Structural Properties for Deductive Argument Systems

Anthony Hunter[1] and Stefan Woltran[2]

[1] Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK
[2] Institute of Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria

Abstract. There have been a number of proposals for using deductive arguments for instantiating abstract argumentation. These take a set of formulae as a knowledgebase, and generate a graph where each node is a logical argument and each arc is a logical attack. This then raises the question of whether for a specific logical argument system S, and for any graph G, there is a knowledgebase such that S generates G. If it holds, then it can be described as a kind of "structural" property of the system. If it fails then, it means that there are situations that cannot be captured by the system. In this paper, we explore some features, and the significance, of such structural properties.

1 Introduction

Abstract argumentation provides a clear and precise approach to formalizing aspects of argumentation. However, in the approach, arguments are treated as atomic. If we want to understand individual arguments, we need to provide content for them. This leads to the idea of "instantiating" abstract argumentation with logical arguments, such as proposed by Cayrol [1].

There are various ways that logical arguments can be defined. A simple kind is a **deductive argument** which is a tuple $\langle \Phi, \alpha \rangle$ where Φ is a set of premises, and α is a claim such that for a consequence relation $\vdash_i$, $\Phi \vdash_i \alpha$ holds. Further constraints include consistency (i.e. $\Phi \nvdash_i \bot$) and minimality (there is no $\Psi \subset \Phi$ s.t. $\Psi \vdash_i \alpha$). Pollock was perhaps the first proponent of deductive arguments [2]. Subsequently, deductive arguments for classical logic [1, 3–5], description logic [6], and defeasible logic [7, 8] have been proposed.

In this paper, we consider how deductive argument systems generate constellations of arguments and counterarguments, and in particular, we are interested in the class of argument graphs they can induce. As we will see, some deductive argument systems only generate certain subclasses of graph. This is, however, not necessarily bad news. In fact, it is known that the computational complexity of evaluating argumentation frameworks (when considered as abstract frameworks following Dung [9]) can be decreased if the class of graphs is restricted, for instance to acyclic, bipartite or symmetric graphs or to graphs which have certain parameters (like treewidth) fixed (see e.g. [10–13]).

2 Preliminaries

In this section we review some established definitions for graph theory, for logical languages and inference, and for logical argumentation.

A graph G is a tuple of the form (N, E) where N is a set of nodes and E is a set of edges. If a graph has no nodes and no edges, then it is the **empty graph**, denoted $G_\emptyset$. We consider various graph types including the following: A graph (N, E) is **weakly connected** iff for all nodes $n, n' \in N$ there is an undirected path (i.e. ignoring the direction of the arrows) in E from n to n'; A graph (N', E') is a **component** of a graph (N, E) iff (N', E') is the maximum subgraph of (N, E) that is weakly connected; A graph (N, E) is a **self-loop graph** iff there is a node $n \in N$ such that there is an edge $(n, n) \in E$; A graph (N, E) is a **rooted graph** iff (N, E) is acyclic and there is a node n in N, called the **root**, such that for all other nodes m in N, there is a path from m to n; A graph (N, E) is a **tree** iff (N, E) is a rooted graph and for each non-root node n in N, there is a unique path from n to the root; A graph (N, E) is **complete bipartite** iff there is a partition N_1 and N_2 of N such that $E = \{(n_1, n_2), (n_2, n_2) \mid n_1 \in N_1 \text{ and } n_2 \in N_2\}$. And a graph (N, E) is a **rational graph** iff (N, E) is a component and (N, E) is not a self-loop graph. The set of all graphs is denoted `Graphs`, with various subsets including the following: `Components` which is the set of all connected graphs; `Trees` which is the set of all trees; `AcyclicGraphs` which is the set of all acyclic graphs; `Bipartites` which is the set of all complete bipartite graphs; `RootedGraphs` which is the set of rooted graphs; and `RationalGraphs` which is the set of rational graphs.

In general, we use `Formulae` to denote the set of formulae of a language. In this paper, we focus on two languages. The **language of defeasible formulae**, denoted `DefFormulae`, is the set of literals and the set of rules of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta$ where $\alpha_1, \ldots, \alpha_n$ are literals, and β is a literal. The **language of propositional formulae**, denoted `PropFormulae`, is the usual language for classical propositional logic that can formed from the logical connectives of $\vee$, $\wedge$, $\neg$ and $\rightarrow$. We consider the **classical consequence relation**, denoted $\vdash$, which is defined as usual, and the **defeasible consequence relation**, denoted $\vdash_d$, which is defined as follows: For $\Delta \subseteq$ `DefFormulae`, if $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta \in \Delta$, and for each $\alpha_i \in \{\alpha_1, \ldots, \alpha_n\}$, either $\alpha_i \in \Delta$ or $\Delta \vdash_d \alpha_i$, then $\Delta \vdash_d \beta$.

We consider two types of deductive argument. For $\Phi \subseteq$ `DefFormulae`, and a literal $\alpha \in$ `DefFormulae`, $\langle \Phi, \alpha \rangle$ is a **defeasible argument** iff $\Phi \vdash_d \alpha$ and there is no proper subset Φ' of Φ such that $\Phi' \vdash_d \alpha$. For $\Phi \subseteq$ `PropFormulae`, and a formula $\alpha \in$ `PropFormulae`, $\langle \Phi, \alpha \rangle$ is a **classical argument** iff $\Phi \vdash \alpha$, $\Phi \nvdash \bot$ and there is no proper subset Φ' of Φ such that $\Phi' \vdash \alpha$. For an argument $A = \langle \Phi, \alpha \rangle$, the function $\mathsf{Support}(A)$ returns Φ and the function $\mathsf{Claim}(A)$ returns α.

