Argumentation Patterns

Serena Villata[1] and Guido Boella[1] and Leendert van der Torre[2]

[1] Dipartimento di Informatica, University of Turin
{villata,boella}@di.unito.it
[2] Computer Science and Communication, University of Luxembourg
leendert@vandertorre.com

Abstract. Argumentation patterns are general reusable solutions to commonly occurring problems in the design of argumentation frameworks, such as the relation between claim and data in the Toulmin scheme. We introduce a formal approach for the semantics of argumentation patterns describing relationships and interactions among arguments, without instantiating the involved abstract arguments. Argumentation patterns are a multi-labeling of a set of arguments, together with constraints on this labeling. Constraints express the relations among the labels of the arguments of the pattern when they interact with other arguments. Moreover, we define argumentation patterns using a two sorted argumentation framework where focal arguments are distinguished from auxiliary arguments, and we show how to compute their semantics by reusing a semantics introduced by Dung. We show how patterns are applied to design conjunction and disjunction of arguments, accrual, proof standards, and second-order attacks.

1 Introduction

An argumentation framework [9] is composed by a set of elements called *arguments* and a binary relation over the arguments called *attack*. A core issue in argumentation theory is the relation between abstract arguments. In modelling argumentation frameworks, this relation has been investigated following different lines [3, 1, 5, 13, 8, 6]. In this paper, we propose to reuse software engineering ideas like patterns to investigate the relation between abstract arguments.

Our context deals with situations where argumentation frameworks are not generated from a knowledge base, but where the knowledge engineer has to directly design the arguments and attacks. In many cases, for the engineer is easier to reuse parts of existing frameworks, so a methodology for representing abstractions facilitating such reuse and for defining their meaning is needed. As methodology we introduce argumentation patterns. Argumentation patterns are visual descriptions for how to solve design problems of argumentation frameworks, that can be used in many situations.

Argumentation patterns are sets of arguments related to each other in such a way that they cannot be expressed directly with the basic attack relation. For example, assume that a modeler believes that the argument "Jones is not

liable" is attacked if both "Jones has a contract" and "Jones has breached the contract" are acceptable. Then how to relate these three arguments such that this property holds? This is an instance of the conjunctive attack pattern: the former argument is attacked only if each of the latter is accepted. To express situations like this one the usual solution is to define extended argumentation frameworks, i.e., introducing a conjunctive attack relation with its own semantics. However, when more solutions must be put together it becomes problematic to unify everything into a single extended argumentation framework. This is why we propose patterns.

Our challenge to define argumentation patterns leads to the following research questions:

1. How to visualize argumentation patterns?
2. How to define argumentation patterns?
3. Which argumentation patterns can be identified in the literature?

First, the problem of finding a good visualization for argumentation patterns is not trivial. We mainly focus on the existing well-known visualizations such as boolean gates and transistors, and we provide the argumentative counterpart of such visualizations. In particular, we use the logic gate AND for visualizing the conjunctive pattern where each "input" argument needs to be acceptable to get the "output" argument unacceptable (or acceptable), and the OR gate for visualizing the disjunctive pattern, where at least one of the "input" arguments needs to be acceptable to get the "output" argument unacceptable (or acceptable). We introduce transistors to represent the second-order pattern where the collector is the attacking argument, the emitter is the attacked argument, and the base is the argument raising the second-order attack. Transistors can be composed to visualize the higher-order pattern. Transistors are used also to visualize part of the Toulmin scheme where the data is the collector, the claim is the emitter, and the warrant is the base.

Second, there are many ways to define argumentation patterns. Formal techniques are needed since the visualization may be ambiguous, and, in particular, not expressive enough to define how to combine argumentation patterns. Formal semantics is needed to define patterns and their use, and a formal syntax is needed to embed them in the overall argumentation framework. We consider two dimensions. First, the perspective of the designer, who knows the meaning of the pattern and how it behaves once inserted in a larger framework. We define an argumentation pattern as a set of arguments together with the specification of their behavior, which is not simply a set of attacks among the arguments of the pattern. We express the meaning of the pattern with a *multi-labeling* function and a set of propositional formulas called *constraints*. The multi-labeling shows the values assigned to the arguments in the pattern while the constraints express relations between these values. In particular, constraints allow to compute the labels of the arguments in the pattern, in case they are attacked by arguments outside the pattern. The multi-labeling, instead, restricts the possible labels of the arguments in the pattern, independently of attacks by arguments outside the pattern. The second dimension concerns the semantics of a framework which

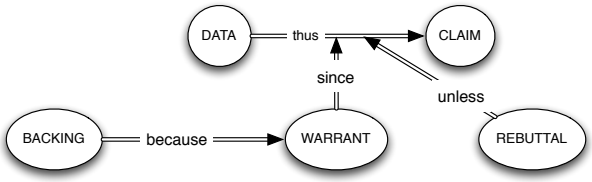


Fig. 1. The Toulmin scheme.

includes patterns. We could define an extended argumentation framework with an *ad hoc* semantics to cope with all the allowed patterns. Instead, we decide to flatten the patterns to abstract argumentation frameworks, by adding, when necessary, auxiliary arguments and suitable attacks. The flattening is driven by the definition of the pattern in terms of multi-labeling and constraints. The advantage of our solution is that it allows to reuse standard semantics, and to introduce further patterns without having to revise the semantics like in extended argumentation frameworks.

Third, the formal framework must be able to model argumentation patterns discussed in the literature. Fig. 1 visualizes the well-known Toulmin scheme [15]. The arrows represent unspecified relations between the elements, e.g., the warrant connects the data and the claim and it is supported with some backing, the rebuttal indicates circumstances in which the authority of the warrant has to be set aside. The framework has to be able to give a formal meaning to these relations – there may even be competing semantics of the Toulmin scheme, e.g., the claim is accepted only if the rebuttal is not accepted and if the warrant is supported by a backing.

Whereas most research in argumentation theory is driven by theoretical concerns, the work reported in this paper is driven by practical concerns. Even if ultimately arguments must be instantiated, in our experience of modeling argumentation [5, 3, 4], there is a need at the abstract level to define argumentation patterns. Our work raises also theoretical questions, but in this paper we restrict ourselves to concepts and examples.

This paper follows the research questions and it is organized as follows. Section 2 introduces the visualization, syntax and semantics of argumentation patterns, and how they are used. Section 3 defines patterns from the argumentation literature. Related work and conclusions end the paper.

