Chapter 1
Assumption-Based Argumentation

Phan Minh Dung, Robert A. Kowalski, and Francesca Toni

1.1 Introduction

Assumption-Based Argumentation (ABA) [4, 3, 27, 9, 12, 20, 22] was developed, starting in the 90s, as a computational framework to reconcile and generalise most existing approaches to default reasoning [24, 25, 4, 3, 27, 26]. ABA was inspired by Dung's preferred extension semantics for logic programming [10, 7], with its dialectical interpretation of the acceptability of negation-as-failure assumptions based on the notion of "no-evidence-to-the-contrary" [10, 7], by the Kakas, Kowalski and Toni interpretation of the preferred extension semantics in argumentation-theoretic terms [24, 25], and by Dung's abstract argumentation (AA) [6, 8].

Because ABA is an instance of AA, all semantic notions for determining the "acceptability" of arguments in AA also apply to arguments in ABA. Moreover, like AA, ABA is a general-purpose argumentation framework that can be instantiated to support various applications and specialised frameworks, including: most default reasoning frameworks [4, 3, 27, 26] and problems in legal reasoning [27, 13], game-theory [8], practical reasoning and decision-theory [33, 29, 15, 28, 14]. However, whereas in AA arguments and attacks between arguments are abstract and primitive, in ABA arguments are deductions (using *inference rules* in an underlying logic) supported by *assumptions*. An attack by one argument against another is a deduction by the first argument of the *contrary* of an assumption supporting the second argument.

Differently from a number of existing approaches to non-abstract argumentation (e.g. argumentation based on classical logic [2] and DeLP [23]) ABA does not have explicit rebuttals and does not impose the restriction that arguments have consistent and minimal supports. However, to a large extent, rebuttals can be obtained "for

Phan Minh Dung
Asian Institute of Technology, Thailand, e-mail: dung@cs.ait.ac.th

Robert A. Kowalski, Francesca Toni
Imperial College London, UK, e-mail: {rak,ft}@doc.ic.ac.uk

free" [27, 9, 33]. Moreover, ABA arguments are guaranteed to be "relevant" and largely consistent [34].

ABA is equipped with a computational machinery (in the form of *dispute derivations* [9, 12, 19, 20, 22]) to determine the acceptability of claims by building and exploring a dialectical structure of a proponent's argument for a claim, an opponent's counterarguments attacking the argument, the proponent's arguments attacking all the opponents' counterarguments, and so on. This computation style, which has its roots in logic programming, has several advantages over other computational mechanisms for argumentation. The advantages are due mainly to the fine level of granularity afforded by interleaving the construction of arguments and determining their "acceptability".

The chapter is organised as follows. In Sections 1.2 and 1.3 we define the ABA notions of argument and attack (respectively). In Section 1.4, we define "acceptability" of sets of arguments, focusing on *admissible* and *grounded* sets of arguments. In Section 1.5 we present the computational machinery for ABA. In Section 1.6 we outline some applications of ABA. In Section 1.7 we conclude.

1.2 Arguments in ABA

ABA frameworks [3, 9, 12] can be defined for any *logic* specified by means of *inference rules*, by identifying sentences in the underlying *language* that can be treated as *assumptions* (see Section 1.3 for a formal definition of ABA frameworks). Intuitively, *arguments* are "deductions" of a conclusion (or claim) supported by a set of assumptions.

The inference rules may be domain-specific or domain-independent, and may represent, for example, causal information, argument schemes, or laws and regulations. Assumptions are sentences in the language that are open to challenge, for example uncertain beliefs ("it will rain"), unsupported beliefs ("I believe X"), or decisions ("perform action A"). Typically, assumptions can occur as premises of inference rules, but not as conclusions. ABA frameworks, such as logic programming and default logic, that have this feature are said to be *flat* [3]. We will focus solely on flat ABA frameworks. Examples of non-flat frameworks can be found in [3].

As an example, consider the following simplification of the argument scheme from expert opinion [37]:

Major premise: Source E is an expert about A.

Minor premise: E asserts that A is true.

Conclusion: A may plausibly be taken as true.

This can be represented in ABA by a (domain-independent) inference rule: [1]

[1] In this chapter, we use inference rule schemata, with variables starting with capital letters, to stand for the set of all instances obtained by instantiating the variables so that the resulting premises and conclusions are sentences of the underlying language. For simplicity, we omit the formal definition of the language underlying our examples.

$h(A) \leftarrow e(E,A), a(E,A), arguably(A)$

with conclusion $h(A)$ ("A holds") and premises $e(E,A)$ ("E is an expert about A"), $a(E,A)$ ("E asserts A"), and an assumption $arguably(A)$. This assumption can be read in several ways, as "there is no reason to doubt that A holds" or " the complement of A cannot be shown to hold" or "the defeasible rule - that a conclusion A holds if a person E who is an expert in A asserts that A is the case – should not apply". The inference rule can be understood as the representation of this defeasible rule as a strict (unchallangable) rule with and extra, defeasible condition – $arguably(A)$ – that is open to challenge. This transformation of defeasible rules into strict rules with defeasible conditions is borrowed from Theorist [31]. Within ABA, defeasible conditions are always assumptions. Different representations of assumptions correspond to different frameworks for defeasible reasoning. For example, in logic programming $arguably(A)$ could be replaced by $not\ \neg h(A)$ (here not stands for negation as failure), and in default logic it could become $M\ h(A)$.

In ABA, attacks are always directed at the assumptions in inference rules. The transformation of defeasible rules into strict rules with defeasible conditions is also used to reduce rebuttal attacks to undercutting attacks, as we will see in Section 1.3.

Note that, here and in all the examples given in this chapter, we represent conditionals as inference rules. However, as discussed in [9], this is equivalent to representing them as object language implications together with modus ponens and and-introduction as more general inference rules. Representing conditionals as inference rules is useful for default reasoning because it inhibits the automatic application of modus tollens to object language implications. However, the ABA approach applies to any logic specified by means of inference rules, and is not restricted in the way illustrated in our examples in this chapter.

Suppose we wish to apply the inference rule above to the concrete situation in which a professor of computer science (cs), say jo, advises that a software product sw meets a customer's requirements ($reqs$) for speed (s) and usability (u). Suppose, moreover, that professors are normally regarded as experts within (w) their field. This situation can be represented by the additional inference rules:

$reqs(sw) \leftarrow h(ok(sw,s)), h(ok(sw,u))$;
$a(jo, ok(sw,s)) \leftarrow$; $\quad a(jo, ok(sw,u)) \leftarrow$; $\quad prof(jo,cs) \leftarrow$;
$w(cs, ok(sw,s)) \leftarrow$; $\quad w(cs, ok(sw,u)) \leftarrow$;
$e(X,A) \leftarrow prof(X,S), w(S,A), c\_prof(X,S)$

Note that all these inference rules except the last are problem-dependent. Note also that, in general, inference rules may have empty premises.