For defeasible arguments A and B, we consider the following type of **defeasible attack**: A is a **defeasible undercut** of B if (1) there is a rule $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta$ in $\mathsf{Support}(B)$ and $\mathsf{Claim}(A)$ is the complement of β (i.e. if $\mathsf{Claim}(A)$ is an atom ψ, then β is $\neg\psi$, and if β is an atom ψ, then $\mathsf{Claim}(A)$ is $\neg\psi$); Or (2) $\mathsf{Claim}(A)$ is the complement of a literal in $\mathsf{Support}(B)$. We have a wider range of options for defining attack for classical logic, such as rebuttals [2, 14], direct undercuts [15,

16, 1], and canonical undercuts [3]. For classical arguments A and B, we consider the following type of **classical attack** in this paper: A is a **classical direct undercut** of B if $\exists \phi \in \mathsf{Support}(B)$ s.t. $\mathsf{Claim}(A) \equiv \neg\phi$; A is a **classical canonical undercut** of B if $\mathsf{Claim}(A) \equiv \neg \bigwedge \mathsf{Support}(B)$; And A is a **classical rebuttal** of B if $\mathsf{Claim}(A) \equiv \neg\mathsf{Claim}(B)$. We give some examples of logical attack in the following.

$\langle\{e, e \rightarrow \neg b\}, \neg b\rangle$ is a defeasible undercut of $\langle\{c, d, c \rightarrow b, b \wedge d \rightarrow a\}, a\rangle$
$\langle\{\neg a \wedge \neg b\}, \neg a\rangle$ is a classical direct undercut of $\langle\{a, b, c\}, a \wedge b \wedge c\rangle$
$\langle\{\neg a \wedge \neg b\}, \neg(a \wedge b \wedge c)\rangle$ is a classical canonical undercut of $\langle\{a, b, c\}, a \wedge b \wedge c\rangle$
$\langle\{a, a \rightarrow b\}, b \vee c\rangle$ is a classical rebuttal of $\langle\{\neg b, \neg c\}, \neg(b \vee c)\rangle$

3 Logical argument systems

In this paper, we consider a variety of logical argument systems based on deductive arguments using either defeasible logic or classical logic.

Definition 1. *A* **logical argument system** *is a tuple* $\langle \mathtt{Kbs}, \mathtt{Arg}, \mathtt{Att}, \mathtt{Con}\rangle$*, denoted* $\mathtt{Sys}$*, where for some language of formulae* $\mathtt{Formulae}$*, we have* $\mathtt{Kbs} = \wp(\mathtt{Formulae})$*,* $\mathtt{Arguments} = \{\langle \Phi, \psi\rangle \mid \Phi \in \mathtt{Kbs}$ *and* $\psi \in \mathtt{Formulae}\}$*, and* $\mathtt{Attacks} = \mathtt{Arguments} \times \mathtt{Arguments}$ *and*

$$\begin{aligned}
&\mathtt{Arg} : \wp(\mathtt{Formulae}) \rightarrow \wp(\mathtt{Arguments})\\
&\mathtt{Att} : \wp(\mathtt{Formulae}) \rightarrow \wp(\mathtt{Attacks})\\
&\mathtt{Con} : \wp(\mathtt{Formulae}) \times \mathtt{Arguments} \rightarrow \mathtt{Graphs}
\end{aligned}$$

This is a general definition that can be instantiated by a wide variety of logical argument systems. We give examples in Section 3.1. The sets $\mathtt{Arguments}$ and $\mathtt{Attacks}$ are given as types for the functions $\mathtt{Arg}$, $\mathtt{Att}$, and $\mathtt{Con}$. We explain the parameters $\langle \mathtt{Kbs}, \mathtt{Arg}, \mathtt{Att}, \mathtt{Con}\rangle$ of the definition as follows. The $\mathtt{Kbs}$ set is the set of knowledgebases that can be used by the system. In this paper, we focus on the knowledgebases given by the languages of defeasible formulae and classical formulae. The $\mathtt{Arg}$ function gives the set of arguments that can be generated from a knowledgebase. In this paper, we focus on defeasible arguments and classical arguments. The $\mathtt{Att}$ function gives the set of attacks that can be generated from a knowledgebase, and so $(A, B) \in \mathtt{Att}(\Delta)$ means that A attacks B (e.g. defeasible undercut or classical rebuttal). The $\mathtt{Con}$ function (called the **constructor function**) that, given a knowledgebase Δ and a specified argument A, called the **focal argument**, returns a graph s.t. if $A \in \mathtt{Arg}(\Delta)$, then the construction starts with A as a node in the graph and then builds the graph using a subset of the attacks relation (i.e. a subset of $\mathtt{Att}(\Delta)$) as the edges, and if $A \notin \mathtt{Arg}(\Delta)$, then the graph is the empty graph $G_\emptyset$.