2 Formal framework

2.1 Dung's abstract argumentation

We express Dung's [9] complete semantics of abstract argumentation using Jakobovits-Vermeir-Caminada's three valued labelings [11, 7], where an argument a can be labeled *in*, *out* or *undecided*. To define the meaning of patterns, we must be able

to express whether arguments are *in*, *out* or *undecided*, and which is the label of an argument given the label of other arguments.

We write the fact that an argument can have one of the three labels by means of propositions $a^{\in}$, $a^{\notin}$, and $a^{?}$, meaning, respectively, that argument a is *in*, *out* or *undecided*. Given this notation we can express the relation between labelings and extensions in the following way. A labeling corresponds to the extension $\{a \mid a^{\in}\}$, and given an extension $E \subseteq A$ of argumentation framework $\langle A, \rightarrow \rangle$, the corresponding labeling is given by $a^{\in}$ iff $a \in E$, $a^{\notin}$ iff $a \notin E$ and $\exists b \in E$ such that $b \rightarrow a$, and $a^{?}$ otherwise. A simple example to start with is the equivalence between two arguments which can be expressed as $a^{\in} \equiv b^{\in} \wedge a^{\notin} \equiv b^{\notin}$. We write $\Rightarrow$ for material implication.

Definition 1 (Complete semantics). *Let U be a set of arguments called the universe of arguments. For any finite set of arguments $A \subseteq U$, a three valued labeling function $l : A \rightarrow \{\in, \notin, ?\}$ is a complete function that partitions a set of arguments into the* in *($\in$),* out *($\notin$) and* undecided *(?) arguments. An acceptance function ϵ is a function that associates with every argumentation framework $\langle A, \rightarrow \rangle$ with $A \subseteq U$ and $\rightarrow \subseteq A \times A$, the set of three valued labelings of A satisfying the following conditions:*

- $\forall a, b \in A : a \rightarrow b \Rightarrow \neg(a^{\in} \wedge b^{\in})$*: an extension is conflict free.*
- $\forall b \in A : b^{\in} \Leftrightarrow \forall a : a \rightarrow b \Rightarrow a^{\notin}$*: an argument is in, if and only if all its attackers are out.*
- $\forall b \in A : b^{\notin} \Leftrightarrow \exists a : a \rightarrow b \wedge a^{\in}$*: an argument is out, if and only if at least one of its attackers is in.*

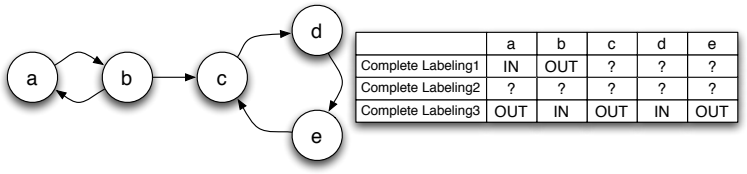
We call these labelings the complete labelings of argumentation framework $\langle A, \rightarrow \rangle$.

The following example due to Caminada [7] illustrates the complete semantics and our notation.

Example 1 (Two cycles). Fig. 2 visualizes the argumentation framework $\langle A, \rightarrow \rangle$ with $A = \{a, b, c, d, e\}$ and $\rightarrow = \{a \rightarrow b, b \rightarrow a, b \rightarrow c, c \rightarrow d, d \rightarrow e, e \rightarrow c\}$,where a="Jones is a spy", b="Jones is not a spy" c="Mary says that Jones lies", d="Jones says that Harry lies," and e="Harry says that Mary lies." This figure must be read as follows: a circle visualizes an argument, and an arrow visualizes an attack. The complete semantics is given by three labelings: $a^{\in} \wedge b^{\notin} \wedge c^{?} \wedge d^{?} \wedge e^{?}$, $a^{?} \wedge b^{?} \wedge c^{?} \wedge d^{?} \wedge e^{?}$, $a^{\notin} \wedge b^{\in} \wedge c^{\notin} \wedge d^{\in} \wedge e^{\notin}$. Other semantics return other labelings, for example the grounded semantics returns only $a^{?} \wedge b^{?} \wedge c^{?} \wedge d^{?} \wedge e^{?}$, the maximal number of undecided arguments, whereas the preferred or stable semantics only return $a^{\notin} \wedge b^{\in} \wedge c^{\notin} \wedge d^{\in} \wedge e^{\notin}$, the minimal number of undecided arguments.

2.2 Semantics of argumentation patterns

An argumentation pattern is a multi-labeling (i.e., a set of labels for each argument) of a set of arguments, together with propositional constraints on the labeling. Roughly, the multi-labeling contains the labelings of the arguments



	a	b	c	d	e
Complete Labeling1	IN	OUT	?	?	?
Complete Labeling2	?	?	?	?	?
Complete Labeling3	OUT	IN	OUT	IN	OUT

Fig. 2. Two cycles (Example 1)

when none of the arguments of the pattern is attacked by arguments not in the pattern, and the constraints represent an invariant expressing the properties which always hold between the labels of the arguments of the pattern, regardless whether the arguments are attacked by other arguments or not. The constraints are expressed in terms of propositions $x^{\in}$, $x^{\notin}$, and $x^{?}$ for all $x \in A$ which represent if an argument is *in*, *out* or *undecided*. Note that this is a possible choice, but other choices are possible too, as we discuss in the conclusion. One criterion to decide is the expressive power of the pattern language.

Definition 2 (Argumentation pattern). *An n-ary argumentation pattern is a triple $\langle A, M, C\rangle$ where $A \subseteq U$ is a sequence of n arguments, $M : A \to 2^{\{\in,\notin,?\}}$ a function from the arguments to the powerset of the labels (called a multi-labeling) and C is a propositional formula on signature $x^{\in}$, $x^{\notin}$, and $x^{?}$ for all $x \in A$. The labelings of an argumentation pattern are the labelings where the label of each argument is one of its multi-labels, and that satisfy the constraints of the pattern.*

At first sight it may seem that the multi-labeling is a constraint too, namely the constraint that the label of the argument contains one of the values of the multi-label. However, the multi-label expresses the values the arguments can have when the pattern is not attacked by other arguments, and the constraints express the relations between the values of the arguments, whether they are attacked by other arguments or not. Both the multi-labeling and the constraints are needed for the case in which patterns are used in a larger argumentation framework, and when they are combined with other argumentation patterns, as explained in Examples 7 and 8.