The potential assumptions in the language underlying all these inference rules are (instances of) the formulae $arguably(A)$ and $c\_prof(X,S)$ ("X is a credible professor in S"). Given these inference rules and pool of assumptions, there is an argument with assumptions $\{arguably(ok(sw,s)),\ arguably(ok(sw,u)),\ c\_prof(jo,cs)\}$ supporting the conclusion (claim) $reqs(sw)$.

In the remainder, for simplicity we drop the assumptions $arguably(A)$ and replace the inference rule representing the scheme from expert opinion simply by $h(A) \leftarrow e(E,A), a(E,A)$. With this simplification, there is an argument for $reqs(sw)$ supported by $\{c\_prof(jo,cs)\}$.

Informally, an argument is a deduction of a conclusion (claim) c from a set of assumptions S represented as a tree, with c at the root and S at the leaves. Nodes in this tree are connected by the inference rules, with sentences matching the conclusion of an inference rule connected as parent nodes to sentences matching the premises of the inference rule as children nodes. The leaves are either assumptions or the special extra-logical symbol τ, standing for an empty set of premises. Formally:

Definition 1. Given a deductive system $(\mathscr{L},\mathscr{R})$, with language $\mathscr{L}$ and set of inference rules $\mathscr{R}$, and a set of assumptions $\mathscr{A} \subseteq \mathscr{L}$, an *argument* for $c \in \mathscr{L}$ (the *conclusion* or *claim*) *supported by* $S \subseteq \mathscr{A}$ is a tree with nodes labelled by sentences in $\mathscr{L}$ or by the symbol τ, such that

- the root is labelled by c
- for every node N
  - if N is a leaf then N is labelled either by an assumption or by τ;
  - if N is not a leaf and l_N is the label of N, then there is an inference rule $l_N \leftarrow b_1,\ldots,b_m$ ($m \geq 0$) and
    either $m = 0$ and the child of N is τ
    or $m > 0$ and N has m children, labelled by $b_1,\ldots,b_m$ (respectively)
- S is the set of all assumptions labelling the leaves.

Throughout this chapter, we will often use the following notation

- an argument for (claim) c supported by (set of assumptions) S is denoted by $S \vdash c$

in situations where we focus only on the claim c and support S of an argument. Note that our definition of argument allows for one-node arguments. These arguments consist solely of a single assumption, say α, and are denoted by $\{\alpha\} \vdash \alpha$.

A portion of the argument $\{c\_prof(jo,cs)\} \vdash reqs(sw)$ is given in Fig. 1.1. Here, for simplicity, we omit the right-most sub-tree with root $h(ok(sw,u))$, as this is a copy of the left-most sub-tree with root $h(ok(sw,s))$ but with s replaced by u throughout.

Arguments, represented as trees, display the structural relationships between claims and assumptions, justified by the inference rules. The generation of arguments can be performed by means of a proof procedure, which searches the space of applications of inference rules. This search can be performed in the forward direction, from assumptions to conclusions, in the backward direction, from conclusions to assumptions, or even "middle-out". Our definition of tight arguments in [9] corresponds to the backward generation of arguments represented as trees. Backward generation of arguments is an important feature of dispute derivations, presented in Section 1.5.2.

Unlike several other authors, e.g. those of [2] (see also Chapter 7) and [23] (see also Chapter 8), we do not impose the restriction that the support of an argument be minimal. For example, consider the ABA representation of the scheme from expert opinion, and suppose that our professor of computer science, jo, is also an engineer

(*eng*). Suppose, moreover, that engineers are normally regarded as experts in computer science. These additional "suppositions" can be represented by the inference rules

$eng(jo) \leftarrow; \quad e(X,A) \leftarrow eng(X), w(cs,A), c\_eng(X)$

with (instances of) the formula $c\_eng(X)$ ("X is a credible engineer") as additional assumptions. There are now three, different arguments for the claim $reqs(sw)$:

$\{c\_prof(jo,cs)\} \vdash reqs(sw)$, $\{c\_eng(jo)\} \vdash reqs(sw)$, and

$\{c\_prof(jo,cs), c\_eng(jo)\} \vdash reqs(sw)$.

Only the first two arguments have minimal support. However, all three arguments, including the third, "non-minimal" argument, are *relevant*, in the sense that their assumptions contribute to deducing the conclusion. Minimality is one way to ensure relevance, but comes at a computational cost. ABA arguments are guaranteed to be relevant without insisting on minimality. Note that the arguments of Chapter 17, defined as inference graphs, are also constructed to ensure relevance.

Some authors (e.g. again [2] and [23]) impose the restriction that arguments have *consistent* support [2]. We will see later, in Sections 1.3 and 1.4, that the problems arising for logics including a notion of (in)consistency can be dealt in ABA by reducing (in)consistency to the notion of attack and by employing a semantics that insists that sets of acceptable arguments do not attack themselves.

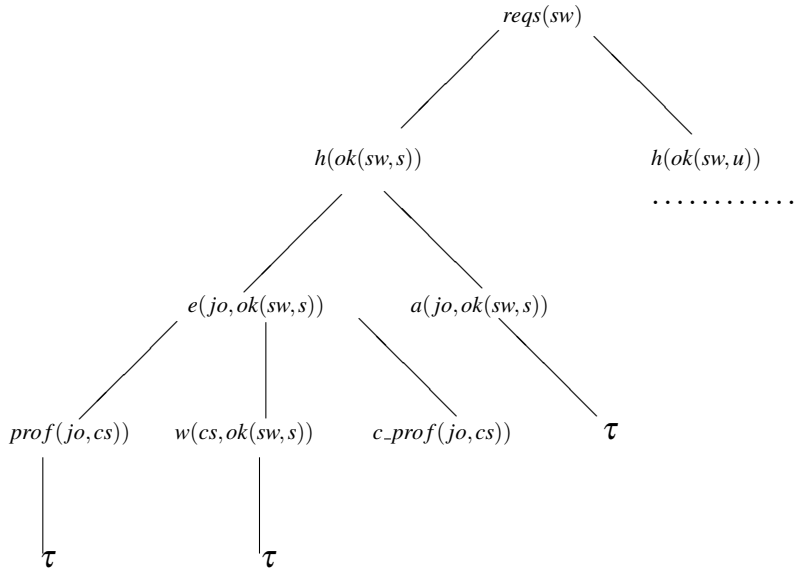

Fig. 1.1 An example argument represented as a tree

[2] Note that these authors define arguments with respect to an underlying logic with an explicit negation and hence a notion of consistency, such that inconsistency implies every sentence in the language. The logic underlying an ABA framework need not have an explicit negation and notion of inconsistency.