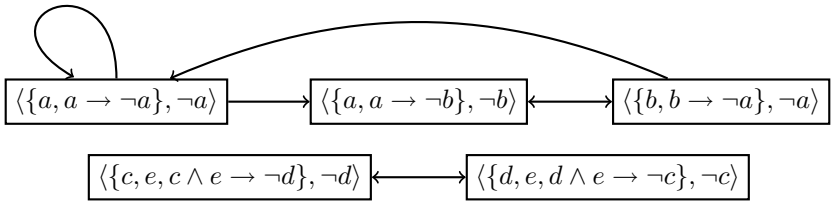
The constructor function encodes the method by which we generate an argument graph from a set of logical arguments and attacks between those arguments. In this paper, we consider the four constructor functions that we define and illustrate in the rest of this subsection.

Definition 2. *Let* $\langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *be a logical argument system.*

- $\texttt{Con}$ *is a* **trivial constructor** *iff for any knowledgebase* Δ*, and for any argument* $A \in \texttt{Arg}(\Delta)$*,* $\texttt{Con}(\Delta, A)$ *is the graph* $(\texttt{Arg}(\Delta), \texttt{Att}(\Delta))$*.*
- $\texttt{Con}$ *is a* **simple constructor** *iff for any knowledgebase* Δ*, and for any argument* $A \in \texttt{Arg}(\Delta)$*,* $\texttt{Con}(\Delta, A)$ *is the component in the graph* $(\texttt{Arg}(\Delta), \texttt{Att}(\Delta))$ *containing* A*.*

We can regard the simple constructor as starting with the focal argument A, and adding all the arguments that attack the argument (by adding the node and arc(s) for each of these counterarguments). Then it repeats this step iteratively until no more arguments can be added.

Example 1. For $\Delta = \{a, b, c, d, e, a \rightarrow \neg a, b \rightarrow \neg a, a \rightarrow \neg b, d \wedge e \rightarrow \neg c, c \wedge e \rightarrow \neg d\}$, let $\texttt{Arg}(\Delta) = \{\ A_1,\ A_2,\ A_3,\ A_4,\ A_5\ \}$, where A_1 is $\langle \{a, a \rightarrow \neg a\}, \neg a \rangle$, A_2 is $\langle \{a, a \rightarrow \neg b\}, \neg b \rangle$, A_3 is $\langle \{b, b \rightarrow \neg a\}, \neg a \rangle$, A_4 is $\langle \{c, e, c \wedge e \rightarrow \neg d\}, \neg d \rangle$, and A_5 is $\langle \{d, e, d \wedge e \rightarrow \neg c\}, \neg c \rangle$, and $\texttt{Att}(\Delta) = \{\ (A_1, A_1),\ (A_1, A_2),\ (A_2, A_3),\ (A_3, A_1),\ (A_3, A_2),\ (A_4, A_5),\ (A_5, A_4)\ \}$. For this, the trivial constructor $\texttt{Con}(\Delta, A)$ returns the following graph, where any of A_1 to A_5 is the focal argument A. Furthermore, the simple constructor $\texttt{Con}(\Delta, A)$ returns the component below containing A_1 to A_3, where any of A_1 to A_3 is the focal argument A. Likewise, if the focal argument A would be A_4 or A_5, the simple constructor would return the other component.



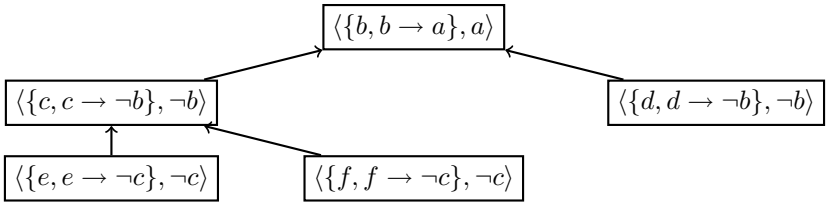
The recursive constructor (defined next) is related to proposals for constructing trees in classical logic [3] and in defeasible logic programming [7]. For the following definition, the constructor starts with the focal argument as the root, and all attackers are added as children. Then by recursion, for each argument in the graph, all the attackers of the argument are added as children. The only restriction to this is the so called "no recycle" condition, which says that when adding an attacker to the graph, it should contain at least one formula in the support that has not been used as a premise in any ancestor argument (i.e. an argument that is on the branch to the root). Consequently, the recursive constructor always yields a (directed) acyclic graph.

Definition 3. *Let* $\langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *be a logical argument system. Let* Δ *be a knowledgebase, and let* A *be an argument.* $\texttt{Con}$ *is a* **recursive constructor** *iff for any knowledgebase* Δ*, and for any argument* $A \in \texttt{Arg}(\Delta)$*,* $\texttt{Con}(\Delta, A)$ *is the directed graph* G *constructed by adding exactly the arguments as follows:*

1. *A is the root of G*

2. *if* $(B, A) \in \texttt{Att}$*, then* B *is a child of* A *in* G
3. *by recursion, for each node* C *in* G*, if* $(D, C) \in \texttt{Att}$*, and the support of* D *contains at least one premise that does not appear in the support of any argument on the branch from* C *to* A*, then* D *is a child of* C *in* G*.*

