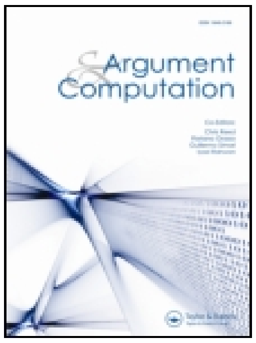


Argument & Computation

ISSN: 1946-2166 (Print) 1946-2174 (Online) Journal homepage: http://www.tandfonline.com/loi/tarc20

A tutorial on assumption-based argumentation

Francesca Toni

To cite this article: Francesca Toni (2014) A tutorial on assumption-based argumentation, Argument & Computation, 5:1, 89-117, DOI: 10.1080/19462166.2013.869878

To link to this article: http://dx.doi.org/10.1080/19462166.2013.869878

Published online: 11 Feb 2014.

Argument and Computation, 2014
Vol. 5, No. 1, 89–117, http://dx.doi.org/10.1080/19462166.2013.869878

A tutorial on assumption-based argumentation

Francesca Toni*

Department of Computing, Imperial College London, South Kensington Campus, London SW72AZ, UK

(*Received 6 November 2013; final version received 19 November 2013*)

We give an introductory tutorial to assumption-based argumentation (referred to as ABA) – a form of argumentation where arguments and attacks are notions derived from primitive notions of rules in a deductive system, assumptions and contraries thereof. ABA is equipped with different semantics for determining 'winning' sets of assumptions and – interchangeably and equivalently – 'winning' sets of arguments. It is also equipped with a catalogue of computational techniques to determine whether given conclusions can be supported by a 'winning' set of arguments. These are in the form of disputes between (fictional) proponent and opponent players, provably correct w.r.t. the semantics. Albeit simple, ABA is powerful in that it can be used to represent and reason with a number of problems in AI and beyond: non-monotonic reasoning, preferences, decisions. While doing so, it encompasses the expressive and computational needs of these problems while affording the transparency and explanatory power of argumentation.

Keywords: argumentation; semantics; computation; disputes

1. Introduction

From the late 1980s, argumentation has emerged in AI as a powerful method for representing a diverse range of knowledge and supporting several kinds of reasoning. Starting with Lin and Shoham (1989), Dung (1991), Kakas, Kowalski, and Toni (1992), Kakas, Kowalski, and Toni (1998) and Bondarenko, Toni, and Kowalski (1993), argumentation was identified as a way for understanding and reconciling differences (as well as point out similarities) of various existing formalisms for non-monotonic, default reasoning as studied in AI (such as default logic (Reiter, 1980) and circumscription (McCarthy, 1980)). This line of research led to the development of two argumentation frameworks: abstract argumentation (AA) (Dung, 1995) and assumption-based argumentation (ABA) (Bondarenko, Dung, Kowalski, & Toni, 1997). Moreover, starting with Pollock (1987), argumentation was identified as a way to understand defeasible reasoning as studied in philosophy, leading, in particular, to DeLP (García & Simari, 2004) and ASPIC+ (Modgil & Prakken, 2013). Furthermore, Krause, Ambler, and Fox (1992), Krause, Ambler, Elvang-Gøransson, and Fox (1995) pointed to an important role for argumentation to support decision-making (in general and in a medical setting), leading to argumentation frameworks such as that of Amgoud and Prade (2009). Finally, Elvang-Gøransson and Hunter (1995) used argumentation for resolving inconsistencies in logic-based reasoning, leading to the argumentation logic of Besnard and Hunter (2008).

This paper provides an introductory tutorial to ABA, while also stressing and exemplifying its use for a wide range of reasoning tasks, including non-monotonic reasoning (for which it was initially proposed), defeasible reasoning, decision-making and reasoning with inconsistent

*Email: f.toni@imperial.ac.uk

information. The tutorial is meant to serve as a reference manual on ABA and argumentation for students and researchers alike.

ABA is closely related to AA, in that it is an instance thereof while at the same time admitting AA as an instance (see Toni, 2012). However, AA takes as given a set of arguments and an attack relationship between arguments in this set, and focuses on ascertaining which sets of arguments can be deemed to be ‘winning’ (according to a catalogue of different, the so-called- semantics). Instead, in ABA arguments and attacks are derived from given rules in a deductive system, assumptions, and their contraries. However, ABA is abstract too, in the sense that, like AA, it does not commit to any specific instance and admits, instead, several instances (including AA). However, ABA’s building blocks (rules, assumptions, contraries) are at a lower level of abstraction than those of AA (arguments and attacks).

ABA is equipped with different semantics for determining ‘winning’ sets of assumptions. The ‘winning’ sets of arguments (à-la-AA) can then be determined from the ‘winning’ sets of assumptions, in a straightforward manner – and the assumption-level and argument-level views are fully equivalent in ABA. Manipulating assumptions directly (and arguments only indirectly) rather than arguments though is advantageous from a computational viewpoint, as it allows to (implicitly) exploit overlappings between arguments and avoid recomputation as well as terminating, even in the presence of loops in the rules, in determining whether a conclusion can be supported by a ‘winning’ set of arguments.

ABA is equipped with a catalogue of computational techniques (Dung, Kowalski, & Toni, 2006; Dung, Mancarella, & Toni, 2007; Toni 2013), which have their foundation in logic programming, and in particular the methods of Eshghi and Kowalski (1989), Dung (1991) and Kakas and Toni (1999). These techniques can be used to determine whether given conclusions can be supported by a ‘winning’ set of arguments. They are in the form of disputes, provably correct w.r.t. the semantics and incorporating various forms of ‘filtering’ for avoiding recomputation. These disputes can be seen as games between (fictional) the proponent and opponent players, and rely upon data structures for representing the state of the game at any given time. They are specified algorithmically in a way that lends itself to principled and provably correct implementations, and indeed several systems exist to support ABA-style argumentation. As a consequence, ABA can be deemed to be a fully *computational form of argumentation*.

The computational complexity of various reasoning tasks for various instances of ABA and various semantics has been studied e.g. in Dimopoulos, Nebel, and Toni (2002) and Dunne (2009). This topic will not be covered in this tutorial paper.

This paper complements the overview in Dung, Kowalski, and Toni (2009) by being an introductory tutorial, illustrating notions, justifying design choices and discussing (disregarded) alternatives.

The paper is organised as follows. In Section 2, we give a simple illustrative example of argumentation in general and ABA in particular. In Section 3, we describe and discuss ABA frameworks, including rules in a deductive system, assumptions and contraries. In Section 4, we define arguments and attacks in ABA. In Section 5, we give the main semantics for ABA, and in Section 6, we give several examples of ABA. In Section 7, we describe the computational machinery for reasoning in ABA, under two different semantics. In Section 8, we consider several frequently asked questions (FAQs) on ABA . In Section 9, we conclude.

2. A simple illustrative example

You like eating good food, this makes you really happy. But you are very health-conscious too, and do not want to eat without a fork and with dirty hands. You are having a walk with some friends, and your friend Nelly is offering you all a piece of lasagna, looking really delicious. You are not quite sure, but typically you carry no cutlery when you go for walks, and your hands will almost certainly be dirty, even if they may not look so. Should you go for the lasagna, trying to snatch it quickly from the other friends? You may argue with yourself as follows:

α: let me go for the lasagna, it looks delicious, and eating it will make me happy!
β: but I am likely to have no fork and my hands will be dirty, I don't want to eat it in these circumstances, let the others have it!

α and β can be seen as *arguments*, and β disagrees with (*attacks*) α. If these are all the arguments put forward, since no argument is proposed that *defends* α against β, the decision to go for the lasagna is not dialectically justified. If, however, you put forward the additional argument

γ: who cares about the others, let me get the lasagna, I may have a fork after all ...

this provides a defence for α against β, and the decision to go for the lasagna becomes dialectically (although possibly not ethically) justified.

The set of arguments $\{\alpha, \beta, \gamma\}$ and the attack relationship between them (that β attacks α, and γ attacks β) form an AA framework (Dung, 1995). All semantics for AA agree with our earlier analysis of what is dialectically justified by sanctioning $\{\alpha, \gamma\}$ as 'winning' arguments. Similarly, given the initial set of arguments $\{\alpha, \beta\}$ and the attack relationship between them that β attacks α, all semantics for AA agree that $\{\beta\}$ is 'winning'.

In this context, ABA provides:

- guidelines to structure knowledge and information so as to construct arguments from a repository of *rules* (e.g. 'happy' if 'eating' 'good food') and *assumptions* (e.g. 'eating') and attacks from a repository of *contraries* of assumptions (e.g. the contrary of 'eating' is 'not eating'); albeit simple, the format of these repositories is sufficiently general to represent and reason with a large class of problems;
- declarative methods (*semantics*) to determine whether *conclusions* (e.g. whether to go for the lasagna or not) are dialectically justified, according to various interpretations; these methods match corresponding methods to determine whether arguments are dialectically justified;
- computational mechanisms to construct arguments and attacks and determine and explain tenability of conclusions; these mechanisms are in the form of provably correct *disputes* and corresponding (principled) implementations, and are specified with computational considerations in mind, and in particular focus on how to avoid wasting computation time without losing correctness; correctness amounts to guaranteeing a-priori (1) *soundness*: every outcome of the disputes is dialectically justified, according to various interpretations, and (2) *completeness*: every conclusion dialectically justified can be computed by the disputes and implementations thereof, in a wide range of cases;
- alternative but equivalent (declarative and computational) views at the level of sentences (in which arguments are expressed) and (high-level, abstract) arguments and attacks, that can be used interchangeably, with the sentence level view core in keeping computation costs down, and the abstract view essential for providing helpful explanations for dialectically justified conclusions.

3. ABA frameworks

An ABA framework is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ where

- $\langle \mathcal{L}, \mathcal{R} \rangle$ is a deductive system, with $\mathcal{L}$ the *language* and $\mathcal{R}$ a set of *rules*, that we assume of the form $\sigma_0 \leftarrow \sigma_1, \ldots, \sigma_m (m \geq 0)$ with $\sigma_i \in \mathcal{L}$ $(i = 0, \ldots, m)$; σ_0 is referred to as the *head* and $\sigma_1, \ldots, \sigma_m$ as the *body* of rule $\sigma_0 \leftarrow \sigma_1, \ldots, \sigma_m$;
- $\mathcal{A} \subseteq \mathcal{L}$ is a (non-empty) set, referred to as *assumptions*;
- $\overline{}$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$; $\bar{a}$ is referred to as the *contrary* of a.

Example 3.1 $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ may consist of

$$\mathcal{L} = \{a, b, c, p, q, r, s, t\}$$
$$\mathcal{R} = \{p \leftarrow q, a,\ q \leftarrow,\ r \leftarrow b, c\}$$
$$\mathcal{A} = \{a, b, c\}$$
$$\bar{a} = r, \quad \bar{b} = s, \quad \bar{c} = t.$$

Note that we will use abstract examples such as this throughout for illustration of the technical notions, but of course these can be given a concrete, real-life reading. In Example 3.1, for instance, p may stand for ‘happy’, a for ‘eating’, q for ‘good food’, r for ‘not eating’, b for ‘no fork’, c for ‘dirty hands’, s for ‘fork’ and t for ‘clean hands’. Thus, this ABA framework may represent the belief base of a simple decision-making agent for Section 2 (although by no means the only representation possible for that agent). Here, the assumptions are either actions (a) or observations/defeasible premises (b and c) and non-assumptions (namely sentences in $\mathcal{L} \setminus \mathcal{A}$) are goals to be achieved or avoided (p, q and r). Contraries of assumptions represent reasons against actions ($\bar{a} = r$) or defeaters of defeasible premises ($\bar{b} = s$, $\bar{c} = t$). This interpretation of the given ABA framework is in line with existing approaches to model agents in computational logic (notably Kakas, Mancarella, Sadri, Stathis, & Toni, 2008; Kowalski & Sadri, 1999).

Language. The language component $\mathcal{L}$ of ABA frameworks can be omitted, as it can always be ‘reconstructed’ from the other components (being the set of all sentences occurring in them). The language is not restricted to propositional atoms (as in the earlier Example 3.1) but can be any language of sentences. Different instances of ABA have been studied (Bondarenko et al., 1997), corresponding to existing formalisms for default reasoning, some with a propositional language, some others with a first-order language, and others still with a modal language. For some studied instances of ABA, $\mathcal{L}$ is closed under classical negation $\neg$ (namely if a sentence σ is in $\mathcal{L}$ then $\neg\sigma$ is as well), whereas for some others it is not. The language may include an implication connective $\supset$ and $\mathcal{R}$ may include inference rules (e.g. modus ponens for $q \supset p$, namely $p \leftarrow q \supset p, q$) to reason with it, as discussed in (Dung et al., 2006). The language for the ABA framework in Example 3.1 includes neither negation nor implication.