1.3 Attacks in ABA

In ABA, the notion of attack between arguments is defined in terms of the *contrary* of assumptions: one argument $S_1 \vdash c_1$ *attacks* another (or the same) argument $S_2 \vdash c_2$ if and only if c_1 is the contrary of an assumption in S_2.

In general, the contrary of an assumption is a sentence representing a challenge against the assumption. For example, the contrary of the assumption "it will rain" might be "the sky is clear". The contrary of the assumption "perform action A" might be "perform action B" (where the actions A and B are mutually exclusive). The contrary of the assumption "I believe X" might be "there is evidence against X". The contrary of an assumption can also represent critical questions addressed to an argument scheme. For example, the argument scheme from expert opinion in Section 1.2 can be challenged by such critical questions as [37]:

CQ1: How credible is E as an expert source?
CQ2: Is E an expert in the field that A is in?
CQ3: Does E's testimony imply A?
CQ4: Is E reliable?
CQ5: Is A consistent with the testimony of other experts?
CQ6: Is A supported by evidence?

For simplicity, we focus here solely on CQ1, because modelling the other questions would require introducing additional assumptions to our earlier representation of the scheme. [3] Providing negative answers to CQ1 can be understood as proving the contrary $\neg c\_prof(jo,cs)$, $\neg c\_eng(jo,cs)$ of the assumptions $c\_prof(jo,cs)$, $c\_eng(jo,cs)$ respectively.

Contraries may be other assumptions or may be defined by inference rules, e.g.

$\neg c\_eng(E,cs) \leftarrow \neg prog(E); \quad \neg prog(E) \leftarrow theo(E)$

where *prog* stands for "programmer" and $theo(E)$ stands for "E is a theoretician". The first rule can be used to challenge the assumption that an engineer is a credible expert in computer science by arguing that the engineer is not a programmer. The second rule can be used to show that an engineer is not a programmer by assuming that he/she is a theoretician (here $theo(E)$ is an additional assumption). Given this representation, the argument $\{c\_eng(jo,cs)\} \vdash reqs(sw)$ is attacked by the argument $\{theo(jo)\} \vdash \neg c\_eng(jo,cs)$.

Definition 2. Given a notion of contrary of assumptions [4],

- *an argument* $S_1 \vdash c_1$ *attacks an argument* $S_2 \vdash c_2$ if and only if the conclusion c_1 of the first argument is the contrary of one of the assumptions in the support S_2 of the second argument;
- *a set of arguments* Arg_1 *attacks a set of arguments* Arg_2 if an argument in Arg_1 attacks an argument in Arg_2.

[3] For example, providing negative answers to CQ5 and CQ6 for A can understood as proving the contrary of the assumption $arguably(A)$ introduced at the beginning of Section 1.2 but ignored afterwards.

[4] See definition 3 for the formal notion of contrary.

This notion of attack between arguments depends only on attacking ("undercutting") assumptions. In many other approaches, however, such as those of Pollock [30] and Prakken and Sartor [32], an argument can attack ("rebut") another argument by deducing the negation of its conclusion. We reduce such "rebuttal" attacks to "undercutting" attacks, as described in [27, 9, 33, 34]. For example, consider the inference rules

$prog(X) \leftarrow works\_for(X, micro), nor(X); \quad works\_for(jo, micro) \leftarrow$

where *micro* is the name of some company, $nor(X)$ stands for "X is normal", and the first inference rule represents the defeasible rule that "normally individuals working at *micro* are programmers". From these and the earlier inference rule for $\neg prog$, we can construct both an argument for $prog(jo)$ supported by $\{nor(jo)\}$ and an argument for $\neg prog(jo)$ supported by $\{theo(ann)\}$. These arguments "rebut" one another but neither one undercuts the other. However, let us set the contrary of assumption $theo(X)$ to $prog(X)$ and the contrary of assumption $nor(X)$ to $\neg prog(X)$. Then, the effect of the rebuttals is obtained by undercutting the assumptions (supporting the arguments for $prog(jo)$ and $\neg prog(jo)$).

Note that an alternative approach to accommodate rebuttals could be to introduce an explicit additional notion of rebuttal attack as done in [11] for logic programming with two kinds of negation.

To complete our earlier definition of argument and attack we need a definition of ABA framework:

Definition 3. An ABA framework is a tuple $\langle \mathscr{L}, \mathscr{R}, \mathscr{A}, \overline{\ } \rangle$ where

- $(\mathscr{L}, \mathscr{R})$ is a deductive system, with a language $\mathscr{L}$ and a set of inference rules $\mathscr{R}$,
- $\mathscr{A} \subseteq \mathscr{L}$ is a (non-empty) set, whose elements are referred to as *assumptions*,
- $\overline{\ }$ is a total mapping from $\mathscr{A}$ into $\mathscr{L}$, where $\overline{\alpha}$ is the *contrary* of α.

1.4 Acceptability of arguments in ABA

ABA can be used to determine whether a given claim is to be "accepted" by a rational agent. The claim could be, for example, a potential belief to be justified, or a goal to be achieved, represented as a sentence in $\mathscr{L}$. In order to determine the "acceptability" of the claim, the agent needs to find an argument for the claim that can be defended against attacks from other arguments. To defend an argument, other arguments may need to be found and they may need to be defended in turn. As in AA, this informal definition of "acceptability" can be formalised in many ways, using the notion of attack between arguments. In this chapter we focus on the following notions of "acceptable" sets of arguments:

- a set of arguments is *admissible* if and only if it does not attack itself and it attacks every argument that attacks it;
- an admissible set of arguments is *complete* if it contains all arguments that it *defends*, where a set of arguments *Arg* defends an argument *arg* if *Arg* attacks all arguments that attack $\{arg\}$;

- the least (with respect to set inclusion) complete set of arguments is *grounded*.

As for AA (see [8] and Chapter 2), in ABA, given a proposed conclusion c, there always exists a grounded set of arguments, and this can be constructed bottom-up [3, 12].