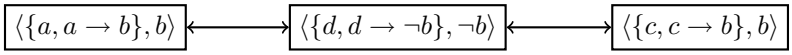
Example 2. For $\Delta = \{b, b \rightarrow a, c, c \rightarrow \neg b, d, d \rightarrow \neg b, e, e \rightarrow \neg c, f, f \rightarrow \neg c\}$, let $(\texttt{Arg}(\Delta) = \{A_1, A_2, A_3, A_4, A_5\}$, where A_1 is $\langle\{b, b \rightarrow a\}, a\rangle$, A_2 is $\langle\{c, c \rightarrow \neg b\}, \neg b\rangle$, A_3 is $\langle\{d, d \rightarrow \neg b\}, \neg b\rangle$, A_4 is $\langle\{e, e \rightarrow \neg c\}, \neg c\rangle$, A_5 is $\langle\{f, f \rightarrow \neg c\}, \neg c\rangle$, A_6 is $\langle\{b, c \rightarrow \neg b\}, \neg c\rangle$, A_7 is $\langle\{c, e \rightarrow \neg c\}, \neg e\rangle$, A_8 is $\langle\{c, f \rightarrow \neg c\}, \neg f\rangle$, and A_9 is $\langle\{b, d \rightarrow \neg b\}, \neg d\rangle$, and $\texttt{Att}(\Delta))$ = { (A_2, A_1), (A_2, A_6), (A_3, A_1), (A_3, A_9), (A_4, A_2), (A_4, A_7), (A_4, A_8), (A_5, A_2), (A_5, A_8), (A_6, A_2), (A_6, A_7), (A_6, A_8), (A_7, A_4), (A_8, A_5), (A_9, A_3) }. For this, the recursive constructor returns the following graph containing arguments $A_1 \ldots A_5$, where A_1 is the focal argument. Note that arguments $A_6 \ldots A_9$ do not appear in the graph, due to the third condition of Definition 3.



The rebuttal constructor (defined next) is similar to proposals for reasoning with pros and cons. Given the focal argument, all arguments with a logically equivalent claim or with a contradictory claim are included. The rebuttal constructor always yields a complete bipartite graph.

Definition 4. *Let* $\langle\texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con}\rangle$ *be a logical argument system. Let* Δ *be a knowledgebase, and let* A *be an argument.* Con *is a* **rebuttal constructor** *iff for any knowledgebase* Δ*, and for any argument* $A \in \texttt{Arg}(\Delta)$*,* $\texttt{Con}(\Delta, A)$ *is the graph obtained by taking the nodes to be the arguments that either have a claim that is logically equivalent to* $\mathsf{Claim}(A)$ *or a claim that is logically equivalent to* $\neg\mathsf{Claim}(A)$ *and by taking the edges to be the rebuttals between these nodes.*

Example 3. For $\Delta = \{a, a \rightarrow b, c, c \rightarrow b, d, d \rightarrow \neg b, d, d \rightarrow \neg c, c, c \rightarrow \neg d\}$, let $(\texttt{Arg}(\Delta) = \{\ A_1,\ A_2,\ A_3,\ A_4,\ A_5\ \}$, where A_1 is $\langle\{a, a \rightarrow b\}, b\rangle$, A_2 is $\langle\{c, c \rightarrow b\}, b\rangle$, A_3 is $\langle\{d, d \rightarrow \neg b\}, \neg b\rangle$, A_4 is $\langle\{d, d \rightarrow \neg c\}, \neg c\rangle$, and A_5 is $\langle\{c, c \rightarrow \neg d\}, \neg d\rangle$, and $\texttt{Att}(\Delta))$ = { (A_1, A_3), (A_2, A_3), (A_3, A_1), (A_3, A_2) }. For this, the rebuttal constructor returns the following graph, where any of A_1 to A_3 is the focal argument.



In general, we do not impose constraints on a logical argument system. In this paper, we only consider instances of Arg, Att, and Con that are monotonic. However, it would be reasonable to consider non-monotonic versions of the functions, but we leave that to future work.

3.1 Examples of logical argument systems

To illustrate the idea of logical argument systems, we present some instances, denoted System 1 to System 5, next, and then in the following section, we will consider properties of these systems.

System 1 *The tuple* $\langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *is a system based on defeasible logic where* Kbs *is* $\wp(\texttt{DefFormulae})$*,* $\texttt{Arg}(\Delta)$ *is the set of defeasible arguments from* Δ *such that if* $B \in \texttt{Arg}(\Delta)$*, then* $\textsf{Support}(B) \subseteq \Delta$*,* $\texttt{Att}(\Delta)$ *is* $\{(B, C) \mid B, C \in \texttt{Arg}(\Delta)$ *and* B *is a defeasible undercut of* $C\}$*, and* $\texttt{Con}(\Delta, A)$ *is the simple constructor.*

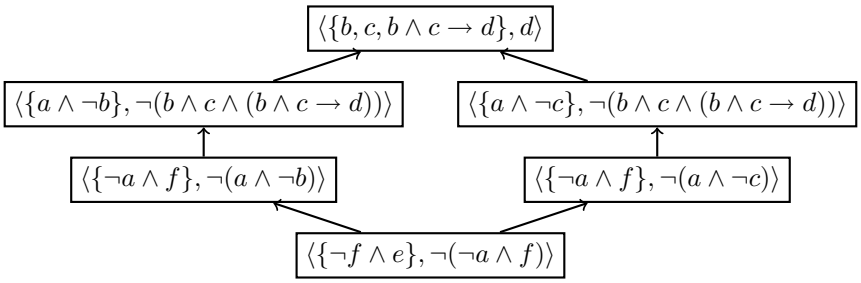
Example 4. Consider System 1 with $\Delta = \{a, b, a \rightarrow \neg a, b \rightarrow \neg a, a \rightarrow \neg b, d \rightarrow \neg c, c \rightarrow \neg d\}$. For the focal argument $\langle \{a, a \rightarrow \neg a\}, \neg a \rangle$, $\texttt{Con}(\Delta, A)$ gives the constructed graph that is the component with three arguments in Example 1.