Rules. The rules in $\mathcal{R}$ may be written using different syntactical conventions, for instance $p \leftarrow q, a$ may be written as $\frac{q,a}{p}$ or $q, a \Rightarrow p$, equivalently. These rules may represent domain-independent inference rules and axioms, as well as domain-dependent knowledge/information. ABA’s early versions (Bondarenko et al., 1993, 1997) as well as other frameworks for structured argumentation (García & Simari, 2004; Modgil & Prakken, 2013) single out domain-dependent and/or factual knowledge/information as a separate component (a theory $T \subseteq \mathcal{L}$ in early-days-ABA). Any theory T can be equivalently written as part of $\mathcal{R}$ as a set of rules with an empty body and is thus omitted in our presentation of ABA here.

Sometimes rules are written in the form of schemata, e.g. $p(X) \leftarrow q(X), a(X)$, using variables that are meant to be instantiated over an implicit vocabulary. For example, for vocabulary $1, f(1), f(f(1)), \ldots$, the schema $p(X) \leftarrow q(X), a(X)$ stands for the (infinite) set of rules

$$p(1) \leftarrow q(1), a(1),$$
$$p(f(1)) \leftarrow q(f(1)), a(f(1)),$$
$$p(f(f(1))) \leftarrow q(f(f(1))), a(f(f(1))), \ldots$$

with implicit language $\mathcal{L} = \{p(1), p(f(1)), \ldots, q(1), q(f(1)), \ldots, r(1), r(f(1)), \ldots\}$.

Rules can be chained to derive/deduce conclusions, where

- a *deduction for* $\sigma \in \mathcal{L}$ *supported by* $S \subseteq \mathcal{L}$ *and* $R \subseteq \mathcal{R}$, denoted $S \overset{R}{\vdash} \sigma$, is a (finite) tree with nodes labelled by sentences in $\mathcal{L}$ or by τ,[1] the root labelled by σ, leaves either τ or sentences in S, non-leaves σ' with, as children, the elements of the body of some rule in $\mathcal{R}$ with head σ', and R the set of all such rules.

The following are examples of deductions for our earlier Example 3.1:

$$\{q, a\} \overset{R_1}{\vdash} p \quad \text{for } R_1 = \{p \leftarrow q, a\},$$

$$\{\} \overset{R_2}{\vdash} q \quad \text{for } R_2 = \{q \leftarrow\},$$

$$\{a\} \overset{R_3}{\vdash} p \quad \text{for } R_3 = R_1 \cup R_2,$$

shown in Figure 1 as trees. Instead, $\{q, a, b\} \overset{R_1}{\vdash} p$ is not a deduction, due to the presence of b in the support. Indeed, b is 'irrelevant' to p, given $R_1 = \{p \leftarrow q, a\}$.

Deductions can alternatively be defined in a forward fashion (as in Bondarenko et al. (1997)) or in a backward fashion (as in Dung et al. (2006)). For example, the earlier deduction

$$\{a\} \overset{R_3}{\vdash} p$$

can be presented in a forward fashion as the sequence a, q, p and in a backward fashion as the sequence $\{p\}$, $\{q, a\}$, $\{a\}$. Each set in this latter sequence corresponds to a frontier in the construction of the tree-style presentation of the argument. Indeed, in general, the backward-style notion of deduction is fully equivalent to the tree-style notion (see Dung et al., 2009). The forward-style notion of deduction is not strictly speaking equivalent to the tree-style notion given earlier, since it loses the 'relevance' feature. For example, a, b, q, p is a forward deduction. However, for each tree-style deduction there is a forward-style deduction and for each forward-style deduction there is a tree-style deduction, possibly with a smaller support (see Dung et al. 2006; Dung, Toni, & Mancarella, 2010). All notions of deductions are equivalent for semantic purposes, as discussed later in Sections 4 and 5.

Assumptions are simply 'special' kinds of sentences in the language, playing the role of the only potential 'weak' points in arguments (namely the bits of an argument that can be attacked)

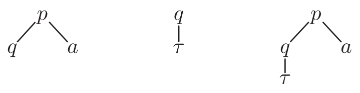


Figure 1. Deductions $\{q, a\} \overset{R_1}{\vdash} p$ (left), $\{\} \overset{R_2}{\vdash} q$ (middle) and $\{a\} \overset{R_3}{\vdash} p$ (right) for Example 3.1.

and argumentation in ABA amounts to identifying 'strong' sets of assumptions. We impose that there needs to be at least one assumption in an ABA framework as otherwise argumentation is trivial – there is no 'weak' point to debate about. Note that the (semantics and computational) machinery for ABA works perfectly well in the case in which the set of assumptions is empty, but it equates to reasoning in the deductive system $\langle \mathcal{L}, \mathcal{R} \rangle$ and is, thus, uninteresting from an argumentative viewpoint. All instances of ABA that we have studied naturally have a non-empty set of assumptions. As we discuss further in Section 8, it is easy to engineer ABA frameworks that have a non-empty set of assumptions, but if the knowledge/information to be reasoned with requires no assumptions then ABA (or argumentation in general) is definitely an overkill for the problem at hand.

In general, assumptions may be hypothesised to construct arguments (as we will see in Section 4) or derived using $\mathcal{R}$ and possibly other assumptions. We say that

- an ABA framework is *flat* iff no assumption is the head of a rule.

Thus, given a flat ABA framework, assumptions can only be hypothesised. The ABA framework in Example 3.1 is flat. The following is a simple example of non-flat ABA framework.

Example 3.2 Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ be the ABA framework with

$$\mathcal{R} = \{p \leftarrow a, c, a \leftarrow b, x \leftarrow y\}$$

$$\mathcal{A} = \{a, b, c\}$$

$$\bar{a} = x, \quad \bar{b} = y, \quad \bar{c} = z,$$

where p may stand for 'picnic', a for 'good weather', b for 'sunny', c for 'free', x for 'bad weather, y for 'rain' and z for 'at work' (basically, this simple ABA framework represents a belief base that one can hold a picnic if he/she is free and the weather is good, the weather is definitely good when it is sunny and bad when it rains, and, by default, the weather is good unless it is bad, sunny unless it rains, and the person is free unless he/she is at work). Here, assumption a is the head of a rule, and thus can be hypothesised but can also be deduced from assumption b, via the rule. Instead, assumptions b and c can only be hypothesised.

Some of the ABA instances that have been studied (Bondarenko et al., 1997) are flat (e.g. the instances corresponding to logic programming and default logic), whereas others are not (e.g. the instance corresponding to auto-epistemic logic). Non-flat ABA frameworks are computationally demanding (Dimopoulos et al., 2002) and indeed all existing computational mechanisms for ABA are defined for flat ABA frameworks. We will focus on flat ABA frameworks.

Contrary. We impose that each assumption has one single contrary sentence (by imposing that $\overline{}$ is a total mapping onto $\mathcal{L}$). Indeed, the contrary of an assumption can be seen as a 'handle' to open debate about the assumption, and thus it needs to be clear what this is. We do not impose that the contrary mapping is bijective, so different assumptions can have the same contrary. Note also that contrary is different from negation, for at least two reasons: negation may not even occur in $\mathcal{L}$, and, even if it does, contrary is only given for assumptions, whereas negation may apply to any sentence, not just assumptions. Also, contrary is not required to be symmetric (the contrary of an assumption may not even be an assumption, and even if it is it may not have the original assumption as its contrary). In other words, contrary is a very generic, flexible notion, in the same way that the attack relationship is in AA (Dung, 1995). For specific instances of ABA contrary can of course correspond to negation and be symmetric, but in general these are not requirements (as shown in our earlier Example 3.1, where contrary is not given via negation and is not symmetric).

4. ABA arguments and attacks

In ABA, *arguments* are deductions of claims using rules and supported by sets of assumptions, and *attacks* are directed at the assumptions in the support of arguments:

- *an argument for (the claim)* $\sigma \in \mathcal{L}$ *supported by* $A \subseteq \mathcal{A}$ ($A \vdash \sigma$ in short) is a deduction for σ supported by A (and some $R \subseteq \mathcal{R}$)
- *an argument* $A_1 \vdash \sigma_1$ attacks an argument $A_2 \vdash \sigma_2$ iff σ_1 is the contrary of one of the assumptions in A_2.

In the case of Example 3.1, the set of arguments consists of

$$\{\} \vdash q, \quad \{a\} \vdash p, \quad \{b, c\} \vdash r, \quad \{a\} \vdash a, \quad \{b\} \vdash b, \quad \{c\} \vdash c.$$

The first and second arguments are the middle and right deductions in Figure 1, respectively. Note that the last three arguments are (implicitly) supported by the empty set of rules, and correspond to assumptions. By definition of (tree-style) deduction, in flat ABA frameworks, any argument with an assumption a as its claim is always supported by the singleton set of assumptions $\{a\}$ and the empty set of rules. Note also that the argument $\{\} \vdash q$ is supported by the empty set of assumptions (but non-empty set of rules $\{q \leftarrow\}$). Finally, note that no other arguments exists for this example framework. For instance, $\{a, b\} \vdash p$ is not an argument, by definition of (tree-style) deduction.

In the case of Example 3.1, the attack relationship is as follows:

$$\{b, c\} \vdash r \text{ attacks } \{a\} \vdash p \text{ as well as } \{a\} \vdash a.$$

Note that there is no attack against $\{\} \vdash q$: in general, if the set of assumptions supporting an argument is empty, then that argument cannot be attacked, as attacks are solely directed at assumptions. Note also that there is no attack against $\{b, c\} \vdash r$, since there are no arguments with the contrary of b or the contrary of c as their claims. For the same reason, there are no attacks against $\{b\} \vdash b$ and $\{c\} \vdash c$. Finally, note that arguments may be attacked by the same arguments if they share assumptions in their support (as in the case of $\{a\} \vdash p$ and $\{a\} \vdash a$) and there are arguments with claim the contrary of (some of) those assumptions. Thus, for example, $\{b, c\} \vdash r$ attacks both $\{a\} \vdash p$ and $\{a\} \vdash a$.

We have seen in Section 3 that the tree-style notion of deduction can be defined alternatively in a backward or forward manner. We have (implicitly) used the tree-style notion in the earlier definition of argument, but any of the other notions can be used to give a notion of argument. The different notions of argument thus obtained are equivalent (Dung et al., 2006, 2009, 2010) as follows:

(a) α is a tree-style argument iff α is a backward-style argument;
(b) every tree-style argument is a forward-style argument;
(c) for every forward-style argument supported by $A \subseteq \mathcal{A}$ there is a tree-style argument supported by $A' \subseteq A$.

In the remainder, unless specified otherwise, all arguments are tree-style.

Attacks between arguments correspond in ABA to attacks between sets of assumptions, where

- *a set of assumptions* A *attacks a set of assumptions* A' iff an argument supported by a subset of A attacks an argument supported by a subset of A'.

In Example 3.1, $\{b, c\}$ attacks $\{a\}$ and A attacks A' for any $A \subseteq \mathcal{A}$ and $A' \subseteq \mathcal{A}$ such that $\{b, c\} \subseteq A$ and $\{a\} \subseteq A'$. The assumption-level attacks from any such A to any such A' corresponds to the argument-level attacks from $\{b, c\} \vdash r$ to $\{a\} \vdash p$ and $\{a\} \vdash a$. In general:

(a) if an argument α attacks another argument α' then the set of assumptions supporting α attacks the set of assumptions supporting α';
(b) if a set of assumptions A attacks another set of assumptions A' then some argument supported by a subset of A attacks some argument supported by a subset of A'.

The notion of attack between sets of assumptions is equivalent to the notion of attack between forward-style arguments (namely arguments defined using forward-style deductions), in that:

(a) a set of assumptions A attacks a set of assumptions A' iff there is a forward-style argument supported by A attacking a forward-style argument supported by A'.

A third, hybrid view can be taken in ABA, whereby

- *an argument α attacks a set of assumptions A* iff
  α attacks some argument α' supported by $A' \subseteq A$, or, equivalently,
  the claim of α is the contrary of an assumption in A;
- *a set of assumptions A attacks an argument α* iff
  there is an argument supported by $A' \subseteq A$ that attacks α, or, equivalently,
  A attacks the set of assumptions supporting α.

In Example 3.1, $\{b, c\} \vdash r$ attacks A for every set of assumptions $A \supseteq \{a\}$, and $\{b, c\}$ attacks $\{a\} \vdash a$ and $\{a\} \vdash p$.

This hybrid notion of attack corresponds to the notions of attack between arguments and between assumptions as follows:

(a) an argument α attacks a set of assumptions A iff α attacks some argument supported by $A' \subseteq A$;
(b) an argument α attacks a set of assumptions A iff the set of assumptions supporting α attacks A.

As a consequence of the correspondence between the three notions of attack, as we will see in the next section, they can be used interchangeably in the definition of semantics for ABA.