Consider again our formulation of the scheme for expert opinion. The set consisting of the two arguments

arg_1=$\{c\_eng(jo,cs)\} \vdash reqs(sw)$

arg_2=$\{nor(jo)\} \vdash prog(jo)$

is admissible, and as a consequence so is the claim $reqs(sw)$. Indeed, this set does not attack itself and it attacks the argument

arg_3=$\{theo(jo)\} \vdash \neg prog(jo)$.

However, the set $\{arg_1, arg_2\}$ is not (a subset of) the grounded set of arguments. But the set $\{arg_4\}$ is grounded, where

arg_4=$\{c\_prof(jo,cs)\} \vdash reqs(sw)$.

The notion of admissibility is *credulous*, in that there can be alternative, conflicting admissible sets. In the example above, $\{arg_3\}$ is also admissible, but in conflict with the admissible $\{arg_1, arg_2\}$.

In some applications, it is more appropriate to adopt a *sceptical* notion of "acceptability". The notion of grounded set of arguments is sceptical in the sense that no argument in the grounded set is attacked by an admissible set of arguments. Other notions of credulous and sceptical "acceptable" set of arguments can be employed within ABA [3, 27, 12].

Note that the notions of "acceptable" sets of arguments given here are more structured than the corresponding notions of "acceptable" sets of assumptions given in [3, 27, 9, 12]. The correspondence between "acceptability" of arguments and "acceptability" of assumptions, given in [12], is as follows:

- If a set of assumptions S is admissible/grounded then the union of all arguments supported by any subset of S is admissible/grounded;
- If a set of arguments S is admissible/grounded then the union of all sets of assumptions supporting the arguments in S is admissible/grounded.

Note that, if the underlying logic has explicit negation and inconsistency, and we apply the transformation outlined in Section 1.3 (to reduce rebuttals to our undercutting attacks), then an argument has an inconsistent support if and only if it attacks itself. Thus, in such a case, no argument belonging to an "acceptable" set may possibly contain an argument with an inconsistent support [34].

1.5 Computation of "acceptability"

The notion of "acceptability" provides a non-constructive specification for the "acceptability" of sets of arguments. In this section we show how to turn the specification into a constructive proof procedure. As argued in [6, 8], at a conceptual level,

a proof procedure for argumentation consists of two tasks, one for generating arguments and one for determining the "acceptability" of the generated arguments. We have already briefly discussed the computation of arguments in Section 1.2. Below, we first demonstrate how to determine the "acceptability" of arguments that are already constructed, in the spirit of AA, by means of *dispute trees* [9, 12]. Then, we discuss how *dispute derivations* [9, 12, 20, 22], which interleave constructing arguments and determining their "acceptability", can be viewed as generating "approximations" to "acceptable" *dispute trees*.

Dispute derivations are inspired by SLDNF, the "EK" procedure of [17, 10, 7], and the "KT" procedure of [36, 26] for logic programming with negation as failure. Like SLDNF and EK, dispute derivations interleave two kinds of derivations (one for "proving" and one for "disproving"). Like EK, they accumulate defence assumptions and use them for filtering. Like KT, they accumulate culprit assumptions and use them for filtering.

1.5.1 Dispute trees

Dispute trees can be seen as a way of representing a winning strategy for a *proponent* to win a dispute against an *opponent*. The proponent starts by putting forward an initial argument (supporting a claim whose "acceptability" is under dispute), and then the proponent and the opponent alternate in attacking each other's previously presented arguments. The proponent wins if he/she has a counter-attack against every attacking argument by the opponent.

Definition 4. A *dispute tree* for an initial argument a is a (possibly infinite) tree $\mathscr{T}$ such that

1. Every node of $\mathscr{T}$ is labelled by an argument and is assigned the status of *proponent* node or *opponent* node, but not both.
2. The root is a proponent node labelled by a.
3. For every proponent node N labelled by an argument b, and for every argument c attacking b, there exists a child of N, which is an opponent node labelled by c.
4. For every opponent node N labelled by an argument b, there exists exactly one child of N which is a proponent node labelled by an argument attacking b.
5. There are no other nodes in $\mathscr{T}$ except those given by 1-4 above.

The set of all arguments belonging to the proponent nodes in $\mathscr{T}$ is called the *defence set* of $\mathscr{T}$.

Note that a branch in a dispute tree may be finite or infinite. A finite branch represents a winning sequence of arguments (within the overall dispute) that ends with an argument by the proponent that the opponent is unable to attack. An infinite branch represents a winning sequence of arguments in which the proponent counter-attacks every attack of the opponent, ad infinitum. Note that our notion of dispute tree intuitively corresponds to the notion of winning strategy in Chapter 5.

Fig. 1.2 illustrates (our notion of) dispute tree for an extension of our running example, augmented with the following rules

$\neg c\_prof(X,S) \leftarrow ret(X), inact(X)$; $\neg c\_prof(X,S) \leftarrow admin(X), inact(X)$;
$act(X) \leftarrow pub(X)$; $ret(jo) \leftarrow$; $\underline{admin(jo)} \leftarrow$; $pub(jo) \leftarrow$

Here $inact(X)$ is an assumption with $\overline{inact(X)} = act(X)$. These additions express that professors cannot be assumed to be credible ($\neg c\_prof$) if they are retired (*ret*) or cover administrative roles (*admin*) and can be assumed to be inactive (*inact*). *inact* cannot be assumed if its contrary (*act*) can be shown, and this is so for professors with recent publications (*pub*). The resulting, overall ABA framework is summarised in Fig. 1.3. Note that the tree in Fig. 1.2 has an infinite (left-most) branch with

$\{nor(jo)\} \vdash prog(jo)$ child of $\{theo(jo)\} \vdash \neg prog(jo)$ and
$\{theo(jo)\} \vdash \neg prog(jo)$ child of $\{nor(jo)\} \vdash prog(jo)$

ad infinitum. Note also that our intention is to label the opponent nodes in the middle and right-most branches by two different arguments, but both denoted by $\{inact(jo)\} \vdash \neg c\_prof(jo,cs)$. The two arguments differ with respect to the inference rules used to obtain them (the first and second inference rule for $\neg c\_prof$ in Fig. 1.3, respectively) and thus have different representations as argument trees (as in Definition 1).

