Encoding Argument Graphs in Logic

Philippe Besnard, Sylvie Doutre, and Andreas Herzig

IRIT-CNRS, University of Toulouse, France
{besnard,doutre,herzig}@irit.fr

Abstract. Argument graphs are a common way to model argumentative reasoning. For reasoning or computational purposes, such graphs may have to be encoded in a given logic. This paper aims at providing a systematic approach for this encoding. This approach relies upon a general, principle-based characterization of argumentation semantics.

1 Introduction

In order to provide a method to reason about argument graphs [1], Besnard and Doutre first proposed encodings of such graphs and semantics in propositional logic [2]. Further work by different authors following the same idea was published later, e.g. [3–7]. However, all these approaches were devoted to specific cases in the sense that for each semantics, a dedicated encoding was proposed from scratch. We aim here at a generalization, by defining a *systematic* approach to encoding argument graphs (which are digraphs) and their semantics in a logic $\vdash$. Said differently, our objective is to capture the extensions under a given semantics of an argument graph in a given logic (be it propositional logic or any other logic). We hence generalize the approach originally introduced in [2] by parametrizing the encoding in various ways, including principles defining a given semantics.

We consider abstract arguments first, and then provide guidelines to extend the approach to structured arguments (made up of a support that infers a conclusion).

2 Argument Graph and Semantics

2.1 Reminder

The notion of an argument graph has been introduced by Dung in [1][1].

Definition 1. *An* argument graph *is a couple* $G = (\mathcal{A}, \mathfrak{R})$ *such that* $\mathcal{A}$ *is a finite set and* $\mathfrak{R} \subseteq \mathcal{A} \times \mathcal{A}$ *is a binary relation over* $\mathcal{A}$.

The elements of the set of vertices $\mathcal{A}$ are viewed as a set of abstract arguments, the origin and the structure of which are unspecified. The edges $\mathfrak{R}$ represent attacks: $(a, b) \in \mathfrak{R}$, also written $a\mathfrak{R}b$, means that a attacks b. A set of arguments S attacks an argument a if a is attacked by some element of S.

[1] Dung uses the term argumentation framework instead of argument graph.

Dung introduced several semantics to define which sets of arguments can be considered as collectively acceptable: the admissible, stable, grounded, preferred and complete semantics. The application of a semantics to a given argument graph results in a set of acceptable sets, called extensions. As an example of a semantics, one may consider the stable semantics [1].

Definition 2. *Given an argument graph $G = (\mathcal{A}, \mathfrak{R})$, a* stable extension $S \subseteq \mathcal{A}$ *is a set that satisfies the following two conditions:*

1. *it does not exist two arguments a and b in S such that $a\mathfrak{R}b$;*
2. *for each argument $b \notin S$, there exists $a \in S$ such that $a\mathfrak{R}b$ (any argument outside the extension is attacked by the extension).*

More generally, a *semantics* gives a formal definition of a method ruling the argument evaluation process. Extensions under a given semantics σ are called *σ-extensions*. $\mathcal{E}_\sigma(G)$ denotes the set of the σ-extensions of an argument graph G. Following Dung, a huge range of semantics have been defined (see [8] for a comprehensive overview). For these semantics, the following notions are essential (where an argument graph $G = (\mathcal{A}, \mathfrak{R})$ is assumed).

A set $S \subseteq \mathcal{A}$ is *conflict-free* iff $\nexists a, b \in S$ such that $a\mathfrak{R}b$.

An argument $a \in \mathcal{A}$ is *defended* by $S \subseteq \mathcal{A}$ iff $\forall b$ such that $b\mathfrak{R}a, \exists c \in S$ such that $c\mathfrak{R}b$.

A set of extensions $\mathcal{E} \subseteq \mathcal{E}_\sigma(G)$ is *inclusive-maximal* iff $\forall E_1, E_2 \in \mathcal{E}$, if $E_1 \subseteq E_2$ then $E_1 = E_2$.

An *admissible set* is a conflict-free set that defends all its elements.

A stable extension is an admissible set, but not all admissible sets are stable extensions.

The set of *preferred extensions* of an argument graph is the inclusive-maximal set of its admissible sets.

A number of complexity results have been established for decision problems in abstract argument graphs [9]. Two such problems are:

Verification $\mathbf{VER}_\sigma$. Given a semantics σ, an argument graph $G = (\mathcal{A}, \mathfrak{R})$ and a set $S \subseteq A$, is S a σ-extension of G?