From the directed graph obtained by the next system, it is simple to obtain the argument tree of Besnard and Hunter[3] [3].

System 2 *The tuple* $\langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *is a system based on classical logic where* Kbs *is* $\wp(\texttt{PropFormulae})$*,* $\texttt{Arg}(\Delta)$ *is the set of classical arguments such that if* $B \in \texttt{Arg}(\Delta)$*, then* $\textsf{Support}(B) \subseteq \Delta$*,* $\texttt{Att}(\Delta)$ *is* $\{(B, C) \mid B, C \in \texttt{Arg}(\Delta)$ *and* B *is a canonical undercut of* $C\}$*, and* $\texttt{Con}(\Delta, A)$ *is the recursive constructor.*

So the constructor function returns the smallest graph obtained by starting with A, adding all the canonical undercuts to A, and by recursion adding all the canonical undercuts A_n to each of the canonical undercuts A_{n-1} subject to the condition that each canonical undercut A_n has a premise that does not appear in any support on the path of arguments A_{n-1}, ..., A_1 where A_1 is A.

Example 5. Consider System 2 with $\Delta = \{b, c, a \wedge \neg b, a \wedge \neg c, \neg a \wedge f, \neg f \wedge e, b \wedge c \rightarrow d\}$. For the focal argument $\langle \{b, c, b \wedge c \rightarrow d\}, d \rangle$, $\texttt{Con}(\Delta, A)$ gives the following constructed graph which is an example of a rooted graph.



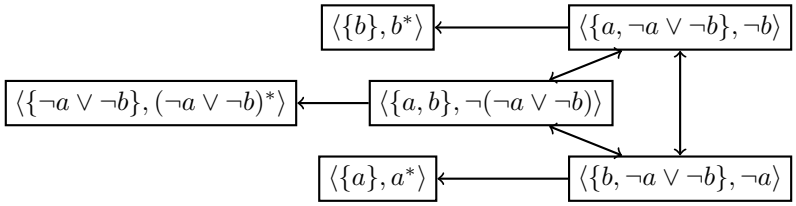
[3] A rooted graph is translated to an argument tree of Besnard and Hunter as follows: Start from the bottom of the graph working upwards. For each node with multiple parents, a copy is made of the node and its offspring for each of its parent, so that each copy has exactly one parent. For Example 5, the bottom node is copied so the argument occurs in two leaf nodes

The next system is based on the idea of exhaustively generating all arguments from a knowledgebase and all attacks (according to a particular definition of attack) and using the resulting graph without restriction (as first proposed in [1], and further explored in [5]).

System 3 *The tuple* $\langle \mathtt{Kbs}, \mathtt{Arg}, \mathtt{Att}, \mathtt{Con} \rangle$ *is a system based on classical logic where* Kbs *is* $\wp(\mathtt{PropFormulae})$, $\mathtt{Arg}(\Delta)$ *is the set of classical arguments such that if* $B \in \mathtt{Arg}(\Delta)$, *then* $\mathsf{Support}(B) \subseteq \Delta$, $\mathtt{Att}(\Delta)$ *is* $\{(B, C) \mid B, C \in \mathtt{Arg}(\Delta)$ *and* B *is a direct undercut of* $C\}$, *and* $\mathtt{Con}(\Delta, A)$ *is the simple constructor.*

Note, here we use the classical direct undercut. But, we could use the classical defeater, classical direct defeater, classical undercut, classical canonical undercut, or classical literal undercut, as an alternative (see definitions in [5]). Hence, we have a range of systems based on the choice of attack.

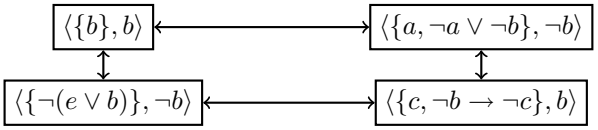
Example 6. Consider System 3 with $\Delta = \{a, \neg a \vee \neg b, b\}$. For the focal argument $A = \langle \{b\}, b \rangle$, $\mathtt{Con}(\Delta, A)$ gives the following constructed graph. For this, each argument with a claim with an asterisk, i.e. a claim of the form α^*, denotes any argument with the same premises and a claim that it is implied by α.



The following system is the same as the previous system but restricts consideration to classical rebuttals rather than direct undercuts [5]. As an alternative we could consider classical direct defeating rebuttals (see definition in [3]).