The definition of dispute tree incorporates the requirement that the proponent must counter-attack every attack, but it does not incorporate the requirement that the proponent does not attack itself. This further requirement is incorporated in the definition of *admissible* and *grounded* dispute trees:

Definition 5. A dispute tree $\mathscr{T}$ is

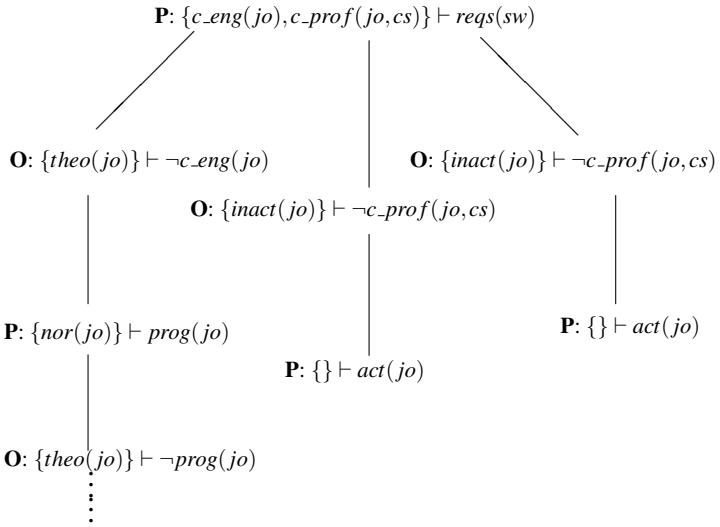

Fig. 1.2 A dispute tree for argument $\{c\_eng(jo), c\_prof(jo,cs)\} \vdash reqs(sw)$, with respect to the ABA framework in Fig. 1.3.

- *admissible* if and only if no argument labels both a proponent and an opponent node. [5]
- *grounded* if and only if it is finite.

Note that, by theorem 3.1 in [12], any grounded dispute tree is admissible. The relationship between admissible/grounded dispute tree and admissible/grounded sets of arguments is as follows:

1. the defence set of an admissible dispute tree is admissible;
2. the defence set of a grounded dispute tree is a subset of the grounded set of arguments;
3. if an argument a belongs to an admissible set of arguments A then there exists an admissible dispute tree for a with defence set A' such that $A' \subseteq A$ and A' is admissible;
4. if an argument a belongs to the grounded set of arguments A (and the set of all arguments supported by assumptions for the given ABA framework is finite) then there exists a grounded dispute tree for a with defence set A' such that $A' \subseteq A$ and A' is admissible.

Results 1. and 3. are proven in [12] (theorem 3.2). Results 2. and 4. follow directly from theorem 3.7 in [26].

Note that the dispute tree in Fig. 1.2 is admissible but not grounded (since it has an infinite branch). However, the tree with root $\{c\_prof(jo,cs)\} \vdash reqs(sw)$ (arg_4 in Section 1.4) and the two right-most subtrees in Fig. 1.2 is grounded.

We can obtain finite trees from infinite admissible dispute trees by using "filtering" to avoid re-defending assumptions that are in the process of being "defended" or that have already successfully been "defended". For example, for the tree in Fig. 1.2, only the (proponent) child of argument $\{theo(jo)\} \vdash \neg prog(jo)$ needs to be constructed. Indeed, since this argument is already being "defended", the remainder of the (infinite) branch can be ignored.

$$
\begin{array}{ll}
\mathscr{R}: & reqs(sw) \leftarrow h(ok(sw,s)), h(ok(sw,u)); \quad h(A) \leftarrow e(E,A), a(E,A); \\
& e(X,A) \leftarrow eng(X), w(cs,A), c\_eng(X); \quad e(X,A) \leftarrow prof(X,S), w(S,A), c\_prof(X,S); \\
& a(jo,ok(sw,s)) \leftarrow; \quad a(jo,ok(sw,u)); \quad eng(jo) \leftarrow; \quad prof(jo,cs) \leftarrow; \\
& w(cs,ok(sw,s)) \leftarrow; \quad w(cs,ok(sw,u)) \leftarrow; \\
& \neg c\_eng(E,cs) \leftarrow \neg prog(E); \quad \neg prog(X) \leftarrow theo(X); \\
& prog(X) \leftarrow works\_for(X,micro), nor(X); \qquad works\_for(bob,micro) \leftarrow; \\
& \neg c\_prof(X,S) \leftarrow ret(X), inact(X); \quad ret(jo) \leftarrow; \\
& \neg c\_prof(X,S) \leftarrow admin(X), inact(X); \quad admin(jo) \leftarrow; \\
& act(X) \leftarrow pub(X); \quad pub(jo) \leftarrow \\
\mathscr{A}: & c\_eng(X); \quad c\_prof(X,S); \quad theo(X); \quad nor(X); \quad inact(X) \\
\overline{}: & \overline{c\_eng(X)} = \neg c\_eng(X); \quad \overline{c\_prof(X,S)} = \neg c\_prof(X,S); \\
& \overline{theo(X)} = prog(X); \quad \overline{nor(X)} = \neg prog(X); \quad \overline{inact(X)} = act(X)
\end{array}
$$

Fig. 1.3 ABA framework for the running example.

[5] Note that admissible dispute trees are similar to the complete argument trees of [2]. We use the term "argument tree" for arguments.

1.5.2 Dispute derivations

Dispute derivations compute ("approximations of) dispute trees top-down, starting by constructing an argument supporting a given claim. While doing so, they perform several kinds of "filtering" exploiting the fact that different arguments may share the same supporting assumptions. Assumptions that are already under attack in the dispute are saved in appropriate data structures (the *defence assumptions* and *culprits*), in order to avoid re-computation. The assumptions used by the proponent (defence assumptions) are kept separate from the assumptions used by the opponent and attacked by the proponent (culprits). The defence assumptions and culprits for the dispute tree in Fig. 1.2 are $\{c\_eng(jo), c\_prof(jo), nor(jo)\}$ and $\{theo(jo), inact(jo)\}$ respectively.

Dispute derivations employ the following forms of filtering:

1. of defence assumptions by defence assumptions, e.g. performed on the defence assumption $theo(jo)$ in the left-most branch of the dispute tree in Fig. 1.2 (this is analogous to the filtering of arguments we discussed earlier);
2. of culprits by defence assumptions and of defence assumptions by culprits, to guarantee that no argument labels both a proponent and opponent node in the tree and thus attacks itself (see Definition 5);
3. of culprits by culprits, for reasons of efficiency; e.g., if the dispute tree in Fig. 1.2 is generated left-to-right, the leaf in the right-most branch does not need to be generated, as the culprit $inact(jo)$ has already been attacked in the middle branch.