Existence $\mathbf{EX}_\sigma$. Given a semantics σ and an argument graph $G = (\mathcal{A}, \mathfrak{R})$, does G have at least one σ-extension?

For instance, as regards the verification problem [9]: VER_{stable} is in P, but $\text{VER}_{preferred}$ is coNP-complete. As regards the existence problem, the question of the existence of a stable extension, EX_{stable}, is NP-complete.

2.2 Encoding

Given any semantics σ, our objective is to capture the σ-extensions of an argument graph $(\mathcal{A}, \mathfrak{R})$ in a logic $\vdash$. The only requirements for this logic are that it should contain all the Boolean connectives (in order to capture "not", "and", and "or").

There are two ways to achieve our objective:

(α) *By providing a formula θ_σ whose models characterize the set $\mathcal{E}_\sigma(G)$ of σ-extensions of $G = (\mathcal{A}, \mathfrak{R})$.* So the set $\mathrm{Mod}(\theta_\sigma)$ of the models of θ_σ is isomorphic to the set of σ-extensions of $(\mathcal{A}, \mathfrak{R})$: every model of θ_σ determines a σ-extension of $(\mathcal{A}, \mathfrak{R})$ and vice-versa. [2]

(β) *By providing a formula $\theta_{\sigma,\mathcal{S}}$, depending on a subset $\mathcal{S}$ of $\mathcal{A}$, that is satisfiable if and only if $\mathcal{S}$ is a σ-extension of $(\mathcal{A}, \mathfrak{R})$.*

Adopting terminology from [2], we call (α) "the model checking approach" and (β) "the satisfiability approach" (answering the verification problem VER_σ).

In (α), we must provide a means to identify extensions in the encoding. (There might for instance be non-effective ways for a model to coincide with an extension.) In the rest of the paper, we will focus on the (β) approach.

An additional issue (γ) may be to find a formula (in the logic $\vdash$) that is satisfiable iff there exists a σ-extension for the argument graph (existence problem EX_σ). This issue is of interest for the stable semantics for instance, but not for other admissibility-based semantics (preferred, complete, grounded...), the empty set being always an admissible set.

3 Encoding Methodology

Now, we provide a methodology for encoding the σ-extensions of an argument graph in a given logic, following the satisfiability approach previously introduced. We are going to illustrate it by a case study in Section 5.

3.1 Encoding Extensions

At the abstract level, given a set of abstract arguments $\mathcal{A} = \{a_1, a_2, \ldots\}$ and an argument graph $G = (\mathcal{A}, \mathfrak{R})$, in order to construct $\theta_{\sigma,S}$, the following questions should be answered:

1. How to represent a subset S of the set of arguments $\mathcal{A}$? For instance, it could be:
$$\chi_S = \bigwedge_{a_i \in S} a_i \wedge \bigwedge_{a_j \notin S} \neg a_j$$
2. How to define that S is a σ-extension of G? For instance, if σ is the stable semantics then we might have:
$$S \text{ is a stable extension of } G \text{ iff } \chi_S \models \bigwedge_{a \in \mathcal{A}} (a \leftrightarrow \bigwedge_{b \in \mathcal{A}: b \mathfrak{R} a} \neg b)$$
In [2], it was shown that $\bigwedge_{a\in\mathcal{A}}(a \leftrightarrow \bigwedge_{b\in\mathcal{A}:b\mathfrak{R}a} \neg b) = \theta_{stable}$. More generally, we are aiming at constructing $\theta_{\sigma,S}$ enjoying the following equivalence: S is a σ-extension of G iff $\chi_S \models \theta_\sigma$, i.e.,
$$\theta_{\sigma,S} \text{ is satisfiable iff } \chi_S \models \theta_\sigma$$

[2] In the case that $\mathrm{Mod}(\theta_\sigma)$ is isomorphic to $\mathcal{E}_\sigma(G)$ then the following consequence holds: $\theta_\sigma \vdash \varphi$ iff φ encodes a $\vdash$-definable property of G.

3. When a semantics involves a notion of maximality or minimality, how to capture the corresponding sets?

3.2 Encoding Set-Theoretic Relations

Our methodology for systematic encodings $\theta_{\sigma,S}$ where S *is the subset to be tested for being an extension* relies on several building bricks and a rule as follows.