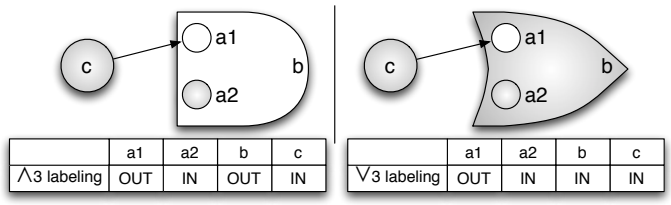
Example 2 illustrates the definition of semantics of argumentation patterns by maybe the simplest patterns, namely conjunction and disjunction. We express the patterns as *patternName: constraints*, e.g., $\wedge_{n+1}(a_1, \ldots, a_n, b)$ is the name of the conjunction pattern.

Example 2 (Conjunction and disjunction). Both patterns are defined by multi-labeling $M(a_1) = \ldots = M(a_n) = M(b) = \{\in\}$, together with respectively:

$$\wedge_{n+1}(a_1, \ldots, a_n, b) : (a_1^{\in} \wedge \ldots \wedge a_n^{\in} \Leftarrow b^{\in}) \wedge (a_1^{\notin} \vee \ldots \vee a_n^{\notin} \Rightarrow b^{\notin})$$

$$\vee_{n+1}(a_1, \ldots, a_n, b) : (a_1^{\in} \vee \ldots \vee a_n^{\in} \Leftarrow b^{\in}) \wedge (a_1^{\notin} \wedge \ldots \wedge a_n^{\notin} \Rightarrow b^{\notin})$$

Fig. 3 visualizes a_1="Jones has a contract", a_2="Jones has breached the contract" and b="Jones is liable" with $\wedge_3$ and $\vee_3$, together with an additional



	a1	a2	b	c
∧3 labeling	OUT	IN	OUT	IN

	a1	a2	b	c
∨3 labeling	OUT	IN	IN	IN

Fig. 3. Conjunction and disjunction (Example 2)

argument c="Jones did not sign the contract" attacking a_1. In the figures, we visualize in grey accepted arguments. In Fig. 3, the whole component is called b, and the two incoming ports are called a_1 and a_2. Attacking a_1 is attacking the port of the component. In the former case we have that Jones is not liable, because $a_1^{\notin} \wedge a_2^{\in} \wedge b^{\notin} \wedge c^{\in}$ is the unique labeling, in the latter case we have that Jones is liable, because $a_1^{\notin} \wedge a_2^{\in} \wedge b^{\in} \wedge c^{\in}$ is the unique labeling. Notice that the labels of the pattern are different from the multi-labeling defined above. The multi-labeling assigns to each argument the label $\in$, but the existence of argument c attacking the argument of the pattern a_1, leads to a change in the labels. Given the presence of this external argument c we cannot assign the label defined by the multi-labeling to the arguments of the pattern, thus we have to satisfy the constraints posed by the patterns.

We have to underline that a multi-label is assigned to an argument if there is the presence of a cycle in the pattern, otherwise, as in the example above, only a single label is assigned. Example 3 illustrates that some extended argumentation frameworks can also be represented by argumentation patterns, by defining second-order attacks as an argumentation pattern. Roughly, second-order attacks "disconnect" the attack relations among the arguments.

Example 3 (Second-order attack). The pattern is given by the multi-labeling $M(a) = M(b) = M(c) = \{\in\}$, because the attack relation among a and c is attacked by the second order attack, together with the constraint

$$\#_3(a, b, c) : (a^{\notin} \vee b^{\in} \Leftarrow c^{\in}) \wedge (a^{\in} \wedge b^{\notin} \Rightarrow c^{\notin})$$

Fig. 4 visualizes the second-order attack as a transistor where the collector is the attacking argument, the emitter is the attacked argument, and the base is the argument raising the second-order attack. The arguments can be read as a="Jones was honored at a special ceremony", b="Intelligence wants to study Jones's behaviour" and c="Jones is a spy".

We now have to consider the behavior of the pattern when it belongs to a wider argumentation framework. For example, consider what happens when argument d attacks the different arguments composing the pattern. We have the following possible situations $a^{\in} \wedge b^{\in} \wedge c^{\in} \wedge d^{\in}$, and respectively $d \rightarrow a$, $a^{\notin} \wedge b^{\in} \wedge c^{\in} \wedge d^{\in}$, $d \rightarrow b$, $a^{\in} \wedge b^{\notin} \wedge c^{\notin} \wedge d^{\in}$, $d \rightarrow a \wedge d \rightarrow b$, $a^{\notin} \wedge b^{\notin} \wedge c^{\in} \wedge d^{\in}$. So we have $c^{\notin}$ only if $a^{\in}$ and thus d does not attack a, together with $b^{\notin}$ and thus d attacks b.

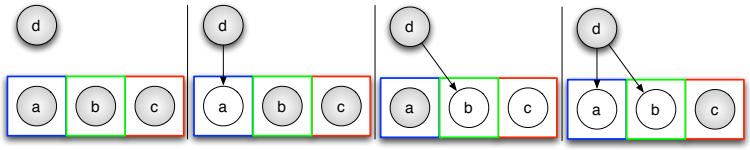


Fig. 4. Second-order attack (Example 3)

2.3 Syntax of argumentation patterns

In the previous section we described how to express the meaning of an argumentation pattern for the designer. It remains to define the structure of the patterns when they appear in an argumentation framework. Rather than proposing an extended argumentation framework with an *ad hoc* semantics to cope with all the allowed patterns, we decide to flatten the argumentation patterns to abstract argumentation frameworks, by adding auxiliary arguments and suitable attacks. In this paper, we are interested in argumentation patterns that can be expressed as a two sorted argumentation framework, distinguishing between auxiliary and focal arguments.