Some existing frameworks for structured argumentation use a notion of *sub-argument* of an argument. This notion is implicit in ABA (where a sub-argument of an argument is a sub-tree of that argument). However, the notion of sub-argument plays no role in ABA, semantically or computationally, since, by virtue of the correspondences discussed in this section, ABA operates at the level of assumptions.

5. ABA semantics

Argumentation semantics offer declarative methods to determine 'acceptable' sets of arguments, namely 'winning' sets of arguments from a dialectical viewpoint. In ABA, these semantics can equivalently be used to determine 'acceptable'/'winning' sets of assumptions. Below, we discuss and illustrate both views on argumentation semantics in ABA, as well as their equivalence (Dung et al., 2007, 2009; Toni, 2012 for more details).

Argument-level view. With arguments and attack between arguments defined for ABA, standard semantics for AA (Dung, 1995) can be applied in ABA. Formally, a *set of arguments* $\mathtt{A}$ is[2]

- *admissible* iff it does not attack itself and it attacks all arguments that attack it;
- *preferred* iff it is maximally (w.r.t. $\subseteq$) admissible[3];
- *sceptically preferred* iff it is the intersection of all preferred sets of arguments;
- *complete* iff it is admissible and contains all arguments it *defends*, where $\mathtt{A}$ defends α iff $\mathtt{A}$ attacks all arguments that attack α;
- *grounded* iff it is minimally (w.r.t. $\subseteq$) complete[4];
- *ideal* iff it is maximally (w.r.t. $\subseteq$) admissible *and* contained in all preferred sets of arguments;
- *stable* iff it does not attack itself and it attacks all arguments it does not contain.

Each such notion can be used to sanction a set of arguments as 'acceptable' or 'winning'. Given the arguments and attacks for the ABA framework in Example 3.1, the set of arguments

$$\mathtt{A} = \{\{b,c\} \vdash r, \{b\} \vdash b, \{c\} \vdash c, \{\} \vdash q\}$$

is admissible, preferred, sceptically preferred, complete, grounded, ideal and stable. Indeed, it is not attacked by any argument (and thus it is admissible and complete) and it attacks the arguments $\{a\} \vdash a$ and $\{a\} \vdash p$ it does not contain (and thus it is stable). Each $\mathtt{A}' \subset \mathtt{A}$ is also admissible, for the same reason that $\mathtt{A}$ is. Therefore, $\mathtt{A}$, being the largest admissible set, is the only preferred set, and it is sceptically preferred too. There are no other complete sets (since, in general, if an argument is not attacked by any other argument then it is defended by the empty set and any set of arguments, and thus must belong to each complete set). Therefore, $\mathtt{A}$ is also grounded. Finally, all admissible $\mathtt{A}' \subseteq \mathtt{A}$ are necessarily contained in all preferred sets (as they are contained in $\mathtt{A}$, the only preferred set) but only $\mathtt{A}$ is ideal as it is the largest such $\mathtt{A}'$.

We give below examples showing how the semantics above differ, sanctioning different sets of arguments as 'acceptable' or 'winning', adapted from Bondarenko et al. (1997) and Dung et al. (2007).

Example 5.1 Consider $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ with

$$\mathcal{R} = \{c_a \leftarrow a,\ c_a \leftarrow b,\ c_b \leftarrow a\}$$
$$\mathcal{A} = \{a, b\}$$
$$\bar{a} = c_a, \quad \bar{b} = c_b$$

Then, for example, $\{a\} \vdash c_a$ attacks itself, $\{a\} \vdash a$ and $\{a\} \vdash c_b$, and

- $\{\}$, $\{\{b\} \vdash c_a\}$, $\{\{b\} \vdash c_a, \{b\} \vdash b\}$, are all admissible,
- $\{\{b\} \vdash c_a, \{b\} \vdash b\}$ is preferred, sceptically preferred, ideal and stable,
- $\{\}$ and $\{\{b\} \vdash c_a, \{b\} \vdash b\}$ are complete,
- $\{\}$ is grounded.

Note that the empty set of arguments is always admissible. The admissible, preferred, complete and stable semantics can be deemed as 'credulous', in that they allow for alternative, sometimes questionable sets of arguments to be deemed 'winning'. For example, if $c_a \leftarrow a$ is removed from $\mathcal{R}$ in Example 5.1, then both $\{\{a\} \vdash c_b, \{a\} \vdash a\}$ and $\{\{b\} \vdash c_a, \{b\} \vdash b\}$ are preferred, stable and complete, despite each questioning the other. Instead, $\{\}$ is the only grounded, sceptically preferred and ideal set in this example. The sceptically preferred, ideal and grounded sets are always unique (Dung, 1995; Dung et al., 2007; Bondarenko et al., 1997 for proofs of these results), and can overall be classified as ‘sceptical’ semantics, sanctioning as ‘winning’ exactly one set and taking no or a limited amount of chance if there is doubt, albeit in different ways, as the following example shows.

Example 5.2 Consider $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ with

$$\mathcal{R} = \{c_a \leftarrow a,\ c_a \leftarrow b,\ c_b \leftarrow a,\ c_c \leftarrow d,\ c_d \leftarrow c,\ c_e \leftarrow c,\ c_e \leftarrow d,\ c_f \leftarrow e\}$$
$$\mathcal{A} = \{a, b, c, d, e, f\}$$
$$\bar{a} = c_a, \quad \bar{b} = c_b, \quad \bar{c} = c_c, \quad \bar{d} = c_d, \quad \bar{e} = c_e, \quad \bar{f} = c_f$$

Then

- $\{\}$ is the grounded set,
- $\{\{b\} \vdash b, \{b\} \vdash c_a\}$ is the ideal set and
- $\{\{b\} \vdash b, \{b\} \vdash c_a, \{f\} \vdash f\}$ is the sceptically preferred set.

This shows that the grounded semantics takes no chances at all (as there is no uncontroversial – unattacked – argument), the ideal semantics takes a small amount of chance (by accepting all arguments supported by b, as this is only challenged by arguments supported by a, which cannot possibly be ‘winning’), and the sceptically preferred semantics takes a moderate amount of chance (by also accepting $\{f\} \vdash f$}, defended by arguments supported by c or d, none of which are in the sceptical preferred set, as they belong to different preferred extensions).

We have seen in Example 5.1 that the ‘credulous’ semantics may also differ in determining which sets of arguments are ‘winning’. This is further illustrated by the following example:

Example 5.3 Consider $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ with

$$\mathcal{R} = \{c_a \leftarrow b,\ c_b \leftarrow a,\ c_c \leftarrow b,\ c_c \leftarrow c\}$$
$$\mathcal{A} = \{a, b, c\}$$
$$\bar{a} = c_a, \quad \bar{b} = c_b, \quad \bar{c} = c_c.$$

Then

- $\{\{b\} \vdash b, \{b\} \vdash c_a, \{b\} \vdash c_c\}$ and $\{\{a\} \vdash a, \{a\} \vdash c_b\}$ are both preferred, but
- $\{\{b\} \vdash b, \{b\} \vdash c_a, \{b\} \vdash c_c\}$ is the only stable set of arguments, as $\{\{a\} \vdash a, \{a\} \vdash c_b\}$ does not attack, e.g. $\{c\} \vdash c$.

Formal relationships amongst semantics have been widely studied (Dung, 1995; Dung et al., 2007; Bondarenko et al., 1997), e.g. every stable set of arguments is guaranteed to be preferred (but not vice versa, as the previous example shows), the grounded set is guaranteed to be a subset of every preferred set.

Assumption-level view. ABA is also equipped with notions of ‘acceptability’ of sets of assumptions, mirroring the notions of ‘acceptability’ of sets of arguments but with the notion of attack between (sets of) arguments replaced by the notion of attack between sets of assumptions. Formally, a set of assumptions A is

- *admissible* iff it does not attack itself and it attacks all sets of assumptions that attack it;
- *preferred* iff it is maximally (w.r.t. $\subseteq$) admissible;

- *sceptically preferred* iff it is the intersection of all preferred sets of assumptions;
- *complete* iff it is admissible and contains all assumptions it *defends*, where A defends a iff A attacks all sets of assumptions that attack a;
- *grounded* iff it is minimally (w.r.t. $\subseteq$) complete;
- *ideal* iff it is maximally (w.r.t. $\subseteq$) admissible *and* contained in all preferred sets of assumptions;
- *stable* iff it does not attack itself and it attacks all assumptions it does not contain.[5]

In Example 3.1, $\{b,c\}$ is admissible, preferred, sceptically preferred, complete, grounded, ideal and stable. Indeed, it is not attacked by any set of assumptions. $\{\}$, $\{b\}$, $\{c\}$ are also admissible. There are no other preferred, complete or stable sets. In Example 5.1, $\{\}$ and $\{b\}$ are admissible and complete, $\{b\}$ is preferred, sceptically preferred, ideal and stable, and $\{\}$ is grounded. In Example 5.2, $\{\}$ is grounded, $\{b\}$ is ideal and $\{b,f\}$ is sceptically preferred. In Example 5.3, $\{a\}$ and $\{b\}$ are preferred and $\{b\}$ is stable. As in the case of sets of arguments, the sceptically preferred, ideal and grounded sets of arguments are always unique, the empty set of assumptions is always admissible, every stable set is preferred etc. (see Bondarenko et al., 1997; Dung et al., 2007 for details).

Equivalently, the notions of admissible and complete sets of assumptions can be defined in terms of the hybrid notion of attack seen in Section 4, where a set of assumptions is

- *admissible* iff it does not attack itself and it attacks all *arguments* that attack it;
- *complete* iff it is admissible and contains all assumptions it *defends*, where A defends a iff A attacks all *arguments* that attack a.

These hybrid reformulations are equivalent to the earlier formulations in terms of attack between sets of assumptions, as shown in (Dung et al., 2006) for admissible sets of assumptions (the proof for complete sets is analogous to the proof for admissible). More precisely, the two formulations for admissible sanction the same sets of assumptions as 'acceptable' or 'winning' (similarly for complete). However, the hybrid reformulation lends itself to more efficient algorithms and implementations, since, in general, there are fewer arguments attacking a set of assumptions than there are sets of assumptions attacking the same set, as illustrated next.

Example 5.4 Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ have[6]

$$\mathcal{R} = \{p \leftarrow q, a,\ q \leftarrow,\ r \leftarrow b, c\} \text{ as in Example 3.1}$$
$$\mathcal{A} = \{a, b, c, d\}$$
$$\bar{a} = r, \quad \bar{b} = s, \quad \bar{c} = t \quad \text{(as in Example 3.1) and } \bar{d} = p.$$

Then $\{b,c,d\}$ is admissible (and preferred, as well as grounded). However, to see this according to the initial formulation in terms of attacks between sets of assumptions, all sets of assumptions containing a (namely $\{a\}, \{a,b\}, \{a,c\}, \{a,d\}, \ldots, \mathcal{A}$) need to be counter-attacked by $\{b,c,d\}$. Instead, according to the hybrid formulation, it is sufficient that $\{a\} \vdash p$ is counter-attacked, as this is the only argument attacking $\{b,c,d\}$ (by attacking d). Indeed, by counter-attacking $\{a\} \vdash p$ (with $\{b,c\}$), then so are all sets of assumptions containing a.

Note that the efficiency of the hybrid view is also afforded by the the argument-level view of semantics. In the case of Example 5.4, $\{\{d\} \vdash d, \{b,c\} \vdash r\}$ is admissible (and preferred), with $\{a\} \vdash p$ the only argument needing to be counter-attacked.

Correspondence between argument-level and assumption-level views. Directly from the results in (Dung et al., 2007) and (Toni, 2012), the two views over the semantics of ABA are equivalent, in the following sense:

(a) if a set $\mathtt{A}$ of arguments is admissible/preferred/sceptically preferred/complete/grounded/ideal/stable, then the union of all sets of assumptions supporting the arguments in $\mathtt{A}$ is admissible/preferred/sceptically preferred/complete/ grounded/ideal/stable (respectively);

(b) if a set A of assumptions is admissible/preferred/sceptically preferred/complete/grounded/ideal/stable, then the union of all arguments supported by any subset of A is admissible/preferred/sceptically preferred/complete/grounded/ideal/ stable (respectively).

We have already illustrated this correspondence when we have exemplified the two views separately with Examples 5.1, 5.2, 5.3. This correspondence is significant when attempting to determine whether a given claim is supported by an argument in a 'winning' set (termed 'sentence-level view' below), in that it allows to focus computation on determining 'winning' sets of assumptions, interleaving the construction of relevant arguments and attacks rather than unnecessarily generating arguments and attacks a-priori.