– **Rule**
An encoding is of the form

$$\theta_{\sigma,S} = \varphi_S \wedge \overline{\varphi}_S \wedge \Psi_S$$

where φ_S encodes the necessary conditions for membership in S, $\overline{\varphi}_S$ encodes the sufficient conditions, and Ψ_S is a Boolean combination over basic building bricks (intuitively, Ψ_S expresses that S enjoys σ).

– **Basic Building Bricks**[3]
- *Membership in a subset of the arguments:* a is an argument in $X \subseteq \mathcal{A}$ is encoded as

$$\varphi_{(a\in X)} = \begin{cases} \varphi_{(a\in S)} & \text{if } X = S \\ \bigvee\limits_{a=x\in X} \top & \text{if } X \neq S \end{cases}$$

  where for each $a \in \mathcal{A}$, we assume a generic formula[4] $\varphi_{(a\in S)}$ expressing that **"a is in the set of arguments S"**.
- *Subset X of the arguments:*

$$\varphi_X = \bigwedge_{a\in X} \varphi_{(a\in X)}$$

- *Complement of X in the set of arguments:*

$$\overline{\varphi}_X = \bigwedge_{a\notin X} \neg\varphi_{(a\in X)}$$

  Intuitively, φ_S expresses that S contains all the elements of S whereas $\overline{\varphi}_S$ expresses that S contains only elements of S.

[3] Remember that an empty conjunction, i.e., a conjunction $\bigwedge_{\mathfrak{C}(x)} \gamma[x]$ whose condition $\mathfrak{C}(x)$ holds for no x, amounts to $\top$. An empty disjunction, i.e., a disjunction $\bigvee_{\mathfrak{C}(x)} \gamma[x]$ whose condition $\mathfrak{C}(x)$ holds for no x, amounts to $\bot$.

[4] By generic formula, we mean a formula that is constructed in a systematic way, by contrast to ad-hoc formulas with no common form. For example, $\varphi_{(a\in S)}$ can be a (provided that, for all arguments a, there is an atom a in the language). This is generic because all such formulas have the same form: an atom naming an argument.

– **Intermediate Building Bricks**
  1. $X \subseteq Y$
     This is captured as
     $$\bigwedge_{a \in \mathcal{A}} \left(\varphi_{(a \in X)} \rightarrow \varphi_{(a \in Y)}\right)$$
  2. *X is maximal such that Ψ_X holds*
     This, which amounts to Ψ_X & $\forall Y \supseteq X$ $(\Psi_Y \rightarrow Y \subseteq X)$, is captured as
     $$\Psi_X \wedge \bigwedge_{X \subseteq Y \in 2^{\mathcal{A}}} \left(\Psi_Y \rightarrow \bigwedge_{a \in \mathcal{A}} \left(\varphi_{(a \in Y)} \rightarrow \varphi_{(a \in X)}\right)\right)$$
  3. *X is minimal such that Ψ_X holds*
     This, which amounts to Ψ_X & $\forall Y \subseteq X$ $(\Psi_Y \rightarrow X \subseteq Y)$, is captured as
     $$\Psi_X \wedge \bigwedge_{X \supseteq Y \in 2^{\mathcal{A}}} \left(\Psi_Y \rightarrow \bigwedge_{a \in \mathcal{A}} \left(\varphi_{(a \in X)} \rightarrow \varphi_{(a \in Y)}\right)\right)$$

Please bear in mind that, as a consequence of the first building brick, the following holds: In all of the above clauses, whenever $X \neq S$ (and similarly for Y and Z), $\varphi_{(a \in X)}$ *must* be encoded as $\bigvee_{a=x \in X} \top$ else it is encoded as $\varphi_{(a \in S)}$.

As an illustration, here are the details for the case of maximality:

max: Let us assume that E satisfies Ψ' (hence Ψ'_E holds). Then, for all $S \subset E$,

$$\theta_{\sigma,S} \stackrel{\text{def}}{=} \varphi_S \wedge \overline{\varphi}_S \wedge \Psi'_S \wedge \bigwedge_{S \subseteq Y \in 2^{\mathcal{A}}} \left(\Psi'_Y \rightarrow \bigwedge_{a \in \mathcal{A}} \left(\varphi_{(a \in Y)} \rightarrow \varphi_{(a \in S)}\right)\right)$$

is not satisfiable *because* the conjunct $\overline{\varphi}_S$ is contradicted by means of $\Psi'_Y \rightarrow \varphi_Y$ (for the case $Y = E$); for $a \in E \setminus S$, it happens that $\overline{\varphi}_S$ entails $\neg\varphi_{(a \in S)}$ whereas $\Psi'_E \rightarrow \varphi_E$ yields $\varphi_{(a \in S)}$ (remember, Ψ'_E holds).