Definition 3 (Two sorted AF). *A two sorted argumentation framework is a triple $\langle A, B, \rightarrow\rangle$ with $A \subseteq B \subseteq U$ and $\rightarrow \subseteq B \times B$, where A are called the focal arguments, and $B \backslash A$ the auxiliary arguments.*

We have to consider two directions. First, an argumentation pattern can be flattened into a two sorted AF by respecting the multi-labeling and constraints. Second, a two sorted argumentation framework induces an argumentation pattern. This direction is more complicated, since we have to abstract away the auxiliary arguments. Moreover, given the constraints on the two sorted AF corresponding to the attack relations, we have to abstract away the propositions concerning the labeling of auxiliary arguments. This abstraction process means that we have to forget the variables referring to arguments which we abstract away, in the technical sense of forgetting defined by Lang and Marquis [12]. Generally, it is sometimes the case that ignoring a small set of propositional atoms of the formulas from an inconsistent set renders it consistent. Lang and Marquis [12] define a framework for reasoning from inconsistent propositional bases, using forgetting as a basic operation for weakening formulas. Belief bases are viewed as finite vectors of propositional formulas, conjunctively interpreted. Forgetting a set X of atoms in a formula consists in replacing it by its logically strongest consequence which is independent of X, in the sense that it is equivalent to a formula in which no atom from X occurs. The key notion is that of recoveries, which are sets of atoms whose forgetting enables restoring consistency. Forgetting is defined by Lang and Marquis [12] as follows. For more details about forgetting, see [12].

Definition 4 (Forgetting [12]). *Let ϕ be a formula from $PROP_{PS}$ and $V \subseteq PS$. The forgetting of V in ϕ, noted $\exists V.\phi$, is a formula from $PROP_{PS}$ that is inductively defined up to logical equivalence as follows:*

- $\exists \emptyset.\phi \equiv \phi$;
- $\exists \{x\}.\phi \equiv \phi_{x \leftarrow 0} \vee \phi_{x \leftarrow 1}$;
- $\exists (\{x\} \cup V).\phi \equiv \exists V.(\exists \{x\}.\phi)$;

where $PROP_{PS}$ denotes the propositional language built up from a finite set PS of atoms, the Boolean constants $\top$ and $\bot$, and the standard connectives and $\phi_{x \leftarrow 0}$ (resp. $\phi_{x \leftarrow 1}$) denotes the formula obtained by replacing in ϕ every occurrence of symbol x by $\bot$ (resp. $\top$).

Definition 5. *The argumentation pattern $\langle A, M, C \rangle$ induced by the two sorted argumentation framework $\langle A, B, \rightarrow \rangle$ is given by the constraint that takes the conjunction of the constraints given in Definition 1 for the auxiliary arguments, i.e.:*

$$\forall b \in B \setminus A : b^{\in} \Leftrightarrow \forall a : a \rightarrow b \Rightarrow a^{\notin}$$

$$\forall b \in B \setminus A : b^{\notin} \Leftrightarrow \exists a : a \rightarrow b \wedge a^{\in}$$

and then forgetting the variables referring to the arguments from which we have abstracted away.

The following two examples illustrate how the argumentation patterns for conjunction, disjunction and second-order attacks are induced by the two sorted argumentation framework.

Example 4 (Conjunction and disjunction).
AND^{n+1}=$\langle A, B, \rightarrow \rangle$ with

$$A = \{a_1, \ldots, a_n, b\}, B = A \cup \{x_1, \ldots, x_n\}$$

$$a_1 \rightarrow x_1, \ldots, a_n \rightarrow x_n, x_1 \rightarrow b, \ldots, x_n \rightarrow b$$

OR^{n+1}=$\langle A, B, \rightarrow \rangle$ with

$$A = \{a_1, \ldots, a_n, b\}, B = A \cup \{x\}$$

$$a_1 \rightarrow x, \ldots, a_n \rightarrow x, x \rightarrow b$$

The patterns AND^{n+1} and OR^{n+1} are visualized in Fig. 5. It can be verified that AND^{n+1} induces $\wedge_{n+1}(a_1, \ldots, a_n, b)$ and that OR^{n+1} induces $\vee_{n+1}(a_1, \ldots, a_n, b)$ by:

1. The multi-labeling of the pattern is that a_1 to a_n are not attacked so they are in, and therefore the auxiliary arguments are out, and therefore b is in;

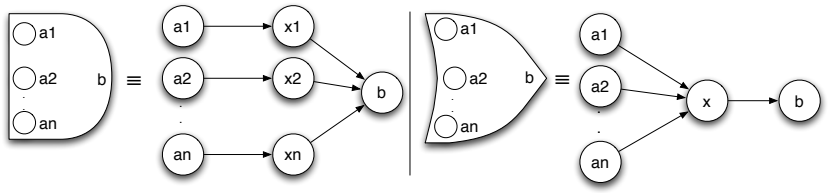


Fig. 5. Conjunction and disjunction (Example 4).

2. The constraint of conjunction is that if either one of the a_i is out, then x_i is in. The reason is that x_i is an auxiliary argument, and therefore, if it is not attacked by a_i, it is not attacked at all. If x_i is in, then b is out. vice versa, if b is in, then the x_i are out, and thus the a_i are in. The constraint of disjunction is that if all of the a_i are out, then x is in. If x is in, then b is out. Vice versa, if b is in, then x is out, and thus one of the a_i is in. By trying out all possibilities, it can be checked that these are the only constraints that hold.

Example 5 (Second-order attack).
ATTACK^3=$\langle A, B, \rightarrow\rangle$ with

$$A = \{a, b, c\}, B = A \cup \{x, y\}$$

$$a \rightarrow x, x \rightarrow y, y \rightarrow c, b \rightarrow y$$

The pattern ATTACK^3 is visualized in Fig. 6. It can be verified that ATTACK^3 induces $\#_3(a, b, c)$ by

1. The multi-labeling of the pattern is that a and b are not attacked so they are in, and therefore the auxiliary arguments are out, and therefore c is in;
2. The constraint of second-order attacks is that if a is in, then x is out. Moreover, if b and x are out, then y is in. If y is in, then c is out. The converse can be checked in the same way. By trying out all possibilities, it can be checked that this is the only constraint that holds.

Example 6 (Equivalence).
EQUIV^2=$\langle A, B, \rightarrow\rangle$ with

$$A = \{a, b\}, B = A \cup \{z_1, z_2\}$$

$$a \rightarrow z_1, z_1 \rightarrow b, b \rightarrow z_2, z_2 \rightarrow a$$

. It can be verified that EQUIV^2 induces $\equiv_2 (a, b) : a^{\in} \equiv b^{\in} \wedge a^{\notin} \equiv b^{\notin}$.

Examples 7 and 8 illustrate the difference between the multi-label and the constraints.