The first form of filtering is employed only for computing admissible sets, whereas the other two forms are employed for computing grounded as well as admissible sets.

Dispute derivations are defined in such a way that, by suitably tuning parameters, they can interleave the construction of arguments and determing "acceptability". This interleaving may be very beneficial, in general, as it allows

- abandoning, during their construction, "potential arguments" that cannot be expanded into an actual argument in an "acceptable" set of proponent's arguments,
- avoiding the expansion of the opponent's "potential arguments" into actual arguments when a culprit has already been identified and defeated.

Informally speaking, a *potential argument* of a conclusion c from a set of premises P can be represented as a tree, with c at the root and P at the leaves. As in the case of "actual" arguments (as in Definition 1), nodes in the tree are connected by inference rules. However, whereas the leaves of an argument tree are only assumptions or τ, the leaves of a potential argument can also be non-assumption sentences in $\mathscr{L} - \mathscr{A}$. Dispute derivations successively *expand* potential arguments, using inference rules backwards to replace a non-assumption premise, e.g. p, that matches the conclusion of an inference rule, e.g. $p \leftarrow Q$, by the premises of the rule, Q. In this case, we also say that p is *expanded to* Q.

Fig. 1.1 without the dots is an example of a potential argument for $reqs(sw)$, supported by assumption $c\_prof(jo, cs)$ and non-assumption $h(ok(sw, u))$. Fig. 1.4

shows another example of a potential argument. In general there may be one, many or no actual arguments that can be obtained from a potential argument. For example, the potential argument in Fig. 1.1 may give rise to two actual arguments (supported by $\{c\_prof(jo,cs)\}$ and $\{c\_prof(jo,cs), c\_eng(jo)\}$ respectively), whereas the potential argument in Fig. 1.4 gives rise to exactly one actual argument (supported by $\{inact(jo)\}$). However, if no inference rules were given for *ret*, then no actual argument could be generated from the potential argument in Fig. 1.4.

The benefits of interleaving mentioned earlier can be illustrated as follows:

- A proponent's potential argument is abandoned if it is supported by assumptions that would make the proponent's defence of the claim unacceptable. For example, in Fig. 1.1, if the assumption $c\_prof(jo,cs)$ is "defeated" before $h(ok(sw,u))$ is expanded, then the entire potential argument can be abandoned.
- An opponent's potential argument does not need to be expanded into an actual argument if a culprit can be selected and defeated already in this potential argument. For example, by choosing as culprit and "defeating" the assumption $inact(jo)$ in the potential argument in Fig. 1.4, the proponent "defeats" *any* argument that can be obtained by expanding the non-assumption $ret(jo)$.

However, when a potential argument cannot be expanded into an actual argument, defeating a culprit in the potential argument is wasteful. Nonetheless, dispute derivations employ a *selection function* which, given a potential or actual argument, selects an assumption to attack or a non-assumption to expand. As a special case, the selection function can be *patient* [9, 20], always selecting non-assumptions in preference to assumptions, in which case arguments will be fully constructed before they are attacked. Even in such a case, dispute derivations still benefit from filtering.

Informally, a *dispute derivation* is a sequence of transitions steps from one state of a dispute to another. In each such state, the proponent maintains a set $\mathscr{P}$ of (sentences supporting) potential arguments, representing a *single* way to defend the initial claim, and the opponent maintains a set $\mathscr{O}$ of potential arguments, representing *all* ways to attack the assumptions in $\mathscr{P}$. In addition, the state of the dispute contains the set D of all defence assumptions and the set C of all culprits already encountered in the dispute. The sets D and C are used to filter potential arguments, as we discussed earlier.

A step in a dispute derivation represents either a move by the proponent or a move by the opponent.

A *move by the proponent* either expands one of his/her potential arguments in $\mathscr{P}$, or it selects an assumption in one of the opponent's potential arguments in $\mathscr{O}$ and

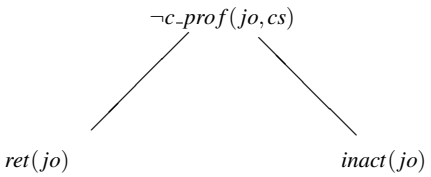

Fig. 1.4 A potential argument for $\neg c\_prof(jo,cs)$, supported by assumption $inact(jo)$ and non-assumption $ret(jo)$.

decides whether or not to attack it. In the first case, it expands the potential argument in only *one* way, and adds any new assumptions resulting from the expansion to D, checking that they are distinct from any assumptions in the culprit set C (filtering defence assumptions by culprits). In the second case, either the proponent ignores the assumption, as a non-culprit, or he/she adds the assumption to C (bearing in mind that, in order to defeat the opponent, he/she needs to counter-attack only one assumption in each of the opponent's attacking arguments). In this latter case, he/she checks that the assumption is distinct from any assumptions in D (filtering culprits by defence assumptions) and checks that it is distinct from any culprit already in C (filtering culprits by culprits). If the selected culprit is not already in C, the proponent adds the contrary of the assumption as the conclusion of a new, one-node potential argument to $\mathscr{P}$ (to construct a counter-attack).

A *move by the opponent*, similarly, either expands one of his/her potential arguments in $\mathscr{O}$, or it selects an assumption to attack in one of the proponent's potential arguments in $\mathscr{P}$. In the first case, it expands a non-assumption premise of the selected potential argument in *all* possible ways, replacing the selected potential argument in $\mathscr{O}$ by all the new potential arguments. In the second case, the opponent does not have the proponent's dilemma of deciding whether or not to attack the assumption, because the opponent needs to generate all attacks against the proponent. Thus, he/she adds the contrary of the assumption as the conclusion of a new, one-node potential argument to $\mathscr{O}$. [6]

A *successful dispute derivation* represents a single way for the proponent to support and defend a claim, but all the ways that the opponent can try to attack the proponent's arguments. Thus, although the proponent and opponent can attack one another before their arguments are fully completed, for a dispute derivation to be successful, all of proponent's arguments must be actual arguments. In contrast, the opponent's defeated arguments may be only potential. In this sense, dispute derivations compute only "approximations" of dispute trees. However, for every successful dispute derivation, there exists a dispute tree that can be obtained by expanding the opponent's potential arguments and dropping tho potential arguments that cannot be expanded, as well as any of the proponent's unnecessary counter-attacks [22].

We give an informal dispute derivation for the running example:

$\mathscr{P}$: I want to determine the "acceptability" of claim $reqs(sw)$ ($D = \{\}$ and $C = \{\}$ initially).