Sentence-level view. Given an 'acceptable'/'winning' set of argument, a sentence can be deemed 'acceptable'/'winning' if it is the claim of an argument in the set. Equivalently, given an 'acceptable'/'winning' set of assumptions, a sentence can be deemed 'acceptable'/'winning' if it is the claim of an argument supported by a set of assumptions contained in the set. An 'acceptable'/'winning' sentence may be a belief held by an agent, an action to be executed by an agent, a joint action by several agents, a decision, etc, depending on the area of application.

Given the arguments and attacks for the ABA frameworks in Examples 3.1 and 5.4, all notions coincide, in that they sanction the same sentences as 'acceptable'/'winning'. Thus, in Example 3.1, r is admissible, grounded etc, whereas p is not. For ABA frameworks where the semantics differ, these notions differ too, by sanctioning different sentences as 'acceptable'/'winning'. As an illustration, in Example 5.3, c_b is preferred but not stable.

Other observations. All the semantics considered earlier, in terms of sets of arguments or sets of assumptions, can be equivalently reformulated in terms of backward-style and forward-style arguments, by virtue of the correspondence results between the various notions of argument given earlier in Section 4.

6. Some examples

Dung et al. (2009) give examples of use of ABA to model reasoning with argument scheme, decision making, dispute resolution and game-theoretic notions. Furthermore, Matt, Toni, and Vaccari (2010), Matt, Toni, Stournaras, and Dimitrelos (2008), Fan and Toni (2013), Fan, Craven, Singer, Toni, and Williams (2013) give several uses of ABA to model decision-making, for various notions of dominant decisions. This section complements these works by illustrating the use of ABA for default reasoning, defeasible reasoning and persuasion. Note that an ABA framework is a (structured) representation of knowledge/information and of a method for reasoning with it. Given some knowledge/information, several alternative ABA frameworks can be chosen to represent and reason with it. Below, we make some specific choices but others may be suitable too.

Default reasoning. One of the most widely studied examples of default reasoning amounts to reasoning with the following information: typically birds fly (they fly by default), but penguins do not, despite being birds. Given this information, do tweety (a penguin) and joe (a bird) fly? Humans can naturally reason with this information, and answer that tweety does not fly whereas joe does. If the additional information that joe is a penguin is given, humans would withdraw the conclusion that it flies and infer that it does not. This type of reasoning cannot be modelled in classical logic, because it trivialises in the presence of inconsistencies (e.g. that tweety flies, being a bird, and does not fly, being a penguin) and since classical logic is monotonic (and thus it does not allow to withdraw conclusions). This problem can be modelled in ABA by using

$$\begin{aligned}
\mathcal{R} = \{& fly(X) \leftarrow bird(X), normal(X),\ \neg fly(X) \leftarrow penguin(X),\\
& bird(X) \leftarrow penguin(X),\ penguin(tweety) \leftarrow,\ bird(joe) \leftarrow\}\\
\mathcal{A} = \{& normal(tweety), normal(joe)\}\\
& \overline{normal(tweety)} = \neg fly(tweety), \quad \overline{normal(joe)} = \neg fly(joe)
\end{aligned}$$

The conclusions $fly(joe)$ and $\neg fly(tweety)$ are 'winning' under all ABA semantics. Indeed, arguments $\{normal(joe)\} \vdash fly(joe)$ and $\{\} \vdash \neg fly(tweety)$ are 'winning'. Argument $\{normal(tweety)\} \vdash fly(tweety)$ is not 'winning', under any semantics, since it is attacked by $\{\} \vdash \neg fly(tweety)$ that cannot be counter-attacked.

Defeasible reasoning. Here we consider a simple example adapted from Dung and Thang (2010), and inspired by a legal setting:

$$\begin{aligned}
\mathcal{R} = \{& innocent(X) \leftarrow notGuilty(X),\\
& killer(oj) \leftarrow DNAshows(oj), DNAshows(X) \supset killer(X),\\
& DNAshows(X) \supset killer(X) \leftarrow DNAfromReliableEvidence(X),\\
& evidenceUnreliable(X) \leftarrow collected(X,Y), racist(Y),\\
& DNAshows(oj) \leftarrow,\ collected(oj, mary) \leftarrow,\ racist(mary) \leftarrow\}\\
\mathcal{A} = \{& notGuilty(oj), DNAfromReliableEvidence(oj)\}\\
& \overline{notGuilty(oj)} = killer(oj),\\
& \overline{DNAfromReliableEvidence(oj)} = evidenceUnreliable(oj)
\end{aligned}$$

The first rule represents the 'presumption of innocence' principle, that everybody should be deemed innocent until and unless proven guilty. This is a defeasible rule, with the defeasibility represented by the assumption $notGuilty(X)$. The second rule is modus ponens for $\supset$, applied to the implication that if DNA tests show that somebody's blood is at the murder scene, then this is evidence for that person to be the killer. The third rule sanctions that this implication is defeasible, represented by the use of the assumption $DNAfromReliableEvidence(X)$. The last four rules represent uncontroversial information, beyond doubt, namely evidence collected by a documented racist is to be deemed unreliable, *mary* is racist and *mary* collected evidence from which DNA was extracted showing that *oj*'s blood was at the murder scene.

Given this ABA framework, argument $\{notGuilty(oj)\} \vdash innocent(oj)$ is attacked by argument $\{DNAfromReliableEvidence(oj)\} \vdash killer(oj)$ which is attacked by argument $\{\} \vdash evidenceUnreliable(oj)$. Thus $innocent(oj)$ is admissible, grounded, etc, and there is a defeasible inference for $innocent(oj)$.

Persuasion. We model in ABA a variant of an example originally proposed in Sartor (1994) and later used in Prakken and Vreeswijk (2002) and Cayrol, Devred, and Lagasquie-Schiex (2006), presenting a dialogue between two agents a and b, trying to persuade one another:

a: The newspapers have no right to publish information *i*.
b: Why?
a: Because *i* is about *x*'s private life and *x* does not agree to *i* being published
b: The information *i* is not private because *x* is a minister and all information concerning ministers is public
a: But *x* is not a minister since he resigned last month

Agent *a* has the last word in the dialogue and persuades agent *b* that the initial claim of the dialogue holds. The knowledge underpinning this dialogue can be represented by an ABA framework with:

$$\begin{aligned}
\mathcal{R} = \{&\neg rightToPublish(n,i,x) \leftarrow aboutPrivateLife(x,i), \neg agree(x,i),\\
&rightToPrivacy(x,i),\\
&public(x,i) \leftarrow minister(x), typicallyPub(x,i),\\
&\neg minister(x) \leftarrow resigned(x), persistsResigned(x),\\
&aboutPrivateLife(x,i) \leftarrow, \ \neg agree(x,i) \leftarrow,\\
&resigned(x) \leftarrow, \ minister(x) \leftarrow\}\\
\mathcal{A} = \{&rightToPrivacy(x), \ typicallyPub(x,i), \ persistsResigned(x)\}\\
&\overline{rightToPrivacy(x,i)} = public(x,i)\\
&\overline{typicallyPub(x,i)} = \neg minister(x)\\
&\overline{persistsResigned(x)} = appointed(x)
\end{aligned}$$

Then, argument $\{rightToPrivacy(x,i)\} \vdash \neg rightToPublish(n,i,x)$ is attacked by argument $\{typicallyPub(x,i)\} \vdash public(x,i)$ which in turn is attacked by argument $\{persistsResigned(x)\} \vdash \neg minister(x)$, and thus $\neg rightToPublish(n,i,x)$ is admissible, grounded, etc, in accordance with the dialogue outcome.

7. Dispute derivations

Several computational techniques and tools have been defined for ABA (Dung et al., 2006, 2007; Toni, 2013), with the following features:

- they can be abstracted away as disputes between a *proponent* and an *opponent*, in a sort of (fictional) zero-sum, two-player game: the proponent aims at proving (constructively) that an initially given *sentence* is 'acceptable'/'winning', the opponent is trying to prevent the proponent from doing so;
- these disputes are defined as *sequences of tuples* (referred to as *dispute derivations*), representing the state of the game while it is being played;
- these disputes interleave the construction of arguments and identification of attacks between them with testing whether the input sentence is 'acceptable'/'winning';
- the rules of the game allow proponent and opponent to perform various kinds of *filtering* during disputes, but different kinds for different argumentation semantics (for determining whether the input sentence is 'acceptable'/'winning');
- the possible outcomes of dispute derivations are as follows:

(a) the input sentence is proven to be 'acceptable'/'winning' (the dispute derivation is *successful*) or is not proven to be so; and
(b) if the dispute derivation is successful, it returns:
  (I) the set of all assumptions supporting the arguments by the proponent (referred to as the *defence set*),
  (II) the set of all assumptions in the support of arguments by the opponent and chosen by the proponent to be counter-attacked (referred to as the *culprits*),
  (III) in the case of the proposal of Toni (2013), the *dialectical tree* of arguments by proponent and opponent.

Various notions of dispute derivations have been proposed for determining whether sentences are 'acceptable'/'winning' according to the admissible (Dung et al., 2006, 2007; Toni, 2013), grounded (Dung et al., 2007; Toni, 2013), and ideal semantics (Dung et al., 2007; Toni, 2013), as well as, by virtue of relationships between semantics, preferred and complete semantics (Toni, 2013). For simplicity, here we focus on the admissible and grounded semantics, and (mostly) ignore the ideal semantics (see Dung et al., 2007; Toni, 2013 for details).

Data structures. The tuples, for all dispute derivations, include components

$\mathcal{P}$ – representing the to-do list for the *proponent* of arguments to be constructed, supporting the input sentence or in defence of arguments for the input sentence or other defending arguments;
$\mathcal{O}$ – representing the to-do list for the *opponent* of arguments to be constructed attacking the arguments constructed by the proponent;
D – the *defence set* constructed so far by the proponent;
C – the *culprits* constructed so far by the proponent.

In addition, the dispute derivations of Toni (2013), include components

Args – the set of all arguments constructed and dealt with so far (defended or attacked by the proponent);
Att – the attacks between arguments in *Args*.

Once successfully terminated, the *Args* and *Att* component of a dispute derivation amount to the dialectical tree (of arguments by proponent and opponent) outcome.

Elements of $\mathcal{P}$, $\mathcal{O}$ and *Args* may be *potential* (rather than *actual*) arguments[7]. Given the ABA framework in Example 3.1, the left-most and middle trees in Figure 2 are potential arguments for p, whereas the right-most tree is an actual argument (namely the argument $\{a\} \vdash p$). Potential arguments may be seen as intermediate steps in the construction of actual arguments or failed attempts to construct actual arguments. For example, the left-most and middle trees above are intermediate steps in the construction of the right-most tree, and, for the ABA framework in Example 3.1, the tree consisting solely of the root s is a potential argument for s, but no actual argument can be obtained from it.

Algorithmic aspects. The flowchart in Figure 3, adapted from Toni (2013), summarises the rules of the game played by proponent and opponent in dispute derivations for grounded and admissible semantics (the grounded version does not use the parts in bold). We show how to