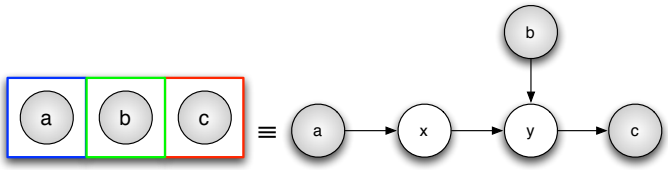


Fig. 6. Second-order attack pattern (Example 5).

Example 7. Consider the argumentation pattern $\langle A, M, C\rangle$, visualized in Fig. 7.a, where $A = \{b, d\}$. The pattern is given by multi-labeling $M(b) = M(d) = \{\in, \notin, ?\}$ together with an empty set of constraints. The two sorted argumentation framework is $\langle A, B, \rightarrow\rangle$ with

$$A = \{b, d\}, B = A \cup \{a, c\}$$

$$a \rightarrow b, b \rightarrow a, c \rightarrow d, d \rightarrow c$$

Consider now another pattern, represented in Fig. 7.b, where $A = \{b, d\}$. The pattern is given by multi-labeling $M(b) = M(d) = \{\in, \notin, ?\}$ together with the following constraint:

$$(d^{\in} \Leftarrow b^{\notin}) \wedge (b^{\in} \Leftarrow d^{\notin})$$

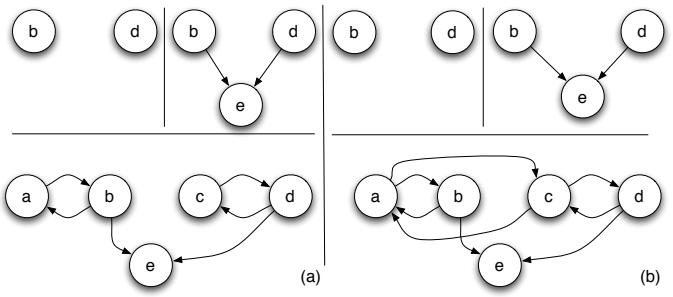
Consider now the introduction of argument e which is attacked by the two arguments b, d of the pattern. In the first case, argument e can have any label $\{\in, \notin, ?\}$ while in the second case, it cannot be $\in$, since b and d cannot both be $\notin$, as given by the constraint of the pattern. Fig. 7 shows in the tables the labelings allowed for each pattern. The two patterns have the same set of arguments and the same multi-labeling but distinct constraints. Notice that only a subset of the labelings satisfying the constraints of the first pattern satisfies the constraints of the second pattern.

Example 8. Consider the two two-sorted AF:

1. a single focal argument a, no attacks,
2. a single focal argument a and an auxiliary argument b which attack each other.

Moreover, consider the use of this pattern. The first should say that a is *in*, the second that a is either *in*, *out* or *undecided*. The constraints induced by the two multi-sorted AFs are the same (empty constraint), but the difference is represented by the multi-label.

In the context of flattening, Gabbay [10] discusses the notion of *critical subsets*. Given two argumentation frameworks where the set of arguments S_1 of the first AF is a subset of the set S_2 of the second AF, Gabbay [10] claims that S_2 is a *critical subset* of S_1 if and only if every Caminada labeling on S_2 can be extended uniquely to a labeling on S_1. This means that the additional arguments



	a	b	c	d	e
labeling1	IN	OUT	IN	OUT	IN
labeling2	IN	OUT	OUT	IN	OUT
labeling3	OUT	IN	OUT	IN	OUT
labeling4	OUT	IN	IN	OUT	OUT
labeling5	?	?	?	?	?
labeling6	?	?	IN	OUT	?
labeling7	?	?	OUT	IN	OUT
labeling8	IN	OUT	?	?	?
labeling9	OUT	IN	?	?	OUT

	a	b	c	d	e
labeling1	IN	OUT	OUT	IN	OUT
labeling2	OUT	IN	IN	OUT	OUT
labeling3	?	?	?	?	?

Fig. 7. Two patterns with the same multi-labeling and different constraints.

of S_1 only help in clarifying what is going on in S_2 and do not add any additional information. Critical subsets may recall the notion of actual arguments, whose labels are assigned, and depending on them, the labels of the auxiliary arguments are assessed.

2.4 Combining patterns

Patterns can be combined, just like boolean operators. For example, we can combine $\wedge^2$ to $\wedge^3$ and $\wedge^4$. Since attack works as a negation, we can form all kind of propositional combinations. For example, we can combine conjunction and attack to a combined conjunctive attack, known as accrual.

Example 9 (Accrual). Consider the following accrual attack pattern:

$$\#_{n+1}(a_1, \ldots, a_n, b) : (a_1^{\notin} \vee \ldots \vee a_1^{\notin} \Leftarrow b^{\in}) \wedge (a_1^{\in} \wedge \ldots \wedge a_1^{\in} \Rightarrow b^{\notin})$$

This is $\wedge_{n+1}(a_1, \ldots, a_n, c)$ extended with an attack $b \rightarrow c$. Here, the latter attack acts as a kind of negation.

When combining two patterns, we can identify some of their arguments, and then abstract these arguments away. The definition of patterns' combination is left for further research.

3 Patterns

In this section, we present how to define the argumentation patterns of well-known extended argumentation frameworks and structures.

3.1 The Toulmin scheme

Dung's argumentation framework introduces a unique binary relation among arguments, called attack relation. The notion of support is rather controversial in argumentation theory. Here, without taking a position in the debate about the representation of this notion, we present an argumentation pattern for modeling support which we adopt in the Toulmin pattern. Our support pattern idea is driven by structured argumentation where argument a supports argument b if a attacks those arguments contradicting b's conclusion, i.e., undercutting b. We move this case to abstract argumentation and the two-sorted argumentation framework in Figure 8 models support with the auxiliary argument $\neg b$ with the meaning that $a^{\notin}$ implies $b^{\notin}$. This interpretation of support in abstract argumentation has been proposed by Cayrol and Lagasquie-Schiex [8] and we represent it by means of patterns.