System 4 *The tuple* $\langle \mathtt{Kbs}, \mathtt{Arg}, \mathtt{Att}, \mathtt{Con} \rangle$ *is a system based on classical logic where* Kbs *is* $\wp(\mathtt{PropFormulae})$, $\mathtt{Arg}(\Delta)$ *is the set of classical arguments such that if* $A \in \mathtt{Arg}(\Delta)$, *then* $\mathsf{Support}(B) \subseteq \Delta$, $\mathtt{Att}(\Delta)$ *is* $\{(B, C) \mid B, C \in \mathtt{Arg}(\Delta)$ *and* B *is a rebuttal of* $C\}$, *and* $\mathtt{Con}(\Delta, A)$ *is the rebuttal constructor.*

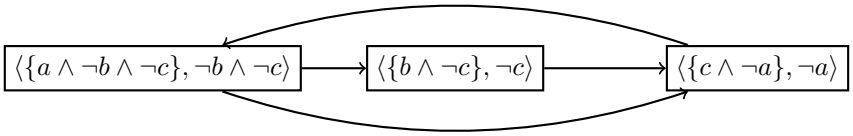
Example 7. Consider System 4 with $\Delta = \{a, \neg a \vee \neg b, b, c, \neg b \rightarrow \neg c, \neg(e \vee b)\}$. For argument $A = \langle \{b\}, b \rangle$, $\mathtt{Con}(\Delta, A)$ gives the following constructed graph.



Finally, we give an example of a new logical argument system that is very constrained with respect to the kinds of arguments that are allowed. Essentially, this system allows us to avoid the symmetrical relationships that usually hold for attack for a classical argument system.

System 5 *The tuple* $\langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *is a system based on classical logic where* $\texttt{ConForm} = \{\ \alpha \wedge \beta_1 \wedge ... \wedge \beta_n \mid \alpha$ *is a positive literal and* $\beta_1, ..., \beta_n$ *are negative literals* $\}$, $\texttt{Kbs}$ *is* $\wp(\texttt{ConForm})$, $\texttt{Arg}(\Delta)$ *is the set of tuples of the form* $\langle \{\phi\}, \psi \rangle$ *where* $\phi \in \Delta$ *and* $\{\phi\} \vdash \psi$ *and* ψ *is the conjunction of negative literals occurring in* ϕ, $\texttt{Att}(\Delta)$ *is* $\{(B, C) \mid B, C \in \texttt{Arg}(\Delta)$ *and* B *is a classical undercut of* $C\}$, *and* $\texttt{Con}(\Delta, A)$ *is the simple constructor.*

Example 8. Consider System 5 with $\Delta = \{a \wedge \neg b \wedge \neg c, b \wedge \neg c, c \wedge \neg a\}$. For the focal argument $A = \langle \{a \wedge \neg b\}, \neg b \rangle$, $\texttt{Con}(\Delta, A)$ gives the following graph.



In this section, we have presented a non-exhaustive range of logical argument systems. Most are based on well-known approaches. The last system is a new proposal for studying structural properties rather than being useful in its own right.

4 Induced graphs

The following definition captures the relationships that we will consider between a logical argument system and a class of graphs. The more general the class of graphs that a logical argument system can cover, the wider the range of argumentation situations the logical argument systems can capture.

Definition 5. *Let* $\texttt{Sys} = \langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *be a logical argument system and let* X *be a graph type.*

- $\texttt{Sys}$ **constructively covers** X *iff for all* $G \in X$, *there is a* $\Delta \in \texttt{Kbs}$, *and there is an* $A \in \texttt{Arg}(\Delta)$, *such that* $\texttt{Con}(\Delta, A) = G$.
- $\texttt{Sys}$ *is* **constructively covered** *by* X *iff for all* $\Delta \in \texttt{Kbs}$, *and for all* $A \in \texttt{Arguments}$, *if* $\texttt{Con}(\Delta, A) = G$, *then* $G \in X$.
- $\texttt{Sys}$ *is* **constructively complete** *for* X *iff* $\texttt{Sys}$ *constructively covers* X *and* $\texttt{Sys}$ *is constructively covered by* X.

Since the constructor function returns a graph, by definition, any logical argument system is constructively covered by $\texttt{Graphs}$. We now consider in more detail the systems from the previous section starting with System 1. Note, if we use the trivial constructor instead of the simple constructor, it is straightforward to show it is constructively complete for $\texttt{Graphs}$.

Proposition 1. *System 1 is constructively complete for* $\texttt{Components}$.

Turning to System 2, Example 5 illustrates that it is not constructively complete for $\texttt{Trees}$.

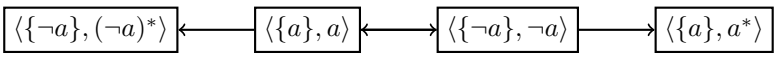
Proposition 2. *System 2 is constructively complete for* `RootedGraphs`.

System 3 does not correspond to any of the classes of graphs presented earlier. In particular, it does not constructively cover `RootedGraphs`, `AcyclicGraphs`, `RationalGraphs`, `Bipartites`, or `Components`. Furthermore, it is not covered by `RootedGraphs`, `AcyclicGraphs`, and `Bipartites`. However, we do show next that as System 3 does not allow inconsistent premises, it excludes self-cycles, and so it is covered by `RationalGraphs`.

Proposition 3. *System 3 is constructively covered by* `RationalGraphs`.