$\mathscr{P}$: I generate a potential argument for $reqs(sw)$ supported by $\{h(ok(sw,s)), h(ok(sw,u))\}$, and then expand it (through several steps) to one supported by $\{c\_prof(jo,cs), h(ok(sw,u))\}$ ($c\_prof(jo,cs)$ is added to D).

$\mathscr{O}$: I attack the assumption $c\_prof(jo,cs)$ in this potential argument by looking for arguments for its contrary $\neg c\_prof(jo,cs)$.

$\mathscr{O}$: I generate two potential arguments for $\neg c\_prof(jo,cs)$, supported by $\{ret(jo), inact(jo)\}$ and $\{admin(jo), inact(jo)\}$ respectively.

[6] If computing admissibility, however, the opponent would not attack assumptions that already belong to D (filtering defence assumptions by defence assumptions).

$\mathscr{P}$: I choose $inact(jo)$ as culprit in the first potential argument by $\mathscr{O}$ ($inact(jo)$ is added to C), and generate (through several steps) an argument for its contrary $act(jo)$, supported by the empty set.

$\mathscr{O}$: There is no point for me to expand this potential argument then. But I still have the attacking argument for $\neg c\_prof(jo,cs)$, supported by $\{admin(jo), inact(jo)\}$.

$\mathscr{P}$: I again choose $inact(jo)$ as culprit, which I have already defeated ($inact(jo) \in C$).

$\mathscr{P}$: There is no attacking argument that I still need to deal with: let me go back to expand the argument for $reqs(sw)$.

$\mathscr{P}$: I need to expand $h(ok(sw,u))$, I can do so and generate (through several steps) an argument for $reqs(sw)$ supported by $\{c\_prof(jo,cs), c\_eng(jo)\}$ ($c\_eng(jo)$ is added to D).

$\mathscr{O}$: I can attack this argument by generating (through several steps) an argument for the contrary $\neg c\_eng(jo)$ of $c\_eng(jo)$: this argument is supported by assumption $theo(jo)$.

$\mathscr{P}$: I can attack this argument by generating (through several steps) a potential argument for the contrary of $theo(jo)$.

...

This dispute corresponds to the top-down and right-to-left construction of (an approximation of) the dispute tree in Fig. 1.2. The dispute ends successfully for computing admissible sets of arguments, but does not terminate for computing grounded sets of arguments. Note that dispute derivations are defined in terms of several parameters: the selection function, the choice of "player" at any specific step in the derivation, the choice of potential arguments for the proponent/opponent to expand etc (see [20, 22]). Concrete choices for some of these parameters (e.g. the choice of the proponent's arguments) correspond to concrete search strategies for finding dispute derivations and computing dispute trees. Concrete choices for other parameters (e.g. the choice of "player") determine how the dispute tree is constructed in a linear manner.

Several notions of dispute derivations have been proposed, differing in the notion of "acceptability" and in the presentation of the computed set of "acceptable" arguments. More specifically, the dispute derivations of [9, 20, 22] compute admissible sets of arguments whereas the dispute derivations of [12] compute grounded and ideal (another notion of sceptical "acceptability") sets of arguments. Moreover, the dispute derivations of [9, 12] compute the union of all sets of assumptions supporting the "acceptable" sets of arguments for the given claim, whereas the dispute derivations of [20, 22] also return explicitly the computed set of "acceptable" arguments and the attacking (potential) arguments, as well as an indication of the attack relationships amongst these arguments.

1.6 Applications

In this section, we illustrate recent applications of ABA for dispute resolution (Section 1.6.1, adapted from [20]) and decision-making (Section 1.6.2, adapted from [15]). For simplicity, the description of these applications is kept short here. For more detail see [13] (for dispute resolution applied to contracts) and [33, 16, 29] (for decision-making in service-oriented architectures). We also show how ABA can be used to model the stable marriage problem (Section 1.6.3, building upon [8]).

1.6.1 ABA for dispute resolution

Consider the following situation, inspired by a real-life court case on contract dispute. A judge is tasked with resolving a disagreement between a software house and a customer refusing to pay for a product developed by the software house. Suppose that this product is the software *sw* in the running example of Fig. 1.3. The judge uses information agree upon by both parties, and evaluates the claim by the software house that payment should be made to them.

All parties agree that payment should be made if the product is delivered on time (del) and is a good product ($goodProd$). They also agree that a product is not good ($badProd$) if it is late ($lateProd$) or does not meet its requirements. As before, we assume that these requirements are speed and usability. There is also the indisputable fact that the software was indeed delivered ($del(sw)$). This situation can be modelled by extending the framework of Fig. 1.3 with inference rules

$payment(sw) \leftarrow del(sw), goodProd(sw)$; $badProd(sw) \leftarrow lateProd(sw)$;
$badProd(sw) \leftarrow \neg reqs(sw)$; $del(sw) \leftarrow$

and assumptions $goodProd(sw)$, $\neg reqs(sw)$, with contraries:

$\overline{goodProduct(sw)} = badProduct(sw)$; $\overline{\neg reqs(sw)} = reqs(sw)$

Given the expert opinions of *jo* (see Section 1.2), the claim that payment should be made is grounded (and thus admissible).

1.6.2 ABA for decision-making

We use a concrete "home-buying" example, in which a buyer is looking for a property and has a number of goals including "structural" features of the property, such as its location and the number of rooms, and "contractual" features, such as the price, the completion date for the sale, etc. The buyer needs to decide both on a property, taking into account the features of the properties ($\mathscr{R}_i$ below), and general "norms" about structural properties ($\mathscr{R}_n$ below). The buyer also needs to decide and agree on a contract, taking into account norms about contractual issues ($\mathscr{R}_c$ below). A simple example of buyer is given by the ABA framework $\langle \mathscr{L}, \mathscr{R}, \mathscr{A}, \overline{\ } \rangle$ with:

- $\mathscr{R} = \mathscr{R}_i \cup \mathscr{R}_n \cup \mathscr{R}_c$ and
  $\mathscr{R}_i$: $number\_of\_rooms = 5 \leftarrow house_1$;
  $number\_of\_rooms = 4 \leftarrow house_2$; $price = £400K \leftarrow house_2$
  $\mathscr{R}_n$: $safe \leftarrow council\_approval, a_1$; $\neg safe \leftarrow weak\_foundations, a_2$;
  $council\_approval \leftarrow completion\_certificate, a_3$
  $\mathscr{R}_c$: $seller\_moves\_abroad \leftarrow house_2$;
  $quick\_completion \leftarrow seller\_moves\_abroad$
- $\mathscr{A} = \mathscr{A}_d \cup \mathscr{A}_u \cup \mathscr{A}_c$ and
  $\mathscr{A}_d = \{house_1, house_2\}$; $\mathscr{A}_c = \{a_1, a_2, a_3\}$; $\mathscr{A}_u = \{\neg council\_approval\ \}$
- $\overline{house_1} = house_2$, $\overline{house_2} = house_1$,
  $\overline{a_1} = \neg safe$, $\overline{a_2} = safe$, $\overline{a_3} = \neg council\_approval$,
  $\overline{\neg council\_approval} = council\_approval$.