4 Encoding Semantic Principles

Baroni and Giacomin have shown in [10] that the existing argumentation semantics satisfy a number of principles. They have provided a comprehensive list of such principles. From some subsets of these principles, it is possible to characterize existing semantics. Based on such a general characterization of a semantics, the objective in this section is to encode into formulas the principles $P_1 \ldots P_n$ that define the semantics.

This requires two things. One is that we must prepare, from the building bricks listed in Section 3.2, encodings of statements (and their denials) such as:

– "a is in E"
– "a attacks b" (in symbols, $a \mathfrak{R} b$) and set versions thereof

– "S is maximal such that..."
– ...

The other thing is to provide a concrete list of such principles P. In Section 2 we have already mentioned conflict-freeness, inclusion-maximality, and admissibility. We recall here some of the list in [10]:

Conflict-Free Principle. A semantics σ satisfies the C-F principle iff $\forall G, \forall E \in \mathcal{E}_\sigma(G)$, E is conflict-free.
From the building bricks in Section 3.2, conflict-freeness can be encoded as[5]

$$\bigwedge_{b\mathfrak{R}a} \neg(\varphi_{(a\in S)} \wedge \varphi_{(b\in S)})$$

Inclusion-Maximality Criterion. A semantics σ satisfies the I-M criterion iff $\forall G$, $\mathcal{E}_\sigma(G)$ is inclusive-maximal.
Here, an encoding has been already explicitly given in Section 3.2.

Encoding defence (for all b such that $b\mathfrak{R}a$, there exists $c \in S$ such that $c\mathfrak{R}b$) is achieved by

$$\bigwedge_{b\mathfrak{R}a} \bigvee_{c\mathfrak{R}b} \varphi_{(c\in S)}$$

which can be used in the encoding of the next three principles that are based on defence, as follows.

Admissibility Criterion. A semantics σ satisfies the admissibility criterion iff $\forall G$, $\forall E \in \mathcal{E}_\sigma(G)$, if $a \in E$ then a is defended by E.
The admissibility criterion can be captured through

$$\varphi_{(a\in S)} \rightarrow (\bigwedge_{b\mathfrak{R}a} \bigvee_{c\mathfrak{R}b} \varphi_{(c\in S)})$$

Reinstatement Criterion. A semantics σ satisfies the reinstatement criterion iff $\forall G$, $\forall E \in \mathcal{E}_\sigma(G)$, if a is defended by E then $a \in E$.
The reinstatement criterion can be captured by means of

$$(\bigwedge_{b\mathfrak{R}a} \bigvee_{c\mathfrak{R}b} \varphi_{(c\in S)}) \rightarrow \varphi_{(a\in S)}$$

Conflict-Free Reinstatement Criterion. A semantics σ satisfies the CFR criterion iff $\forall G$, $\forall E \in \mathcal{E}_\sigma(G)$, if a is defended by E and $E \cup \{a\}$ is conflict-free then $a \in E$.
Now, the CFR criterion can be captured just as the reinstatement criterion, only adding (in the antecedent) a conjunct for conflict-freeness (see above the conflict-free principle).

[5] Please observe that S instead of E occurs in the specification of the formula because we construct a formula (parameterized by S) which is satisfiable iff S is an extension.

In addition to the criteria listed in [10], we propose the following one:

Complement Attack Criterion. A semantics σ satisfies this criterion iff $\forall G$, $\forall E \in \mathcal{E}_\sigma(G)$ it holds that $\forall b \in \mathcal{A}$ if $b \notin E$ then $\exists a \in E$ such that $a \mathfrak{R} b$. The complement attack criterion can be captured by means of

$$\bigwedge_{a \in \mathcal{A}} \left(\neg \varphi_{(a \in S)} \rightarrow \left(\bigvee_{b \mathfrak{R} a} \varphi_{(b \in S)} \right) \right)$$

Another encoding is to be found in the example detailed in Section 5.

The stable semantics is characterized by the conflict-free principle and the complement attack criterion. The stable semantics satisfies the admissibility criterion as well.

5 A Case Study

There are two approaches, (a) and (f). One approach (a) introduces dedicated atoms in the object language to represent the attack relation. This means that there is a list of fresh Att_{xy} atoms, one for each ordered pair of nodes (x, y) in the graph. The other approach (f) dispenses from such atoms, and the properties to be encoded must be expressed in such a way that reference to attacks can be captured as a range over conjunction or disjunction.