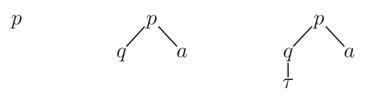
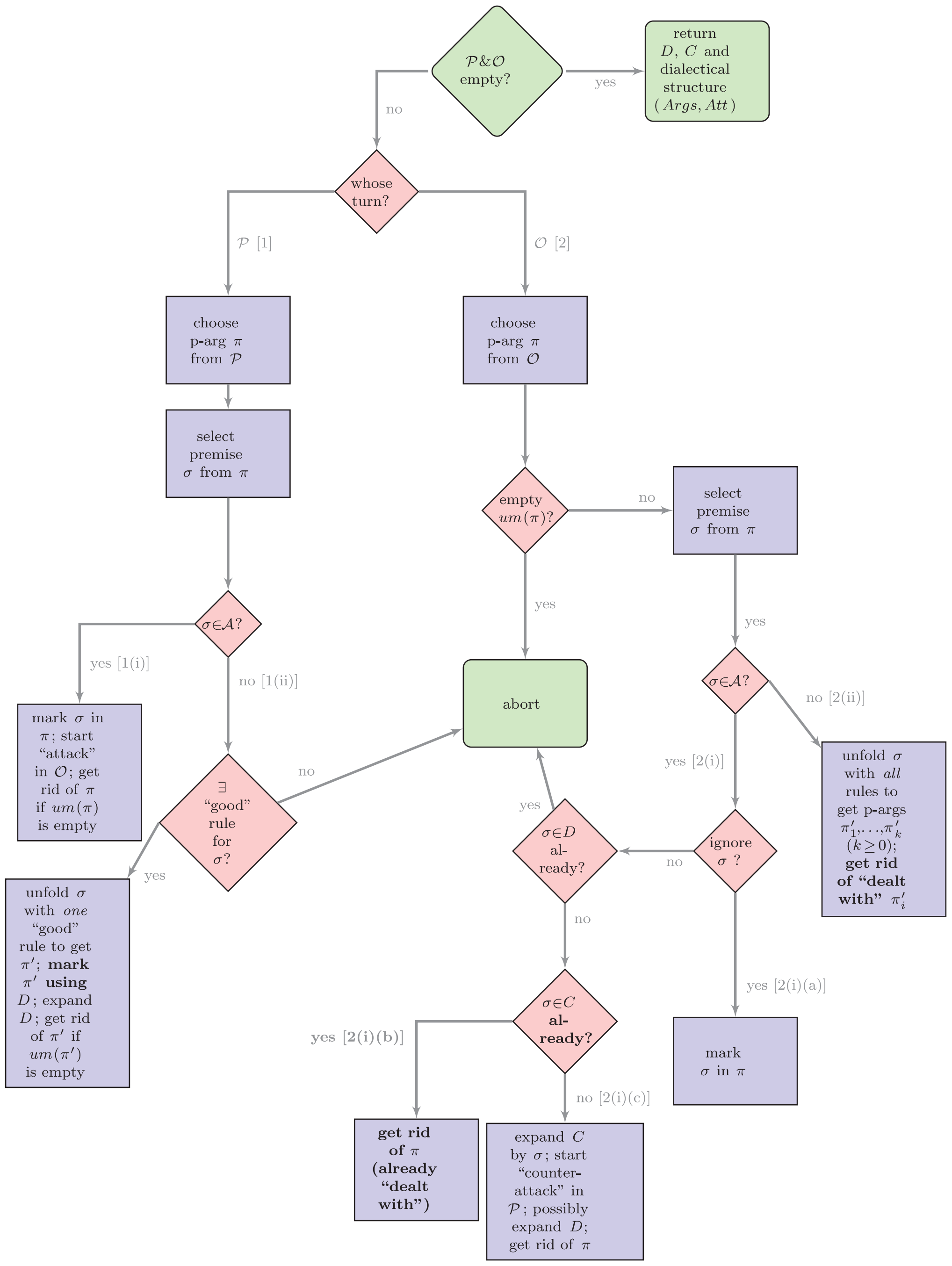


Figure 2. Potential (left and middle) and actual (right) arguments for Example 3.1.



$\mathcal{P}$&$\mathcal{O}$ empty?
yes
return D, C and dialectical structure $(Args, Att)$
no
whose turn?
$\mathcal{P}$ [1]
$\mathcal{O}$ [2]
choose p-arg π from $\mathcal{P}$
choose p-arg π from $\mathcal{O}$
select premise σ from π
empty $um(\pi)$?
no
select premise σ from π
$\sigma \in \mathcal{A}$?
yes [1(i)]
mark σ in π; start "attack" in $\mathcal{O}$; get rid of π if $um(\pi)$ is empty
no [1(ii)]
yes
abort
yes
$\sigma \in \mathcal{A}$?
no [2(ii)]
unfold σ with *all* rules to get p-args $\pi'_1,\ldots,\pi'_k$ $(k \geq 0)$; **get rid of "dealt with"** π'_i
yes [2(i)]
$\exists$ "good" rule for σ?
no
yes
unfold σ with *one* "good" rule to get π'; **mark π' using D**; expand D; get rid of π' if $um(\pi')$ is empty
yes
$\sigma \in D$ already?
no
ignore σ ?
no
yes [2(i)(a)]
mark σ in π
$\sigma \in C$ **already?**
yes [2(i)(b)]
get rid of π (already "dealt with")
no [2(i)(c)]
expand C by σ; start "counter-attack" in $\mathcal{P}$; possibly expand D; get rid of π

Figure 3. Flowchart summarising dispute derivations for admissible and (without the bits in bold) grounded semantics. Labels in square brackets refer to steps of the derivations as defined in Dung et al. (2006, 2007) and Toni (2013) but are solely used here to connect examples to this flowchart. 'p-arg' stands for 'potential argument', and $um(\pi)$ stands for 'unmarked part of p-arg π'. Square boxes without outgoing edges are final steps in one iteration of the algorithm and implicitly link back to the root of the flowchart (rounded diamond), where each iteration starts. Rounded boxes are exit points.

Step	$\mathcal{P}$	$\mathcal{O}$	D	C	Notes
0	$\{\{\} \vdash_{\{d\}} d\}$	$\{\}$	$\{d\}$	$\{\}$	initial set-up
1	$\{\}$	$\{\{\} \vdash_{\{p\}} p\}$	$\{d\}$	$\{\}$	$\mathcal{P}$: selected d, applied 1(i)
2	$\{\}$	$\{\{\} \vdash_{\{q,a\}} p\}$	$\{d\}$	$\{\}$	$\mathcal{O}$: selected p, applied 2(ii)
3	$\{\}$	$\{\{\} \vdash_{\{a\}} p\}$	$\{d\}$	$\{\}$	$\mathcal{O}$: selected q, applied 2(ii)
4	$\{\{\} \vdash_{\{r\}} r\}$	$\{\}$	$\{d\}$	$\{a\}$	$\mathcal{O}$: selected a, applied 2(i)(c)
5	$\{\{\} \vdash_{\{b,c\}} r\}$	$\{\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{P}$: selected r, applied 1(ii)
6	$\{\{b\} \vdash_{\{c\}} r\}$	$\{\{\} \vdash_{\{s\}} s\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{P}$: selected b, applied 1(i)
7	$\{\}$	$\{\{\} \vdash_{\{s\}} s,$ $\{\} \vdash_{\{t\}} t\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{P}$: selected c, applied 1(i)
8	$\{\}$	$\{\{\} \vdash_{\{t\}} t\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected s, applied 2(ii)
9	$\{\}$	$\{\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected t, applied 2(ii)

Figure 4. A successful dispute derivation for Example 5.4.

$$\begin{tikzcd}
\{d\} \vdash_{\{\}} d & {[\mathit{Step}\,1]} & \{d\} \vdash d \\
\{a\} \vdash_{\{\}} p \arrow[u] & {[\mathit{Step}\,4]} & \{a\} \vdash p \arrow[u] \\
\{b,c\} \vdash_{\{\}} r \arrow[u] & {[\mathit{Step}\,7]} & \{b,c\} \vdash r \arrow[u]
\end{tikzcd}$$

Figure 5. Left: the dialectical tree corresponding to the *Args* and *Att* components computed by the dispute derivation of Figure 4: nodes of the tree hold elements of *Args* and an edge $N_1 \to N_2$ in the tree corresponds to an attack in *Att* between the elements of *Args* held at N_1 and N_2. Middle: the step in the derivation when the argument and the attack relationship were added to the dialectical tree. Right: the actual argument view of the dialectical tree.

use this flowchart to generate dispute derivations below. First, however, note that dispute derivations are algorithms, and as such have procedural features such as 'marking' of assumptions that have already been 'processed' in the support of (potential or actual) arguments. We use the notation $A \vdash_S \sigma$ for an argument with claim σ, where $A \subseteq \mathcal{A}$ is 'marked' and $S \subseteq \mathcal{L}$ is not. Thus, in Figure 2, the left-most potential argument is denoted $\{\} \vdash_{\{p\}} p$, the middle potential argument may be denoted $\{\} \vdash_{\{q,a\}} p$ or $\{a\} \vdash_{\{q\}} p$ (depending on whether the assumption a has already been 'processed' or not), and the right-most actual argument ($\{a\} \vdash p$ in the notation of Section 4) may be denoted $\{\} \vdash_{\{a\}} p$ or $\{a\} \vdash_{\{\}} p$ (again, depending on whether the assumption a has already been 'processed' or not). Premises of (actual or potential) arguments are selected from the 'unmarked' part.

Figure 4 gives a dispute derivation for determining whether d is grounded and admissible,[8] given the ABA framework in Example 5.4 (the *Args* and *Att* components for this derivation are given separately in Figure 5). This derivation is obtained as follows, using the flowchart in Figure 3.

At **step 1**, $\mathcal{P}$ is not empty, so the branch labelled [1] is followed in the flowchart. Since $\mathcal{O}$ is empty, it makes sense for $\mathcal{P}$ to play next. There is only one p-argument that can be chosen in $\mathcal{P}$ (i.e. $\{\} \vdash_{\{d\}} d$), and only one 'unmarked' premise in it (i.e. d). This is an assumption, so branch [1(i)] applies, resulting in an attack against the chosen p-argument (on the selected premise) being added to $\mathcal{O}$ (i.e. $\{\} \vdash_{\{p\}} p$, since $\overline{d} = p$). The 'marking' of d in $\{\} \vdash_{\{d\}} d$ results in $\{d\} \vdash_{\{\}} d$, and, since this is an actual argument with empty 'unmarked' part, it is removed from $\mathcal{P}$ and added to the dialectical structure (see Figure 5)[9].

A new iteration then starts (**step 2**). Since $\mathcal{P}$ is empty and $\mathcal{O}$ is non-empty, branch [2] applies, $\{\} \vdash_{\{p\}} p$ is chosen in $\mathcal{O}$ and p is selected in the (non-empty) 'unmarked' part of this p-argument (if no premise were left in the 'unmarked' part then the derivation would be aborted). Since the selected p is not an assumption, branch [2(ii)] applies, giving a new p-argument $\{\} \vdash_{\{q,a\}} p$, obtained by unfolding the chosen argument with the (single) rule in $\mathcal{R}$ usable for unfolding, namely with head p. This new p-argument is added to $\mathcal{O}$. To compute admissibility, the algorithm also checks whether this new p-argument is already 'dealt with' (if the premise of a p-argument in $\mathcal{O}$ is already in C then that p-argument can be deemed to be 'dealt with', as it has already been 'counter-attacked', implicitly). Since none of the assumptions in the support of $\{\} \vdash_{\{q,a\}} p$ is already a culprit in (the empty) C at this stage, this p-argument is not already 'dealt with' and is in $\mathcal{O}$ at the end of step 2.

The next iteration (**step 3**) is analogous to step 2.

At **step 4**, the selected premise a is an assumption, thus branch [2(i)] applies. Here the algorithm allows a non-deterministic choice: whether to ignore the assumption or not. Once an assumption is ignored, it is 'marked' and can no longer be selected. Ignoring an assumption amounts to deciding not to pick it as a culprit in the chosen opponent argument, and is important to guarantee completeness of dispute derivations (namely that one can be generated if the input is 'acceptable', see Dung et al., 2006). In our example derivation, a is not ignored and, since it is not already a defence (in D) or, for admissibility, a culprit (in C), branch [2(i)(c)] applies, a is added to C and a 'counter-attack' against $\{a\} \vdash_{\{\}} p$ is started in $\mathcal{P}$. The (actual) argument $\{a\} \vdash_{\{\}} p$ is dropped from $\mathcal{O}$ and, together with an attack from it to $\{d\} \vdash_{\{\}} d$, is added to the dialectical structure (see Figure 5).

Step 5 applies branch [1(ii)] for $\sigma = r$. Here, a 'good' rule is a rule with head r and not containing any culprits in its body (if no such 'good' rule exists then the derivation is aborted). The choice of one such rule is non-deterministic, and there may be several choices possible in some cases (but not in this example). The chosen rule is $r \leftarrow b, c$, used to unfold $\{\} \vdash_{\{r\}} r$ to give the new p-argument $\{\} \vdash_{\{b,c\}} r$. All assumptions in the premises of this newly generated p-argument are used to expand D. In general, for admissibility, all amongst these assumptions already in D would be put in the 'marked', rather than 'unmarked', support of the newly generated p-argument. Since $D \cap \{b, c\} = \{\}$, no 'marking' needs to be performed in our example. The p-argument $\{\} \vdash_{\{b,c\}} r$ has 'unmarked' elements, so it is in $\mathcal{P}$ at the end of step 5.

The remaining steps are similar to earlier steps, except that, at **step 8** and **step 9**, p-arguments 'disappear' from $\mathcal{O}$ since there are no rules in $\mathcal{R}$ for s and t (when applying branch [2(ii)], namely $k = 0$ in the flowchart). The derivation is successful and computes defence set $\{d, b, c\}$ (which is admissible and grounded, as seen in Example 5.4), culprits $\{a\}$ and the dialectical tree of Figure 5.

When derivations are aborted, this does not mean that no successful derivation for the same input sentence exists, but solely that the non-deterministic choices made in that derivation did not lead to success.

Alternative derivations, success and correctness. There are several other successful dispute derivations for d in Example 5.4, computing the same defence set and culprits but potentially different dialectical structures. For example, consider steps 0-2, 4-9 in Figure 4. They give another successful derivation for d, but with dialectical structure

$$\begin{tikzcd}
\{d\} \vdash_{\{\}} d \\
\{a\} \vdash_{\{q\}} p \arrow[u] \\
\{b, c\} \vdash_{\{\}} r \arrow[u]
\end{tikzcd}$$

where $\{a\} \vdash_{\{q\}} p$ is a potential, non-actual argument. This alternative dispute derivation is obtained by selecting a rather than q in the 'unmarked' support of the opponent's argument. In general, dispute derivations are defined w.r.t. a selection operator given up-front, and this determines how the derivation progresses. The alternative derivation here uses a *non-patient* selection operator, selecting eagerly assumptions in the 'unmarked' support of arguments. This alternative derivation is shorter than the initial derivation, and shows the benefits, in some cases, of using this type of selection operator.