Example 10 (Support). The support pattern is defined by multi-labeling $M(c_1) = \ldots = M(c_n) = \{\in\}$ and $M(a) = M(b) = \{\notin\}$, together with:

$$\triangleright_{n+2}(a, b, c_1, \ldots, c_n) : ((c_1^{\in} \wedge \ldots \wedge c_n^{\in}) \Rightarrow b^{\notin} \wedge a^{\notin}) \wedge$$

$$((a^{\in} \wedge (c_1^{\notin} \wedge \ldots \wedge c_n^{\notin})) \Leftarrow b^{\in})$$

Consider $\triangleright_4$ a="Jones was born in England", b="Jones is a British citizen", c_1="Jones does not have a British passport" and c_2="Jones has a dutch accent". Now consider the pattern together with argument d. We have the following situation: d = "Mary says she saw Jones' British passport and he has no dutch accent", so $d \rightarrow c_1 \wedge c_2$, leading to the labeling $d^{\in} \wedge c_1^{\notin} \wedge c_2^{\notin} \wedge b^{\in} \wedge a^{\in}$. If d = "Jones' birth certificate is Bermudian", so $d \rightarrow a$, with the labeling $d^{\in} \wedge c_1^{\in} \wedge c_2^{\in} \wedge b^{\notin} \wedge a^{\notin}$. SUPPORT^{n+2}=$\langle A, B, \rightarrow \rangle$ with

$$A = \{a, c_1, \ldots, c_n\}, B = A \cup \{b, \neg b\}$$

$$a \rightarrow \neg b, \neg b \rightarrow b, c_1 \rightarrow b, \ldots, c_n \rightarrow b, c_1 \rightarrow a, \ldots, c_n \rightarrow a$$

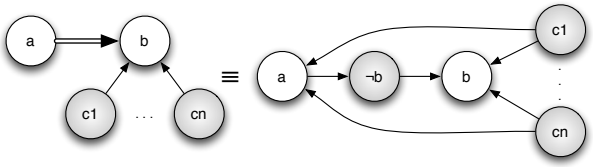


Fig. 8. The support pattern (Example 10).

Notice that the support pattern includes all attackers c_i of b. This means that we embed them in the pattern and argument b cannot be attacked by any argument external to the pattern. Thus b is an auxiliary argument which cannot

be attacked, but it still can attack other arguments. Argument b is an "output" node of the pattern. Another approach to support has been introduced by Boella et al. [4], but in this case a deductive model of support is provided where the label *out* of argument a does not imply the same label for argument b.

The Toulmin scheme, in Fig. 1, is one of the most famous patterns in argumentation theory. There is not a unique model for representing the Toulmin scheme, there are many versions in which the warrant and the rebuttal support and attack different elements of the scheme. We provide a possible pattern but many other patterns are suitable for this scheme. Consider the following well-known example about the British citizenship.

Example 11 (Toulmin scheme). Jones tries to convince Mary that he is a British citizen. The claim is "I am a British citizen". Then Jones can support his claim with the data "I was born in Bermuda". In order to move from the data to the claim, Jones has to supply a warrant to bridge the gap between them with the rule "A man born in Bermuda will legally be a British citizen". If Mary does not deem the warrant as credible, Jones should supply the legal provisions as backing statement to show that it is true that the rule holds. Finally, the rebuttal of Mary is exemplified as follows "A man born in Bermuda will legally be a British citizen, unless he has betrayed Britain and has become a spy of another country."

In Example 11, the warrant, which can be modeled as a strict rule in structured argumentation, connects the data and the claim and it is supported by the backing. Moreover, the warrant is attacked by the rebuttal. We model the rules, i.e., the warrant, in the Toulmin pattern following the example of Wyner et al. [16] for the strict rule where $z \rightarrow c$. Moreover we have to model the support given by the backing to the warrant and finally, the attack from the rebuttal to the warrant and the claim. Note that the Toulmin scheme is the combination of patterns we defined thus far, as shown in Fig. 9. It combines a transistor where the collector is the data, the emitter is the claim, and the base is the warrant, a support pattern, and a conjunctive pattern.

Example 12 (Continued). The Toulmin pattern is defined by multi-labeling $M(d) = \{\in\}$, $M(r) = \{\in\}$ and $M(b) = M(w) = M(c) = \{\notin\}$, together with:

$$TS(d, c, w, b, r) : (r^{\notin} \wedge (w^{\in} \wedge b^{\in}) \Leftarrow c^{\in}) \wedge (d^{\notin} \Rightarrow w^{\notin} \wedge c^{\notin})$$

The pattern is visualized in Fig. 9. Now consider the pattern together with argument e. We have the following situation: e = "Mary lies asserting that Jones is a spy", so $e \rightarrow r$, the labeling is $e^{\in} \wedge r^{\notin} \wedge b^{\in} \wedge d^{\in} \wedge w^{\in} \wedge c^{\in}$. The labeling satisfies the invariant expressed by the constraints. TS=$\langle A, B, \rightarrow \rangle$ with

$$A = \{d, c, w, b, r\}, B = A \cup \{\neg z, z, \neg c, \neg w, x_1, x_2, y_1, y_2\}$$

$$z \rightarrow \neg z, \neg z \rightarrow z, \neg z \rightarrow w, w \rightarrow \neg c, d \rightarrow \neg z, \neg c \rightarrow c, c \rightarrow \neg c,$$

$$r \rightarrow x_1, r \rightarrow x_2, x_1 \rightarrow y_1, x_2 \rightarrow y_2, y_1 \rightarrow c, y_2 \rightarrow w, y_2 \rightarrow b, b \rightarrow \neg w, \neg w \rightarrow w$$

Notice that the relation between b and w is a support relation as modeled above where the attacker c_i is identified by auxiliary argument y_2.

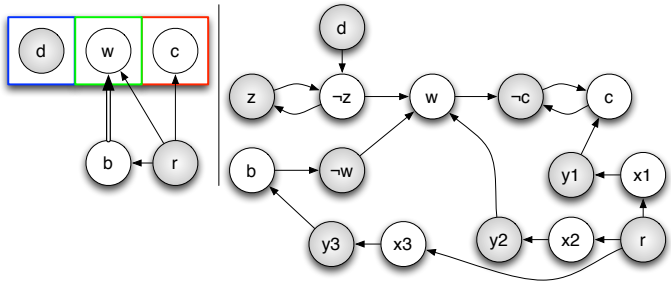


Fig. 9. The Toulmin pattern (Example 12).