Example

Let us find $\theta_{\sigma,S}$ for $\sigma = \mathit{stable}$. The definition for a set of arguments S being conflict-free is

$$\forall a \in S \; \nexists b \in S \quad b \mathfrak{R} a$$

According to the (f)-approach, i.e., no dedicated atom is used, we must reformulate the condition as follows: *for all a in S, it is not the case that there exists some b attacking a such that b is in S*. The (f)-encoding for being conflict-free is

$$\bigwedge_{a \in S} \neg \bigvee_{b \mathfrak{R} a} \varphi_{(b \in S)}$$

or, equivalently,

$$\bigwedge_{a \in S} \bigwedge_{b \mathfrak{R} a} \neg \varphi_{(b \in S)}$$

In the case of stable extensions, we need to additionally encode the property of S attacking its complement:

$$\forall a \notin S \exists b \in S \quad b \mathfrak{R} a$$

According to the (f)-approach, we must reformulate the condition in the form: *for all a not in S, there exists some b attacking a such that b is in S*. Hence the (f)-encoding for S attacking its complement is

$$\bigwedge_{a \notin S} \bigvee_{b \mathfrak{R} a} \varphi_{(b \in S)}$$

Combining both conditions, we obtain the building brick Ψ_S (introduced above: the general case) for stable extensions:

$$\Psi_S \quad = \quad (\bigwedge_{a\in S}\bigwedge_{b\mathfrak{R}a}\neg\varphi_{(b\in S)}) \quad \wedge \quad (\bigwedge_{a\notin S}\bigvee_{b\mathfrak{R}a}\varphi_{(b\in S)})$$

Conjoining with φ_S and $\overline{\varphi}_S$, we obtain:

$$\varphi_S \wedge \overline{\varphi}_S \wedge \Psi_S \quad = \bigwedge_{a\in S} \quad (\varphi_{(a\in S)} \wedge \bigwedge_{b\mathfrak{R}a}\neg\varphi_{(b\in S)}) \quad \wedge \bigwedge_{a\notin S} \quad (\neg\varphi_{(a\in S)} \wedge \bigvee_{b\mathfrak{R}a}\varphi_{(b\in S)})$$

which is exactly the formula encoding stable extensions in Proposition 10 of [2] where $\varphi_{(x\in S)}$ is the atom x.

6 Towards Encoding Graphs of Structured Arguments

The aim of this section is to indicate how the approach, designed for abstract argument graphs, may be extended to graphs with structured arguments. Our exposition follows [11].

The very first issue is to choose how to represent nodes of an argumentation graph in this case. Then, arises the question of the representation of edges, and next, of how to define the extensions. Moreover, the properties of the logic underlying the arguments play a role now.

We assume that nodes in the graph (i.e., arguments) enjoy a minimal amount of structure as pairs (A_i, c_i) where A_i is a set of formulas, the premises of the argument, and c_i is a formula, the claim of the argument. Importantly,

$$A_i \nvdash c_i$$

In any case, our approach involves the following steps.

1. Representing nodes, with two options:
   - Either every two arguments of the form (A_i, c_i) and (A_i, c_i') are treated as two distinct entities,
   - or they are treated as equals.

   The former might be justified on the grounds that we have some (granted, rudimentary) structure within an argument (it is less abstract) and there must be the possibility to detail the content of the attack relation. We choose to leave these two options open.
2. Representing edges, with the question:
   - Is it necessary that the attack relation be captured at the object level, i.e., by a formula of the logic?

   Whether representing or not (i.e., simply capturing constraints and conditions to be satisfied by the attack relation), we would need the language to include something capturing consistency (and presumably, inference, hence the need for the deduction theorem). E.g., there could be a modal possibility operator $\diamond$ and a naming device $\lceil\cdot\rceil$. All these would express consistency of support of two arguments in an extension, as for example in

$$\diamond(A_i \wedge A_j) \rightarrow \neg\lceil i\mathfrak{R}j\rceil.$$

Another option would be to use QBF, as done in [12] to characterize the complexity, or to use Dynamic Logic of Propositional Assignments, as done in [13] to provide a dynamic account of the construction of extensions.

3. Representing extensions.
   As a start, we must decide whether $\neg A_i$ is to mean that (A_i, c_i) fails to be in the extensions being tested. The answer determines whether or not an argument with the same support as an argument in the extension at hand has to be in the extension.