A different dispute derivation would have been obtained if the selection operator had picked c rather than b at step 6. This operator would still be patient. A further alternative dispute derivation for d, w.r.t. the same (patient) selection operator of Figure 4, is obtained by selecting s in $\mathcal{O}$ after step 6, leading to

$7'$	$\{\{b\} \vdash_{\{c\}} r\}$	$\{\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected s, applied 2(ii)
$8'$	$\{\}$	$\{\{\} \vdash_{\{t\}} t\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{P}$: selected c, applied 1(i)
9	$\{\}$	$\{\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected t, applied 2(ii)

A further alternative derivation can be obtained by selecting t, rather than s, in $\mathcal{O}$ after step 7 in the original derivation, leading to

$8'$	$\{\}$	$\{\{\} \vdash_{\{s\}} s\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected t, applied 2(ii)
$9'$	$\{\}$	$\{\}$	$\{d,b,c\}$	$\{a\}$	$\mathcal{O}$: selected s, applied 2(ii)

The choice of opponent/proponent, the choice of argument in proponent/opponent and the selection operator are all parameters of dispute derivations (Toni, 2013), decided up-front before the derivations are started, and determining which alternative derivations are obtained. However, the choice of these parameters does not affect soundness, namely for all possible choices the computed defence set is 'winning' (namely dispute derivations are *sound*). Moreover, for a large class of ABA frameworks (namely *p-acyclic* ones (Dung et al., 2007; Toni, 2013)) the existence of derivations computing 'winning' defence sets for 'winning' sentences is guaranteed (namely dispute derivations are *complete*, for p-acyclic ABA frameworks). Note that the choice of parameters may affect speed and computation time, and parallel programming techniques can be used beneficially to explore different choices of parameters concurrently (see Craven, Toni, Hadad, Cadar, & Williams, 2012).

In general, also, alternative dispute derivations may be obtained depending on the available rules in $\mathcal{R}$ (as there is a non-deterministic choice point at branch [1(ii)]), and depending on whether assumptions in the opponent's arguments are ignored or not (again a non-deterministic choice, see branch [2(i)(a)]). For example, if an additional rule for r were in $\mathcal{R}$, then at step 5 a different (potential) argument may have been obtained in $\mathcal{P}$, possibly giving rise to a different successful derivation. Moreover, if $q \leftarrow c$ were in $\mathcal{R}$ rather than $q \leftarrow$, then at step 3 $\{\} \vdash_{\{a,c\}} p$ would be obtained in $\mathcal{O}$ and whichever of a and c, if selected, could be ignored. Some of these non-deterministic choices may give rise to a successful derivation and some may not. For example, ignoring a in $\{\} \vdash_{\{a,c\}} p$ would not give a successful dispute derivation (as c cannot be a culprit, as it cannot be counter-attacked, since there is no argument against it).

Filtering. Dispute derivations incorporate various kinds of filtering, some common to all types of dispute derivations, some different for computing different semantics. We illustrate the common forms of filtering for the ABA framework in Example 5.4, but with $\mathcal{R}$ extended with rules $s \leftarrow a,b,c$ and $t \leftarrow d,p$. Consider input sentence s: obviously, this is neither admissible nor grounded. Consider the following attempt to construct a dispute derivation for this sentence (again, ignoring the *Args* and *Att* components):

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
0	$\{\{\} \vdash_{\{s\}} s\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{\{\} \vdash_{\{a,b,c\}} s\}$	$\{\}$	$\{a,b,c\}$	$\{\}$
2	$\{\{a\} \vdash_{\{b,c\}} s\}$	$\{\{\} \vdash_{\{r\}} r\}$	$\{a,b,c\}$	$\{\}$
3	$\{\{a\} \vdash_{\{b,c\}} s\}$	$\{\{b,c\} \vdash_{\{\}} r\}$	$\{a,b,c\}$	$\{\}$

At this point, if it is the opponent's turn, then whichever possible culprit is selected in $\mathcal{O}$ (by applying [2(i)]), this is already a defence and the derivation is aborted. This form of filtering is referred to as *filtering of culprits by defences*.

Consider now input sentence t, again neither admissible nor grounded, and the following attempt to construct a dispute derivation for this sentence:

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
0	$\{\{\} \vdash_{\{t\}} t\}$	$\{\}$	$\{\}$	$\{\}$
1	$\{\{\} \vdash_{\{d,p\}} t\}$	$\{\}$	$\{d\}$	$\{\}$
2	$\{\{d\} \vdash_{\{p\}} t\}$	$\{\{\} \vdash_{\{p\}} p\}$	$\{d\}$	$\{\}$
3	$\{\{d\} \vdash_{\{p\}} t\}$	$\{\{\} \vdash_{\{q,a\}} p\}$	$\{d\}$	$\{\}$
4	$\{\{d\} \vdash_{\{p\}} t, \{\} \vdash_{\{r\}} r\}$	$\{\{a\} \vdash_{\{q\}} p\}$	$\{d\}$	$\{a\}$

At this point, if it is the proponent's turn and the first potential argument $\{d\} \vdash_{\{t\}} p$ in $\mathcal{P}$ is chosen, p is necessarily selected and [1(ii)] applies: there is only a candidate rule for p, i.e. $p \leftarrow q, a$, and this is not 'good', since a is already a culprit and cannot also be added to the defence set. Thus the derivation is aborted. This form of filtering is referred to as *filtering of defences by culprits*.

Both these forms of filtering guarantee that the computed defence set does not attack itself, an essential condition for admissibility (and groundedness). E.g., in the case of input sentence s, $\{a, b, c\}$ attacks itself (as $\{b, c\} \vdash r$ attacks $\{a\}$), and, in the case of input sentence t, $\{a, d\}$ attacks itself (as $\{a\} \vdash p$ attacks $\{d\}$).

We illustrate the third kind of filtering, used for admissible (and ideal) input sentences, in the context of Example 5.1. The following is a successful dispute derivation for input sentence b (this is admissible but not grounded):

Step	$\mathcal{P}$	$\mathcal{O}$	D	C
0	$\{\{\} \vdash_{\{b\}} b\}$	$\{\}$	$\{b\}$	$\{\}$
1	$\{\}$	$\{\{\} \vdash_{\{c_b\}} c_b\}$	$\{b\}$	$\{\}$
2	$\{\}$	$\{\{\} \vdash_{\{a\}} c_b\}$	$\{b\}$	$\{\}$
3	$\{\{\} \vdash_{\{c_a\}} c_a\}$	$\{\}$	$\{b\}$	$\{a\}$
4	$\{\}$	$\{\}$	$\{b\}$	$\{a\}$

At step 4, applying [1(ii)], *filtering of defences by defences* is applied to avoid generating the potential argument $\{\} \vdash_{\{b\}} c_a$, whose ('unmarked') support is already a subset of the defence set. This form of filtering is not applied for the grounded semantics (and indeed b is not grounded). It is used for the admissible (and ideal) semantics in order to successfully terminate in the presence of loops (as in this example) and/or to avoid recomputation.

The last form of filtering, *filtering of culprits by culprits*, is similarly applied, for the admissible (and ideal) semantics, to successfully terminate and/or to avoid recomputation.

The latter two forms of filtering (of defences by defences and of culprits by culprits) are performed by the bits in bold in the flowchart in Figure 3.

Systems. Several systems are available for computing successful dispute derivation for admissible and grounded sentences, namely CaSAPI,[10] proxdd[11] and grapharg.[12] These are implementations of the dispute derivations of Gaertner and Toni (2007), Toni (2013), and Craven, Toni, and Williams (2013), respectively. The latter incorporates a form of minimality of arguments. All systems are in Prolog. They all take, as input, an ABA framework and a sentence, whose 'acceptability' is under investigation, and output, when possible, a dispute derivation and its outcome.

8. Knowledge representation and reasoning: FAQs

In this section we summarise answers to some most FAQs on how to use ABA for knowledge representation and reasoning purposes. We organise the FAQs and answers in three groups, respectively, concerning (1) the format of ABA frameworks, (2) the types of arguments and attacks available in ABA, and (3) the semantics or computational mechanisms for ABA frameworks.

8.1. *Format of ABA frameworks*

Q: How can I guarantee that $\mathcal{A} \neq \{\}$? A: Add a bogus assumption and its contrary, in total two new sentences, to the given language. However, argumentation in general (and ABA in particular) is probably not what you need if you cannot find any assumptions naturally in your domain. For example, if you want to represent and reason with non-defeasible information, e.g. that humans are mortal and Socrates, being human, is mortal, standard classical logic would suffice (as no possibility of debate exists as to the mortality of Socrates). If you want to see this (unquestionable) information as an instance of ABA nonetheless, you may choose, for example, $\mathcal{R} = \{mortal(s) \leftarrow human(s), \quad human(s) \leftarrow\}$, $\mathcal{A} = \{bogus\}$, $\overline{bogus} = c_{bogus}$, with $\mathcal{L} = \{mortal(s), human(s), bogus, c_{bogus}\}$.

Q: How can I guarantee that each assumption has a contrary? A: If an assumption has no contrary and you cannot find any sentence in the given language that can be set to be its contrary then add a new sentence to the given language and set it to be the contrary of the assumption. This contrary will serve as a handle for disagreement with the assumption, but no attack can be generated against the assumption if its contrary is a new sentence. However, a number of such attacks can be obtained should the set of rules be extended with rules for the new sentence/contrary. For example, if you believe that Socrates's mortality should be questionable on grounds of super powers being ascribable to him, you may choose an ABA framework with $\mathcal{R} = \{mortal(s) \leftarrow human(s), \neg super(s), \quad human(s) \leftarrow\}$, $\mathcal{A} = \{\neg super(s)\}$, $\overline{\neg super(s)} = super(s)$, where $super(s)$ is included in the language solely to make sure that the assumption has a contrary.

Q: It seems unnatural that each assumption should have exactly a contrary, e.g. the contrary of assumption *sunny* may be either *cloudy* or *rainy*, and the contrary of *comfy* (e.g. for a dress) may be *tight* and *rigid*. A: These more general forms of contraries can be naturally encoded in ABA using a single contrary per assumption. For example, you can set $\overline{sunny} = not\_sunny$ with $not\_sunny \leftarrow cloudy$ and $not\_sunny \leftarrow rainy$ in $\mathcal{R}$, and $\overline{comfy} = not\_comfy$ with $not\_comfy \leftarrow tight, rigid$ in $\mathcal{R}$. Thus, imposing that each assumption has a single contrary does not cause a loss of generality.

Q: Can sentences be repeated in the body of rules? A: Yes, since semantically it makes no difference. Computationally however it may make a difference, e.g., consider $\mathcal{R} = \{p \leftarrow q, q, \quad q \leftarrow a, \quad q \leftarrow b, \ldots\}$ with $a, b \in \mathcal{A}$. Then a standard dispute derivation for p may add to the defence set both a and b, and have to defend both unnecessarily (note however that a dispute derivation of Craven et al. (2013) would introduce only one of a and b into the defence set). In any case, without loss of generality, it is possible to impose that no repetitions occur in rules.

Q: Can sentences in $\mathcal{L}$ have classical connectives, e.g. implication? A: Yes, we have discussed this issue in Section 3 and seen an example in Section 6, making use of classical implication $\supset$. This issue is further discussed in Dung et al. (2006). The designer of ABA frameworks needs to decide whether to include connectives within the (object-level) language or use the (meta-level) features of ABA. For example, should $\mathcal{R}$ include rules $p \leftarrow q$, alongside $q \leftarrow$, or rules (with object-level implication) $p \leftarrow q \supset p, q$, alongside $q \supset p \leftarrow$ and $q \leftarrow$? In both cases an argument $\{\} \vdash p$ is obtained. This is a knowledge representation choice and challenge, afforded by the generality and abstract nature of ABA.

Q: The only natural form of contrary seems to be (classical) negation, why are you not equating contrary and negation? A: We have discussed this in Section 3. A further argument in favour of keeping the distinction is that otherwise ABA would not be as general as required to capture some non-monotonic reasoning formalisms. For example, default logic (Reiter, 1980) is captured in ABA by using assumptions $M\sigma$, where σ is a sentence in classical logic, with contrary $\neg\sigma$, the classical negation of σ but not of $M\sigma$.