3.2 Patterns for higher-order attacks

In Section 2, we introduced the pattern for second-order attacks where we follow the model of Boella et al. [3] for the multi-sorted argumentation framework. However, Modgil and Bench-Capon [13] and Baroni et al. [1] propose another way to model second-order attacks. Using patterns, we can shows that the three models are equivalent from the point of view of multi-labeling and constraints while they differ for the two sorted argumentation frameworks which induce the pattern.

Example 13 (Second-order patterns). The two patterns of Modgil and Bench-Capon [13] and Baroni et al. [1] are given by the same multi-labeling of Example 3 $M(a) = M(b) = M(c) = \{\in\}$, together with the same constraints:

$$\#_3(a, b, c) : (a^{\notin} \vee b^{\in} \Leftarrow c^{\in}) \wedge (a^{\in} \wedge b^{\notin} \Rightarrow c^{\notin})$$

Fig. 4 visualizes the multi-sorted AFs proposed by [13], ATTACK$^3{}_1$, and [1], ATTACK$^3{}_2$, which are formalized as follows: ATTACK$^3{}_1$=$\langle A, B, \rightarrow\rangle$ with

$$A = \{a, b, c\}, B = A \cup \{r\text{-}c, r\text{-}a, a\text{-def-}c, b\text{-def-}(a\text{-def-}c), r\text{-}b\}$$

$$a \rightarrow r\text{-}a, r\text{-}a \rightarrow a\text{-def-}c, a\text{-def-}c \rightarrow c, c \rightarrow r\text{-}c, b \rightarrow r\text{-}b,$$

$$r\text{-}b \rightarrow b\text{-def-}(a\text{-def-}c), b\text{-def-}(a\text{-def-}c) \rightarrow a\text{-def-}c$$

ATTACK$^3{}_2$=$\langle A, B, \rightarrow\rangle$ with $A = \{a, b, c\}, B = A \cup \{\alpha, \beta\}$

$$\alpha \rightarrow \beta, \beta \rightarrow c$$

Now consider the pattern, where arguments have the same meaning as in Example 3, together with argument $d =$ "In the Intelligence's documents there is nothing about controlling Jones", such that $d \rightarrow b$, as visualized in Fig. 10. We have the following situation for the first pattern [13]: $a^{\in} \wedge d^{\in} \wedge b^{\notin} \wedge c^{\notin}$, and the same holds for the second pattern [1]. Notice that the two patterns are the same pattern as the one of Example 3, and only the two-sorted argumentation framework which induces the pattern differs. This means that they differ only in the choice of the auxiliary arguments and the constraints which hold for these auxiliary arguments.

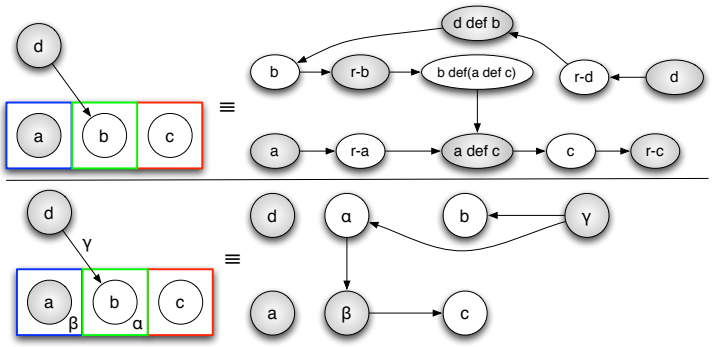


Fig. 10. Second-order attack pattern (Example 13).

3.3 Patterns for Proof Standards

In everyday reasoning and in legal reasoning, proof standards play a relevant role in those situations in which, involving risk, we apply higher standards rather than in cases where there is not much to loose. Two standards of proof have been recently analyzed by Brewka and Woltran [6] using the acceptability conditions of the abstract dialectical frameworks. The proof standards we consider are: (i) argument s is labeled $\in$ if the set of arguments R contains no node attacking s and at least one node supporting s and, (ii) s is labeled $\in$ if R contains all nodes supporting s and no node attacking s.

Example 14 (Proof standards). The patterns for proof standards are given by the same multi-labeling $M(t_1) = \ldots = M(t_n) = M(s) = \{\in\}$ and $M(r_1) = \ldots = M(r_m) = \{\notin\}$, together with different constraints:

$$PS1_{n+m+1}(t_1, \ldots, t_n, r_1, \ldots, r_m, s) :$$

$$(t_i^{\in} \wedge (r_1^{\notin} \wedge \ldots \wedge r_m^{\notin}) \Leftarrow s^{\in}) \wedge ((t_1^{\notin} \wedge \ldots \wedge t_n^{\notin}) \vee r_i^{\in} \Rightarrow s^{\notin})$$

$$PS2_{n+m+1}(t_1, \ldots, t_n, r_1, \ldots, r_m, s) :$$

$$((t_1^{\in} \wedge \ldots \wedge t_n^{\in}) \wedge (r_1^{\notin} \wedge \ldots \wedge r_m^{\notin}) \Leftarrow s^{\in}) \wedge (t_i^{\notin} \vee r_i^{\in} \Rightarrow s^{\notin})$$

Fig. 11 visualizes the two-sorted AFs which induce these patterns.
PS1^{n+m+1}=$\langle A, B, \rightarrow\rangle$ with

$$A = \{t_1, \ldots, t_n, r_1, \ldots, r_m\}, B = A \cup \{s, \neg s, \neg r_1, \ldots, \neg r_m\}$$

$$t_1 \rightarrow \neg s, \ldots, t_n \rightarrow \neg s, \neg s \rightarrow s,$$

$$r_1 \rightarrow s, \ldots, r_m \rightarrow s, \neg r_1 \rightarrow r_1, \ldots, \neg r_m \rightarrow r_m$$

PS2^{n+m+1}=$\langle A, B, \rightarrow\rangle$ with

$$A = \{t_1, \ldots, t_n, r_1, \ldots, r_m\}, B = A \cup \{s, x_1, \ldots, x_n\}$$

$$t_1 \rightarrow x_1, \ldots, t_n \rightarrow x_n, x_1 \rightarrow s, \ldots, x_n \rightarrow s, r_1 \rightarrow s, \ldots, r_m \rightarrow s$$

$$\neg r_1 \rightarrow r_1, \ldots, \neg r_m \rightarrow r_m$$

Notice that, in the two sorted argumentation framework, we avoid to have argument s attacked by other arguments external to the pattern because we consider every argument r_i attacking s.