To illustrate the difficulty in identifying a tighter bound on the set of graphs that System 3 is covered by, we consider the problem of constructing a component with two arguments attacking each other. We indicate by the following example that this is not possible. Note, this is not a pathological example as there are many simple graphs that cannot be generated by System 3.

Example 9. For System 3, let $\Delta = \{a, \neg a\}$. Hence, there are two classical arguments $\langle\{a\}, a\rangle$ and $\langle\{\neg a\}, \neg a\rangle$ that are direct undercuts of each other. Plus, there are two further kinds of argument, $\langle\{a\}, a^*\rangle$ and $\langle\{\neg a\}, (\neg a)^*\rangle$, where a^* is strictly weaker than a (i.e. $\{a\} \vdash a^*$ and $\{a^*\} \nvdash a$). and $(\neg a)^*$ is strictly weaker than $\neg a$.



For System 4, the definition of the rebuttal constructor renders it straightforward to show that the system is constructively complete for `Bipartites`.

Proposition 4. *System 4 is constructively complete for* `Bipartites`.

The restrictions on the form of the arguments arising in System 5 allow us to show that even with classical logic, we can get almost the same completeness results as with defeasible logic. What are missing are the self-loop components.

Proposition 5. *System 5 is constructively complete for* `RationalGraphs`.

In this section, we have shown how restricted systems such as those based on defeasible logic (e.g. System 1), or those based on very restricted arguments (e.g. System 5) are constructively complete for rational graphs, or even components, whereas unrestricted use of classical logic means these properties do not hold.

5 Local and global constructors

To get completeness results for components, graphs, or rational graphs, the logical argument system needs to be restricted in some way. For example, the proof theory of System 1, for generating arguments is weak (it is modus ponens) and for System 5, the arguments are restricted to having a single premise and the

claim being a conjunction of negative literals. From the systems we have considered so far, we see a trade-off with regard to how restricted the system is and the completeness results that hold for it. We investigate this issue in this section by classifying constructors. For this we need the subsidiary notion of a characteristic function $\texttt{Test}$ which is a function from sets of attacks to $\{$"*yes*", "*no*"$\}$.

Definition 6. *A constructor function* $\texttt{Con}$ *is* **local** *iff there is a characteristic function* $\texttt{Test}$ *s.t. for all* Δ, A, *if* $\texttt{Con}(\Delta, A) = (N, E)$, *and* (N, E) *is a component, and* $B_i \in N$, *and* $(B_j, B_i) \in \texttt{Att}(\Delta)$, *and* $\texttt{Test}((B_j, B_i)) =$ "*yes*", *then* $(B_j, B_i) \in E$ *and* $B_j \in N$. *A constructor function* $\texttt{Con}$ *is* **global** *iff* $\texttt{Con}$ *is not local.*

A local constructor function thus constructs a component by adding nodes and arcs incrementally starting with A. The local constructor makes a local decision on whether to add a node or arc based on the nature of the attack. It does not take into account any other aspect of the graph. In other words, no global view is taken into account when constructing the graph.

Proposition 6. *The trivial constructor function, simple constructor function and the rebuttal constructor function are local, whereas the recursive constructor function is global.*

The following results show that unless a system is highly restricted, it is not possible to generate every graph with a local constructor function. In order to directly compare defeasible and classical logics, we have used a restricted version of the defeasible logic system considered earlier.

Theorem 1. *Let* $\texttt{Sys} = \langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *be a logical argument system such that* $\texttt{Kbs}$ *is the set of defeasible knowledgebases,* $\texttt{Arg}$ *is the set of non-self attacking defeasible arguments from* Δ *(i.e. for each argument* $A \in \texttt{Arg}(\Delta)$, A *does not attack* A*), and* $\texttt{Att}$ *is defeasible undercut. There is a constructor* $\texttt{Con}$ *such that* $\texttt{Con}$ *is local and* $\texttt{Sys}$ *is constructively complete for* $\texttt{RationalGraphs}$.

Theorem 2. *Let* $\texttt{Sys} = \langle \texttt{Kbs}, \texttt{Arg}, \texttt{Att}, \texttt{Con} \rangle$ *be a logical argument system such that* $\texttt{Kbs}$ *is the set of classical knowledgebases,* $\texttt{Arg}$ *is the set of classical arguments from* Δ, *and* $\texttt{Att}$ *is classical defeater, classical direct defeater, classical undercut, classical canonical undercut, or classical direct undercut. If* $\texttt{Sys}$ *is constructively complete for* $\texttt{RationalGraphs}$, *then* $\texttt{Con}$ *is global.*

The main ramification of the above result is that if we want to use richer logics such as classical logic, then we need to use global constructors. In other words, to reflect any abstract argument graph in a logical argument system based on a richer logic, we need to be selective in the choice of arguments taken from $\texttt{Arg}(\Delta)$ and the choice of attacks taken from $\texttt{Att}(\Delta)$ for any given Δ and A. Therefore, these results in a sense justify the need to better understand the notion of global constructors.