Q: How do you decide what is an assumption and what is a 'fact' (namely a rule with an empty body)? A: An assumption is defeasible, whereas a fact is not. So anything that is not an opinion and not arguable needs to be written as a fact, and anything that may be argued against as an assumption.

Q: What if I need to reason with a non-flat ABA framework? A: The semantics needs to be generalised – actually, it was defined in its general format to start with, see (Bondarenko et al., 1997). We have presented here a simplification, sufficient for flat-frameworks. See also Section 8.3 for an outline of the fully general semantics. In addition, the notion of dispute derivation would need to be generalised to match the generalised semantics.

Q: Where do I get ABA frameworks from? How do I know that an ABA framework is correctly representing a problem? A: This is the well-known knowledge acquisition bottleneck problem of knowledge representation. In some cases, ABA frameworks can be automatically extracted from other data, e.g. to support medical decision-making from a tabular representation of clinical trials (Fan et al., 2013). Otherwise, they need to be created by hand from suitable (human) knowledge, and several alternative representations may work. In order to make sure that a representation is adequate, unit testing as in software engineering can be deployed.

Q: Why do ABA frameworks disallow an explicit knowledge base/theory in addition to (inference) rules? A: Originally (see Bondarenko et al., 1997) ABA frameworks included a fifth component – indeed a knowledge base/theory – but this is not needed as this extra component can be represented, without loss of generality, as a set of inference rules with an empty body (facts). Overall, rules in ABA may be domain-specific (from a theory) as well as domain-independent.
Q: Some problems are naturally represented in terms of two kinds of rules: strict and defeasible, whereas ABA only allows one type of rules in $\mathcal{R}$, why? A: ABA has one single form of defeasibility: the assumptions. Indeed assumptions can be seen as defeasible rules with an empty body. All other rules 'inherit' defeasibility from assumptions: if a rule has one or more assumptions in the body, then it is defeasible, if it has no assumptions it is strict. This issue is further discussed in Dung et al. (2009). Moreover, Toni (2008) gives a method for representing strict and defeasible rules in ABA in such a way as to fulfil rationality postulates for defeasible reasoning (Caminada & Amgoud, 2007).

Q: Some problems are naturally represented in terms of preferences over rules: why are preferences not allowed in ABA? A: Some forms of reasoning in the presence of preferences can be naturally encoded in ABA, e.g. following the last link principle (Kowalski & Toni, 1996; Toni, 2008), or Brewka's approach (Brewka, 1989) to incorporating preferences into default reasoning (Thang & Luong, 2013), or decision-making in the presence of preferences over goals (Fan et al., 2013). Rather than extending ABA frameworks and modifying the semantics, the approach followed is to provide a mapping from different forms of reasoning with preferences onto standard ABA without preferences, in the spirit of keeping the underlying argumentation framework as simple as possible while supporting rich forms of reasoning. For example, a preference of defeasible rule that 'penguins don't fly' over defeasible rule that 'birds fly' may be represented using the ABA framework with $\mathcal{R}$ including $fly(X) \leftarrow bird(X), asm_f(X)$ and $\neg fly(X) \leftarrow penguin(X), asm_{\neg f}(X)$, $\mathcal{A}$ including (all instances over some vocabulary of) $asm_f(X)$ and $asm_{\neg f}(X)$ and $\overline{asm_f(X)} = \neg fly(X)$.

Q: ABA is an instance of AA (Dung, 1995), why not represent problems directly in terms of abstract arguments and attacks? A: AA focuses on resolving (via suitable notions of argumentation semantics) conflicts arising amongst arguments. However, it does not address the issue as to where arguments and attacks are coming from. ABA provides a finer-grained level of representation addressing also the issue of constructing arguments and attacks, while at the same time resolving conflicts between them, in a way that fully matches AA semantics (Dung et al., 2007; Toni, 2012). In principle, it would be possible to extract an AA framework from an ABA framework (e.g. as we have done at the beginning of Section 4 for Example 3.1) and then operate at the abstract level, but this would have very severe computational implications in general, typically giving rise to very large sets of arguments and attacks unnecessarily if the aim is to determine 'acceptability' of sentences that may only require considering few such arguments and attacks. Moreover, it would prevent the computational advantages afforded by filtering in dispute derivations, as this takes advantage of the structure of arguments while constructing them. Dispute derivations can be seen as generating only (abstract) arguments and attacks 'relevant' to the given input sentences.

Note that not only is ABA an instance of AA but also AA is an instance of ABA (Toni, 2012) and each AA framework can be equivalently written as an ABA framework. For instance, the ABA framework of Example 5.1 is the ABA rewriting of the AA framework with arguments $\{a, b\}$ and attacks $\{(a, a), (b, a), (a, b)\}$. Thus, dispute derivations can be deployed to determine acceptable arguments in AA.

8.2. *Arguments and attacks*

Q: Most approaches to structured argumentation impose consistency of the support of arguments, why not ABA? A: Consistency of the support of arguments is costly to check/ensure, as discussed in Dung et al. (2010), and ABA has a strong focus on computation and the viable support of applications. Also, since ABA does not impose that the underlying language $\mathcal{L}$ is closed under (classical) negation, imposing consistency would not be natural in ABA. ABA imposes instead that, under all ABA semantics, the set of all 'winning' arguments does not attack itself, in turn implying that no such argument has a support that attacks itself. In some special cases of ABA instances where $\mathcal{L}$ is closed under negation and the notion of contrary coincides with negation, this implies consistency of the support of all 'winning' arguments (Toni, 2008).

Q: Most approaches to structured argumentation impose minimality of the support of arguments, why not ABA? A: Minimality is imposed in some forms of argumentation to ensure relevance of arguments w.r.t. their claim. Our way to impose relevance in ABA is by defining arguments as trees, as discussed in Dung et al. (2010). Different forms of minimality can be imposed on ABA arguments, at different computational expenses, as discussed in Craven et al. (2013).

Q: Many approaches to structured argumentation allow different types of attacks, e.g. undercutting and rebuttal attacks, why not ABA? A: ABA only allows attacks against the support of arguments. Other forms of attacks, e.g. on the claim of arguments (rebuttal) or on the application of a rule, can be obtained in ABA by suitable mappings, as discussed in Kowalski and Toni (1996) and Dung et al. (2006, 2009). Again, similarly to preferences as discussed in Section 8.1, the approach followed is to provide a mapping from different forms of attacks onto standard ABA with attacks on assumptions, keeping the framework as simple as possible while still allowing rich forms of reasoning. For example, in the well-known Nixon diamond scenario, the mutual attack between argument 'Nixon is a pacifist, being a quaker' and argument 'Nixon is not a pacifist, being a republican', on their claim, is reduced to the mutual attack between arguments $\mathit{normal\_quaker}(n) \vdash \mathit{pacifist}(n)$ and $\mathit{normal\_republican}(n) \vdash \neg \mathit{pacifist}(n)$ obtained from the ABA framework with $\mathcal{R}$ with rules

$$\begin{aligned} &\mathit{pacifist}(X) \leftarrow \mathit{quaker}(X), \mathit{normal\_quaker}(X) \\ &\neg \mathit{pacifist}(X) \leftarrow \mathit{republican}(X), \mathit{normal\_republican}(X) \\ &\mathit{quaker}(n) \leftarrow \mathit{republican}(n) \leftarrow \end{aligned}$$

assumptions including $\mathit{normal\_quaker}(n)$ and $\mathit{normal\_republican}(n)$ and $\overline{\mathit{normal\_quaker}(n)} = \neg \mathit{pacifist}(n)$, $\overline{\mathit{normal\_republican}(n)} = \mathit{pacifist}(n)$.

Q: Why ABA arguments are solely supported by assumptions, and not rules? A: ABA arguments are supported by rules too, as we have seen in Section 3, but for simplicity of presentation they are mostly omitted from shorthands for arguments, especially when focusing on computation. Indeed, for all forms of dispute derivations except the ones in Craven et al. (2013) rules play no role in the construction of 'winning' sets of arguments and can be omitted. However, in some applications of ABA, e.g. for providing justifications for literals in or out of answer sets in Answer Set Programming (Gelfond, 2007), it is useful to single out facts used to construct arguments (Schulz & Toni, 2013), and a different shorthand for ABA arguments is used.

8.3. *Semantics/computation*

Q: Several semantics for ABA are available, which one should I use? A: This very much depends on the application/problem at hand, as discussed also in Section 5. Some semantics are more *credulous* (by sanctioning several sets of arguments as 'winning') whereas others are *sceptical*, to a greater or lesser degree (by sanctioning only one set of arguments as 'winning', potentially the empty set). In a decision-making setting, a credulous semantics may be more suitable when a decision is required under all circumstances, unless the decision is too critical and it is better to be sceptical and go only for an uncontroversial decision, if any, or none, else.

Q: How do I know that a dispute derivation actually computes what it is supposed to, and it terminates? A: Dispute derivations are proven to be sound (compute correctly 'winning' arguments for input sentences) and, for a large class of ABA frameworks, complete (namely they finitely compute 'winning' arguments for a given input sentence if some such argument exists). Details on these results for several semantics (notions of 'winning' arguments) as well as details on restrictions on ABA frameworks for guaranteeing completeness can be found in Dung et al. (2007) and Toni (2013).

Q: I have chosen an ABA semantics suitable for my knowledge representation and reasoning needs, but this is not one amongst admissible/grounded/ideal semantics. Is there a computational mechanism that I can use for determining 'winning' sentences under this other semantics? A: Some theoretical results on the correspondence between semantics can be used to extend existing soundness and completeness results to other semantics. For example, since the grounded set of assumptions/arguments is contained in all 'winning' sets of assumptions/arguments, under any ABA semantics, dispute derivations that are sound for the grounded semantics are guaranteed to be sound for all other semantics. Toni (2013) shows which mechanisms are guaranteed to be sound and complete, and under which conditions, for semantics other than admissible/grounded/ideal semantics.

Q: Dispute derivations are query-oriented, focusing on determining whether a given input sentence is 'winning'. What if I need to compute full 'winning' extensions (i.e. sets of assumptions/arguments) for finding solutions for a problem represented in ABA? A: No mechanisms for computing full 'winning' extensions are currently available in ABA. We have found that, for all applications we have considered, one can typically identify a handful of sentences to be checked, and full extensions are not required. For example, in the medical application of Fan et al. (2013), the input sentences amount to the available clinical trials, and, in e-procurement (Matt et al., 2008), the input sentences amount to the available providers.

Q: How can I reason with non-flat ABA frameworks? A: Sets of assumptions (and the support of arguments) need to be *closed*, i.e. contain all assumptions that can be deduced from them (see Bondarenko et al., 1997). Consider the ABA framework with:

$$\mathcal{R} = \{x \leftarrow c,\ z \leftarrow b,\ a \leftarrow b\}$$

$$\mathcal{A} = \{a, b, c\}$$

$$\overline{a} = x, \quad \overline{b} = y, \quad \overline{c} = z$$

and sentence c. This is attacked by $\{b\}$, which cannot be counter-attacked alone. However, a can be deduced from $\{b\}$, and the closed attack $\{a, b\}$ is counter-attacked by $\{c\}$. Thus, c is admissible, correctly. Similarly, $\{b\}$ is not admissible, because it is not closed. The closed $\{a, b\}$ is admissible because it does not attack itself and $\{b\}$ counter-attacks the closed $\{c\}$ which attacks $\{a, b\}$.

9. Conclusion

We have provided a brief introduction to several essential features of ABA, a form of structured argumentation that has found several applications in practice (and whose development has been driven by applications) and is supported by solid theoretical foundations as well as usable computational mechanisms and systems.

For lack of space, we have ignored some important aspects, including computational complexity and the deployment of ABA in multi-agent settings: details can be found in Dimopoulos et al. (2002) and Dunne (2009) for the former and in Fan and Toni (2011, 2012) for the latter.

ABA is still a very active research area with several avenues for future work, including: the definition of failed dispute derivations and variants of dispute derivations *unconditionally* complete w.r.t. several semantics (i.e. complete also for non-p-acyclic ABA frameworks); further applications, e.g. in medical and legal settings; more efficient implementations, to better support applications, possibly using parallel programming techniques along the lines of Craven et al. (2012); modelling of other forms of preferences, if demanded by applications, e.g. building up and extending the work of Fan et al. (2013); the integration of ontologies within ABA (see a discussion of this aspect in Toni (2012)); the extension of ABA with probabilistic reasoning, e.g. following Dung and Thang (2010).

Acknowledgements

I am grateful to several colleagues for useful feedback and suggestions on earlier versions of this paper, and in particular to Tony Hunter, Claudia Schultz, Tony Kakas, Henry Prakken, Xiuyi Fan, Phan Minh Dung, and Bob Kowalski.