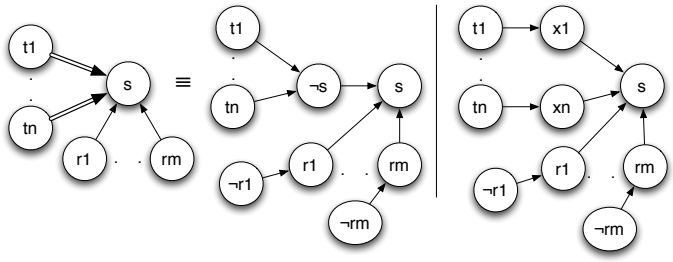


Fig. 11. Proof standards (Example 14).

4 Related work

Argumentation patterns may recall to mind the structure of syllogisms, rules as modus ponens or argumentation schemes. Reed et al. [14] explain argumentation schemes as argument forms that represent inferential structures of arguments used in everyday discourse, and in special contexts like legal argumentation and artificial intelligence. Besides forms of reasoning like modus ponens, some of the most common schemes are neither deductive nor inductive, but defeasible and presumptive. One of the issues which brings argumentation theory and computer science closer together is the need to diagram such arguments [14]. Diagramming is of interest both to those in argumentation as a tool in the analytical toolbox, and to computer scientists as a precursor to implementable formalization. We agree about the relevance of diagrams in representing the relationships of the arguments but, as we have shown in the paper, it is not precise enough to define all the relations among the arguments.

Our patterns together with their multi-labeling and constraints can be compared to the abstract dialectical frameworks, defined by Brewka and Woltran [6], and their acceptance functions. They provide a generalization of Dung's argumentation framework. Such a framework is defined as a tuple $D = (S, L, C)$ where S is a set of nodes, $L \subseteq S \times S$ is a set of links and C is an acceptance condition associated to each node. C_s specifies the exact conditions under which argument s is accepted. Summarizing, if for some $R \subseteq par(s)$, where $par(s)$ are the parents of node s, we have $C_s(R) = in$ then s will be accepted provided

the nodes in R are accepted. We can express the acceptability condition with a conjunction pattern where the set R contains the arguments $a_1, \ldots, a_n$ and argument s corresponds to our argument b. An advantage in using patterns is that we can compose them together to form a larger pattern while Brewka and Woltran [6] need to define the acceptance function from scratch.

5 Conclusions

The success criteria of argumentation patterns lies for the 90% in the proposed visualization. The contribution of this paper with respect to visualization is to use transistors for second-order attack patterns, and introduce visualizations for the conjunction and disjunction patterns inspired by visualizations of AND and OR gates in boolean circuits. Moreover, we show how these visualizations can be combined, as in the case of the Toulmin scheme.

Argumentation patterns are reusable solutions to common problems in argumentation theory, and are driven by practical rather than theoretical concerns. We define argumentation patterns by a multi-labeling, i.e., the labels of the arguments inside the pattern, together with a set of constraints showing the relations among the arguments, even if some of them are attacked by arguments external to the pattern.

We identify, among others, the following patterns in the argumentation literature, and formalize them in our framework: the support relation, the Toulmin scheme, second-order attacks, accrual, and the standards of proof. Patterns avoid us to define extended argumentation frameworks *ad hoc* for particular application domains.

Two main points emerge from this initial exploration of how to visualize and formalize argumentation patterns. First, the language has to distinguish the description of the behaviour of the pattern as standalone framework, and it has to contain a description of how the behaviour of the pattern changes when it is attacked from outside the framework. In this paper, we use multi-labelling for the former, and constraints for the latter. The general point is that a pattern definition has to provide the definition of part of an argumentation framework, or an argumentation framework in an environment. The SCC recursive scheme [2] can bring some inspiration since here also Dung's semantics are associated to a context to define the base function. The second technical issue which is emerged is the soundness and completeness proofs needed for patterns. We define the semantics of patterns in terms of multi-labelling and constraints, then the syntax in terms of flattening. We need to show now that they are equivalent. All this is left as future work.

References

1. Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. AFRA: Argumentation framework with recursive attacks. *Int. J. Approx. Reasoning*, 52(1):19–37, 2011.

2. Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. Scc-recursiveness: a general schema for argumentation semantics. *Artif. Intell.*, 168(1-2):162–210, 2005.
3. Guido Boella, Dov M. Gabbay, Leendert van der Torre, and Serena Villata. Meta-argumentation modelling i: Methodology and techniques. *Studia Logica*, 93(2-3):297–355, 2009.
4. Guido Boella, Dov M. Gabbay, Leendert van der Torre, and Serena Villata. Support in abstract argumentation. In *Procs. of the 3rd Int. Conf. on Computational Models of Argument*, pages 40–51. Frontiers in Artificial Intelligence, IOS Press, 2010.
5. Guido Boella, Leendert van der Torre, and Serena Villata. Analyzing cooperation in iterative social network design. *Journal of Universal Computer Science*, 15(13):2676–2700, 2009.
6. Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *KR*. AAAI Press, 2010.
7. Martin Caminada. On the issue of reinstatement in argumentation. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *JELIA*, volume 4160 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2006.
8. Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Coalitions of arguments: A tool for handling bipolar argumentation frameworks. *Int. J. Intell. Syst.*, 25(1):83–109, 2010.
9. Phan Minh 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.
10. Dov M. Gabbay. Fibring argumentation frames. *Studia Logica*, 93(2-3):231–295, 2009.
11. Hadassa Jakobovits and Dirk Vermeir. Robust semantics for argumentation frameworks. *J. Log. Comput.*, 9(2):215–261, 1999.
12. Jérôme Lang and Pierre Marquis. Reasoning under inconsistency: A forgetting-based approach. *Artif. Intell.*, 174(12-13):799–823, 2010.
13. S. Modgil and T.J.M Bench-Capon. Metalevel argumentation. Technical report, page www.csc.liv.ac.uk/research/ techreports/techreports.html, 2009.
14. Chris Reed, Douglas Walton, and Fabrizio Macagno. Argument diagramming in logic, law and artificial intelligence. *Knowledge Eng. Review*, 22(1):87–109, 2007.
15. Stephen Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.
16. Adam Wyner, Trevor Bench-capon, and Paul Dunne. Instantiating knowledge bases in abstract argumentation frameworks. In *The Uses of Computational Argumentation: Papers from the AAAI Fall Symposium*, pages 76–83, 2009.