Furthermore, this is not just for theoretical interest. Practical argumentation often seems to use richer logics such as classical logic, and often the presentation of arguments and counterarguments is not exhaustive. Therefore, we

need to better understand how the arguments presented are selected. For example, suppose agent 1 posits $A_1 = \langle\{b, b \rightarrow a\}, a\rangle$, and agent 2 then posits $A_2 = \langle\{c, c \rightarrow \neg b\}, \neg b\rangle$. It would be reasonable for this dialogue to stop at this point even though there are further arguments that can be constructed from the public knowledge such as $A_3 = \langle\{b, c \rightarrow \neg b\}, \neg c\rangle$. So in terms of constructing the constellation of arguments and counterarguments from the knowledge, we need to know what the underlying principle is for ascertaining that just the two arguments are sufficient given the public knowledge, and that this means we need to know more about the global constructor function. It may also mean that we need to better understand how meta-knowledge (about the premises and/or about the participants) is used to select arguments and counterarguments.

6 Discussion

In this paper we have provided: (1) A general framework for describing diverse logical argument systems; (2) A classification scheme for logical argument systems in terms of the class of graphs that they induce; (3) An analysis of local and global methods of constructing argument graphs from a knowledgebase which has ramifications for using richer logics in argumentation.

There are further options that we may consider for logical argument systems by for instance changing the definition of attack or changing the choice of base logic: (i) defeasible logic with annotations for truth values (such as for Belnap's four-valued logic) [17] and for possibility theory [18], (ii) temporal reasoning calculi [19, 20], (iii) minimal logic [21], and (iv) modal logic [22]. Indeed, any logic could be potentially used as a base logic [8].

Whilst, the focus of the paper has been on deductive arguments, the issues raised may also have ramifications for further argumentation systems such as ASPIC+ [23] and ABA [24]. We leave investigation of this to future work.

Acknowledgments. This research has partially been supported by the Austrian Science Fund (FWF) through project I1102.

References

1. Cayrol, C.: On the relation between argumentation and non-monotonic coherence-based entailment. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95). (1995) 1443–1448
2. Pollock, J.: Defeasible reasoning. Cognitive Science **11**(4) (1987) 481–518
3. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artif. Intell. **128** (2001) 203–235
4. Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence **34** (2002) 197–215
5. Gorogiannis, N., Hunter, A.: Instantiating abstract argumentation with classical logic arguments: Postulates and properties. Artif. Intell. **175**(9-10) (2011) 1479–1497

6. Black, E., Hunter, A., Pan, J.: An argument-based approach to using multiple ontologies. In: Proceedings of the Third International Conference on Scalable Uncertainty Management (SUM'09). Volume 5785., Springer (2009) 68–79
7. García, A., Simari, G.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming **4** (2004) 95–138
8. Hunter, A.: Base logics in argumentation. In: Proceedings of the 3th Conference on Computational Models of Argument (COMMA'10). Volume 216 of Frontiers in Artificial Intelligence and Applications., IOS Press (2010) 275–286
9. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. Artif. Intell. **77** (1995) 321–357
10. Dunne, P.: Computational properties of argument systems satisfying graph-theoretic constraints. Artif. Intell. **171**(10-15) (2007) 701–729
11. Coste-Marquis, S., Devred, C., Marquis, P.: Symmetric argumentation frameworks. In Godo, L., ed.: Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005). Volume 3571 of Lecture Notes in Computer Science., Springer (2005) 317–328
12. Dvorák, W., Szeider, S., Woltran, S.: Abstract argumentation via monadic second order logic. In: Proceedings of the Sixth International Conference on Scalable Uncertainty Management (SUM'09). Volume 7520 of Lecture Notes in Computer Science., Springer (2012) 85–98
13. Dvorák, W., Pichler, R., Woltran, S.: Towards fixed-parameter tractable algorithms for abstract argumentation. Artif. Intell. **186** (2012) 1–37
14. Pollock, J.: How to reason defeasibly. Artif. Intell. **57**(1) (1992) 1–42
15. Elvang-Gøransson, M., Krause, P., Fox, J.: Acceptability of arguments as 'logical uncertainty'. In: Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'93). Volume 747 of Lecture Notes in Computer Science., Springer (1993) 85–90
16. Elvang-Gøransson, M., Hunter, A.: Argumentative logics: Reasoning with classically inconsistent information. Data & Knowledge Engineering **16**(2) (1995) 125–145
17. Takahashi, T., Sawamura, H.: A logic of multiple-valued argumentation. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), IEEE Computer Society (2004) 800–807
18. Alsinet, T., Chesñevar, C., Godo, L., Simari, G.: A logic programming framework for possibilistic argumentation: Formalization and logical properties. Fuzzy Sets and Systems **159**(10) (2008) 1208–1228
19. Augusto, J., Simari, G.: Temporal defeasible reasoning. Knowledge and Information Systems **3**(3) (2001) 287–318
20. Mann, N., Hunter, A.: Argumentation using temporal knowledge. In: Proceedings of the 2nd Conference on Computational Models of Argument (COMMA'08). Volume 172 of Frontiers in Artificial Intelligence and Applications., IOS Press (2008) 204–215
21. Krause, P., Ambler, S., Elvang-Gøransson, M., Fox, J.: A logic of argumentation for reasoning under uncertainty. Computational Intelligence **11** (1995) 113–131
22. Fox, J., Das, S.: Safe and Sound: Artificial Intelligence in Hazardous Applications. MIT Press (2000)
23. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation **1** (2010) 93–124
24. Dung, P., Kowalski, R., Toni, F.: Dialectical proof procedures for assumption-based admissible argumentation. Artificial Intelligence **170** (2006) 114–159