   Another point is that extensions are defined as conditions using the attack relation. This may mean building bricks, other than those we have examined, from which various notions of extensions can be defined.
4. Lastly, attention must be paid to properties of the logic that can play a role. For example, contraposition and transitivity make the inference constraint to turn the conflict between arguments i and j into $A_i \Vdash \neg A_j$ as follows:

$$\begin{array}{ll} A_i \Vdash \neg c_j & \text{(assumption)} \\ A_j \Vdash c_j & \text{(assumption)} \\ \neg c_j \Vdash \neg A_j & \text{(contraposition)} \\ A_i \Vdash \neg A_j & \text{(transitivity)} \end{array}$$

7 Conclusion

We have proposed in this paper a methodology to encode an argument graph and a semantics in a given logic. Few constraints are imposed on the logic. We have considered abstract arguments, and we have given guidelines to extend the approach to structured, non-abstract arguments.

In this methodology, we have made the assumption that a semantics is defined by a set of principles. Even though evaluation principles have been put forward by [8, 10], the characterization of a semantics by a set of principles is an issue that has not been addressed yet. The example of the complement attack criterion (mandatory to characterize the stable semantics), shows that in current approaches, criteria are missing to obtain such a characterization.

In line with such a general characterization, it can be noticed that [14] contains a general recursive schema that captures all of Dung's semantics and that is able to capture other admissibility-based, or non-admissibility-based, semantics. However, the schema embeds a "base function", that basically characterizes the extensions of an argument graph made of only one strongly connected component; that is, the semantics is not described by the principles it is based on.

As regards computational issues, [15] surveys implementations of abstract argument graphs. Many abstract semantics have been implemented, with various techniques, but none of these implementations is built from a principle-based characterization of the semantics.

We have encoded in this paper a number of semantic principles, but others remain to be defined (and encoded) in order to characterize the existing semantics, and possibly, new semantics as well.

Moreover, the guidelines given to extend the approach to structured, non-abstract arguments, will be further explored.

Acknowledgements. This work benefited from the support of the AMANDE project (ANR-13-BS02-0004) of the French National Research Agency (ANR).

References

1. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
2. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: Delgrande, J.P., Schaub, T. (eds.) Proc. NMR 2004, pp. 59–64 (2004)
3. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation 1(2), 147–177 (2010)
4. Amgoud, L., Devred, C.: Argumentation frameworks as constraint satisfaction problems. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 110–122. Springer, Heidelberg (2011)
5. Walicki, M., Dyrkolbotn, S.: Finding kernels or solving SAT. Discrete Algorithms 10, 146–164 (2012)
6. Dyrkolbotn, S.K.: The same, similar, or just completely different? Equivalence for argumentation in light of logic. In: Libkin, L., Kohlenbach, U., de Queiroz, R. (eds.) WoLLIC 2013. LNCS, vol. 8071, pp. 96–110. Springer, Heidelberg (2013)
7. Nofal, S., Atkinson, K., Dunne, P.E.: Algorithms for decision problems in argument systems under preferred semantics. Artificial Intelligence 207, 23–51 (2014)
8. Baroni, P., Giacomin, M.: Semantics of abstract argument systems. In: Simari, G., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 25–44. Springer (2009)
9. Dunne, P.E., Wooldridge, M.: Complexity of abstract argumentation. In: Simari, G., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 85–104. Springer (2009)
10. Baroni, P., Giacomin, M.: On principle-based evaluation of extension-based argumentation semantics. Artificial Intelligence 171(10), 675–700 (2007)
11. Besnard, P., Garcia, A., Hunter, A., Modgil, S., Prakken, H., Simari, G., Toni, F.: Special issue: Tutorials on structured argumentation. Argument and Computation 5(1) (2014)
12. Arieli, O., Caminada, M.W.: A QBF-based formalization of abstract argumentation semantics. Journal of Applied Logic 11(2), 229–252 (2013)
13. Doutre, S., Herzig, A., Perrussel, L.: A dynamic logic framework for abstract argumentation. In: Baral, C., De Giacomo, G. (eds.) Proc. KR 2014. AAAI Press (2014)
14. Baroni, P., Giacomin, M., Guida, G.: SCC-recursiveness: A general schema for argumentation semantics. Artificial Intelligence 168(1), 162–210 (2005)
15. Charwat, G., Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Implementing abstract argumentation — A survey. Technical Report DBAI-TR-2013-82, Technische Universität Wien, Fakultät für Informatik, Vienna, Austria (2013)