation.* MIT Press, 2008.

[4] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artif. Intell.*, 171(5-6):286–310, 2007.

[5] M. Caminada, S. Modgil, and N. Oren. Preferences and unrestricted rebut. In *Proc. COMMA*, pages 209–220, 2014.

[6] M. Caminada and B. Verheij. On the existence of semi-stable extensions. *argumentation*, 3:4, 2010.

[7] M. Croitoru and S. Vesic. What Can Argumentation Do for Inconsistent Ontology Query Answering? In *SUM 2013*, pages 15–29, 2013.

[8] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artif. Intell.*, 77(2):321–358, 1995.

[9] P. E. Dunne and M. Wooldridge. Complexity of Abstract Argumentation. In *Argumentation in Artificial Intelligence*, pages 85–104. 2009.

[10] A. J. García and G. R. Simari. Defeasible Logic Programming: An Argumentative Approach. *TPLP*, 4(1-2):95–138, 2004.

[11] B. Konat, J. Lawrence, J. Park, K. Budzynska, and C. Reed. A Corpus of Argument Networks: Using Graph Properties to Analyse Divisive Issues. In *Proc. 10th Int'l Conf. on Language Resources and Evaluation*, 2016.

[12] M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient Defeasible Reasoning Systems. *Int. J. on A.I. Tools*, 10(4):483–501, 2001.

[13] S. Modgil and M. Caminada. Proof Theories and Algorithms for Abstract Argumentation Frameworks. In *Argumentation in Artificial Intelligence*, pages 105–129. 2009.

[14] S. Modgil and H. Prakken. The ASPIC+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[15] E. Salvat and M.-L. Mugnier. Sound and Complete Forward and backward Chaining of Graph Rules. In *ICCS 1996*, pages 248–262, 1996.

[16] F. Toni. A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117, 2014.

[17] B. Yun, M. Croitoru, S. Vesic, and P. Bisquert. DAGGER: Datalog+/- Argumentation Graph GEneRator. In *AAMAS 2018*, pages 1841–1843, 2018.

[18] B. Yun, S. Vesic, and M. Croitoru. Toward a More Efficient Generation of Structured Argumentation Graphs. In *COMMA 2018*, 2018.

[19] B. Yun, S. Vesic, M. Croitoru, P. Bisquert, and R. Thomopoulos. A Structural Benchmark for Logical Argumentation Frameworks. In *IDA*, pages 334–346, 2017.