Notes

1. $\tau \notin \mathcal{L}$ represents 'true' and stands for the empty body of rules. In other words, each rule $\sigma_0 \leftarrow$ can be interpreted as $\sigma_0 \leftarrow \tau$ for the purpose of presenting deductions as trees.
2. We use here the following notions, for sets of arguments $\mathtt{A}$, $\mathtt{A}'$ and arguments α, α':
   (i) $\mathtt{A}$ attacks $\mathtt{A}'$ iff there exist $\alpha \in \mathtt{A}$ and $\alpha' \in \mathtt{A}'$ such that α attacks α';
   (ii) α attacks $\mathtt{A}'$ iff $\{\alpha\}$ attacks $\mathtt{A}'$;
   (iii) $\mathtt{A}$ attacks α' iff $\mathtt{A}$ attacks $\{\alpha'\}$.
3. A set S is maximally (w.r.t. $\subseteq$) fulfilling property p iff there is no $S' \supset S$ such that S' fulfils property p.
4. A set S is minimally (w.r.t. $\subseteq$) fulfilling property p iff there is no $S' \subset S$ such that S' fulfils property p.
5. A set of assumptions A attacks an assumption a iff A attacks $\{a\}$.
6. In the context of the realistic reading of the ABA framework in Example 3.1, here d may stand for 'sad'.
7. The arguments in $\mathcal{P}$, $\mathcal{O}$ are not explicit in the dispute derivations of Dung et al. (2006, 2007) and are instead flattened out to a set of sentences (in $\mathcal{P}$) and sets of sets of sentences (in $\mathcal{O}$). We focus our discussion here on the dispute derivations of Toni (2013).
8. The derivation is the same for both semantics, as the bits of the algorithm in bold make no difference in this example, as we will see.
9. The flowchart focuses on changes to the $\mathcal{P}$, $\mathcal{O}$, D and C components only and ignores changes to the dialectical structure – for simplicity and compactness of presentation.
10. www.doc.ic.ac.uk/~ft/CaSAPI/, no longer maintained.
11. www.doc.ic.ac.uk/~rac101/proarg/.
12. www.doc.ic.ac.uk/~rac101/proarg/.

References

Amgoud, L., & Prade, H. (2009). Using arguments for making and explaining decisions. *Artificial Intelligence*, *173*, 413–436.

Besnard, P., & Hunter, A. (2008). *Elements of argumentation*. Cambridge, MA: MIT Press.

Bondarenko, A., Dung, P.M., Kowalski, R.A., & Toni, F. (1997). An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, *93*, 63–101.

Bondarenko, A., Toni, F., & Kowalski, R.A. (1993). An assumption-based framework for non-monotonic reasoning. In L.M. Pereira & A. Nerode (Eds.), *Proceedings of the 2nd international workshop on logic programming and non-monotonic reasoning (LPNMR 1993)*, June (pp. 171–189). Lisbon, Portugal: MIT Press.

Brewka, G. (1989). Preferred subtheories: An extended logical framework for default reasoning. In *Proc. IJCAI* (pp. 1043–1048), San Francisco, CA: Morgan Kaufmann.

Caminada, M., & Amgoud, L. (2007). On the evaluation of argumentation formalisms. *Artificial Intelligence*, *171*, 286–310.

Cayrol, C., Devred, C., & Lagasquie-Schiex, M.C. (2006). Handling controversial arguments in bipolar argumentation systems. In P.E. Dunne & T.J.M. Bench-Capon (Eds.), *Computational models of argument: Proceedings of COMMA 2006, September 11–12, 2006, Liverpool, UK*, Volume 144, *Frontiers in artificial intelligence and applications* (pp. 261–272). Amsterdam, The Netherlands: IOS Press.

Craven, R., Toni, F., Hadad, A., Cadar, C., & Williams, M. (2012). Efficient support for medical argumentation. In G. Brewka, T. Eiter & S.A. McIlraith (Eds.), *Proc. 13th international conference on principles of knowledge representation and reasoning* (pp. 598–602). Palo Alto, CA: AAAI Press.

Craven, R., Toni, F., & Williams, M. (2013). Graph-based dispute derivations in assumption-based argumentation. In E. Black, S. Modgil, & N. Oren (Eds.), *Theories and applications of formal argumentation – second international workshop, TAFA 2013*, Lecture Notes in Artificial Intelligence, Springer, to appear.

Dimopoulos, Y., Nebel, B., & Toni, F. (2002). On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence*, *141*, 57–78.

Dung, P.M. (1991). Negations as hypotheses: An abductive foundation for logic programming. In *Proceedings of the international conference in logic programming*, pp. 3–17, Cambridge, MA: MIT Press.

Dung, P.M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, *77*, 321–358.

Dung, P.M., Kowalski, R.A., & Toni, F. (2006). Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, *170*, 114–159.

Dung, P.M., Kowalski, R.A., & Toni, F. (2009). Assumption-based argumentation. In I. Rahwan and G. Simari (Eds.), *Argumentation in AI* (pp. 199–218). Berlin, Germany: Springer.

Dung, P.M., Mancarella, P., & Toni, F. (2007). Computing ideal sceptical argumentation. *Artificial Intelligence, Special Issue on Argumentation in Artificial Intelligence*, *171*, 642–674.

Dung, P.M., & Thang, P.M. (2010). Towards (probabilistic) argumentation for jury-based dispute resolution. In P. Baroni, F. Cerutti, M. Giacomin, & G.R. Simari (Eds.), *Computational models of argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, Vol. 216 of *Frontiers in Artificial Intelligence and Applications* (pp. 171–182). Amsterdam, The Netherlands: IOS Press.

Dung, P.M., Toni, F., & Mancarella, P. (2010). Some design guidelines for practical argumentation systems. In P. Baroni, F. Cerutti, M. Giacomin and G. Simari (Eds.), *Proceedings of the Third International Conference on Computational Models of Argument (COMMA'10)* (Vol. 216, pp. 183–194). IOS Press,

Dunne, P.E. (2009). The computational complexity of ideal semantics. *Artificial Intelligence*, *173*, 1559–1591.

Elvang-Gøransson, M., & Hunter, A. (1995). Argumentative logics: Reasoning with classically inconsistent information. *Data & Knowledge Engineering*, *16*, 125–145.

Eshghi, K., & Kowalski, R.A. (1989). Abduction compared with negation by failure. In *ICLP*, pp. 234–254, Cambridge, MA: MIT Press.

Fan, X., Craven, R., Singer, R., Toni, F., & Williams, M. (2013). Assumption-based argumentation for decision-making with preferences: A medical case study. In J. Leite, T.C. Son, P. Torroni, L. van der Torre, & S. Woltran (Eds.), *Computational logic in multi-agent systems – 14th international workshop, CLIMA XIV, Corunna, Spain, September 16–18*. Lecture Notes in Artificial Intelligence, ol. 8143. Berlin, Germany: Springer.

Fan, X., & Toni, F. (2011). Assumption-based argumentation dialogues. In T. Walsh (Ed.), *IJCAI 2011, proceedings of the 22nd international joint conference on artificial Intelligence* (pp. 198–203). Palo Alto, CA: IJCAI/AAAI.

Fan, X., & Toni, F. (2012). Agent strategies for aba-based information-seeking and inquiry dialogues. In L.D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, & P.J.F. Lucas (Eds.), *Proceedings of 20th European conference on artificial intelligence (ECAI 2012)*, Vol. 242 of *Frontiers in artificial intelligence and a Applications* (pp. 324–329). Amsterdam, The Netherlands: IOS Press.

Fan, X., & Toni, F. (2013). Decision making with assumption-based argumentation. In E. Black, S. Modgil, & N. Oren (Eds.), *Theories and applications of formal argumentation – second international workshop, TAFA 2013*, Lecture Notes in Artificial Intelligence, Springer, to appear.

Gaertner, D., & Toni, F. (2007). On computing arguments and attacks in assumption-based argumentation. *IEEE Intelligent Systems, Special Issue on Argumentation Technology*, *22*, 24–33.

García, A.J., & Simari, G.R. (2004). Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, *4*, 95–138.

Gelfond, M. (2007). Answer sets. In F. van Harmelen, V. Lifschitz, & B. Parter (Eds.), *Handbook of knowledge representation* (Chapter 7, pp. 285–316). New York City, USA: Elsevier.

Kakas, A.C., Kowalski, R.A., & Toni, F. (1992). Abductive logic programming. *Journal of Logic and Computation*, *2*, 719–770.

Kakas, A.C., Kowalski, R.A., & Toni, F. (1998). The role of abduction in logic programming. In D.M. Gabbay, C.J. Hogger, & J.A. Robinson (Eds.), *Handbook of logic in artificial intelligence and logic programming* (pp. 235–324). Oxford: Oxford University Press.

Kakas, A.C., Mancarella, P., Sadri, F., Stathis, K., & Toni, F. (2008). Computational logic foundations of KGP agents. *Journal of Artificial Intelligence Research*, *33*, 285–348.

Kakas, A.C., & Toni, F. (1999). Computing negation as failure via argumentation. *Journal of Logic and Computation*, *9*, 515–562.

Kowalski, R.A., & Sadri, F. (1999). From logic programming towards multi-agent systems. *Annals of Mathematics in Artificial Intelligence*, *25*, 391–419.

Kowalski, R.A., & Toni, F. (1996). Abstract argumentation. *Artificial Intelligence and Law*, *4*, 275–296 (also published in 'Logical Models of Argumentation').

Krause, P., Ambler, S., Elvang-Gøransson, M., & Fox, J. (1995). A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, *11*, 113–131.

Krause, P., Ambler, S., & Fox, J. (1992). The development of a 'logic of argumentation'. In B. Bouchon-Meunier, L. Valverde, & R.R. Yager (Eds.), *IPMU '92 – advanced methods in artificial intelligence, proceedings of the 4th international conference on processing and management of uncertainty in knowledge-based systems*, Vol. 682 of *Lecture notes in computer science* (pp. 109–118). Berlin, Germany: Springer.

Lin, F., & Shoham, Y. (1989). Argument systems: A uniform basis for nonmonotonic reasoning. In R.J. Brachman, H.J. Levesque, & R. Reiter (Eds.), *Proceedings of the 1st international conference on principles of knowledge representation and reasoning (KR'89). Toronto, Canada, May 15–18 1989* (pp. 245–255). San Francisco, CA: Morgan Kaufmann.

Matt, P.A., Toni, F., Stournaras, T., & Dimitrelos, D. (2008). Argumentation-based agents for eProcurement. In M. Berger, B. Burg, & S. Nishiyama (Eds.), *Proceedings of the 7th int. conf. on autonomous agents and multiagent systems (AAMAS 2008) – industry and applications track* (pp. 71–74), New York, NY: ACM.

Matt, P.A., Toni, F., & Vaccari, J. (2010). Dominant decisions by argumentation agents. In P. McBurney, I. Rahwan, S. Parsons, & N. Maudet (Eds.), *Proceedings of the sixth international workshop on argumentation in multi-agent systems (ArgMAS 2009), affiliated to AAMAS 2009*, Vol. 6057 of *Lecture notes in computer science* (pp. 42–59). Berlin, Germany: Springer.

McCarthy, J. (1980). Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, *13*, 27–39.

Modgil, S., & Prakken, H. (2013). A general account of argumentation with preferences. *Artificial Intelligence*, *195*, 361–397.

Pollock, J.L. (1987). Defeasible reasoning. *Cognitive Science*, *11*, 481–518.

Prakken, H., & Vreeswijk, G. (2002). Logics for defeasible argumentation. In D. Gabbay & F. Guenthner (Eds.), *Handbook of philosophical logic, second edition* (pp. 219–318). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, *13*, 81–132.

Sartor, G. (1994). A formal model of legal argumentation. *Ratio Juris*, *7*, 212–226.

Schulz, C., & Toni, F. (2013). ABA-based answer set justification (technical communication). *Theory and Practice of Logic Programming*, on-line supplement, Vol. 13, no. 4–5.

Thang, P., & Luong, H. (2013). Translating preferred subtheories into structured argumentation. *Journal of Logic and Computation*, to appear.

Toni, F. (2008). Assumption-based argumentation for closed and consistent defeasible reasoning. In K. Satoh, A. Inokuchi, K. Nagao, & T. Kawamura (Eds.), *New frontiers in artificial intelligence: JSAI 2007* *conference and workshops revised selected papers*. Lecture Notes in Computer Science 4914 (pp. 390–402). Berlin, Germany: Springer.

Toni, F. (2012). Reasoning on the web with assumption-based argumentation. In T. Eiter and T. Krennwallner (Eds.), *Reasoning web. Semantic technologies for advanced query answering 8th international summer school 2012, Vienna, Austria, September 3-8, 2012. Proceedings*. Lecture Notes in Computer Science 7487 (pp. 370–386). Berlin, Germany: Springer.

Toni, F. (2013). A generalised framework for dispute derivations in assumption-based argumentation. *Artificial Intelligence*, *195*, 1–43.