Here, there are two properties for sale, $house_1$ and $house_2$. The first has 5 rooms, the second has 4 rooms and costs £ 400K ($\mathscr{R}_i$). The buyer believes that a property approved by the council is normally safe, a completion certificate normally indicates council approval, and a property with weak foundations is normally unsafe ($\mathscr{R}_n$). The buyer also believes that the seller of the second property is moving overseas, and this means that the seller aims at a quick completion of the sale ($\mathscr{R}_c$). Some of the assumptions in the example have a defeasible nature ($\mathscr{A}_c$), others amount to mutually exclusive decisions ($\mathscr{A}_d$) and finally others correspond to genuine uncertainties of the buyer ($\mathscr{A}_u$).

This example combines default reasoning, epistemic reasoning and practical reasoning. An example of "pure" practical reasoning in ABA applied to a problem described in [1] can be found in [35].

1.6.3 ABA for the stable marriage problem

Given two sets M, W of n men and n women respectively, the stable marriage problem (SMP) is the problem of pairing the men and women in M and W in such a way that no two people of opposite sex prefer to be with each other rather than with the person they are paired with. The SMP can be viewed as a problem of finding *stable extensions* in an abstract argumentation framework [8].

Here we show that the problem can be naturally represented in an ABA framework $\langle \mathscr{L}, \mathscr{R}, \mathscr{A}, \overline{\ } \rangle$ with:

- $\mathscr{A} = \{pair(A,B) \mid A \in M, B \in W\}$
- $\overline{pair(A,B)} = contrary\_pair(A,B)$
- $\mathscr{R}$ consists of inference rules
  $contrary\_pair(A,B) \leftarrow prefers(A,D,B), pair(A,D)$;
  $contrary\_pair(A,B) \leftarrow prefers(B,E,A), pair(E,B)$
  together with a set $\mathscr{P}$ of inference rules of the form $prefer(X,Y,Z) \leftarrow$ such that for each person A and for each two different people of the opposite sex

B and C, either $prefer(A,B,C) \leftarrow$ or $prefer(A,C,B) \leftarrow$ belongs to $\mathscr{P}$, where $prefer(X,Y,Z)$ stands for X prefers Y to Z.

The standard formulation of the stable marriage problem combines the rules, assumptions and contraries of this framework with the notion that a set of arguments/assumptions is "acceptable" if and only if it is stable, where

- A set of arguments/assumptions is *stable* if and only if it does not attack itself, but attacks all arguments/assumptions not in the set.

In SMP, the semantics of stable sets forces solutions to be total, pairing all men and women in the sets M and W. The semantics of admissible sets is more flexible. It does not impose totality, and it can be used when the sets M and W have different cardinalities. Consider for example the situation in which two men, a and b, and two women, c and d have the following preferences:

$prefer(a,d,c) \leftarrow$; $prefer(b,c,d) \leftarrow$; $prefer(c,a,b) \leftarrow$; $prefer(d,b,a) \leftarrow$

Although there is no stable solution in which all people are paired with their highest preference, there exist two alternative stable solutions: $\{pair(a,c), pair(b,d)\}$ and $\{pair(a,d), pair(b,c)\}$. However, suppose a third woman, e, enters the scene, turns the heads of a and b, and expresses a preference for a over b, in effect adding to $\mathscr{P}$:

$prefer(a,e,d) \leftarrow$; $prefer(a,e,c) \leftarrow$;

$prefer(b,e,c) \leftarrow$; $prefer(b,e,d) \leftarrow$; $prefer(e,a,b) \leftarrow$

This destroys both stable solutions of the earlier SMP, but has a single maximally admissible solution: $\{pair(a,e), pair(b,c)\}$.

Thus, by using admissibility we can drop the requirement that the number of men and women is the same. Similarly, we can drop the requirement that preferences are total, namely all preferences between pairs of the opposite sex are given.

1.7 Conclusions

In this chapter, we have reviewed assumption-based argumentation (ABA), focusing on relationships with other approaches to argumentation, computation, and applications. In contrast with a number of other approaches, ABA makes use of undercutting as the only way in which one argument can attack another. The effect of rebuttal attacks is obtained in ABA by adding appropriate assumptions to rules and by attacking those assumptions instead. The extent to which such undercutting is an adequate replacement for rebuttals has been explored elsewhere [34], but merits further investigation. Also, again in contrast with some other approaches, we do not insist that the support of an argument in ABA be minimal and consistent. Instead of insisting on minimality, we guarantee that the support of an argument is relevant, as a side-effect of representing arguments as deduction trees. Instead of insisting that the support of an argument is consistent, we obtain a similar effect by imposing the restriction that "acceptable" sets of arguments do not attack themselves.

ABA is an instance of abstract argumentation (AA), and consequently it inherits its various notions of "acceptable" sets of arguments. ABA also shares with AA the

computational machinery of dispute trees, in which a proponent and an opponent alternate in attacking each other's arguments. However, ABA also admits the computation of dispute derivations, in which the proponent and opponent can attack and defeat each other's potential arguments before they are completed. We believe that this feature of dispute derivations is both computationally attractive and psychologically plausible when viewed as a model of human argumentation.

The computational complexity of ABA has been investigated for several of its instances [5] (see also Chapter 6). The computational machinery of dispute derivations and dispute trees is the basis of the CaSAPI argumentation system [7] [19, 20, 22].

Although ABA was originally developed for default reasoning, it has recently been used for several other applications, including dispute resolution and decision-making. ABA is currently being used in the ARGUGRID project [8] to support service selection and composition of services in the Grid and Service-Oriented Architectures. ABA is also being used for several applications in multi-agent systems [21, 18] and e-procurement [29].

References

1. T. Bench-Capon and H. Prakken. Justifying actions by accruing arguments. In *Proc. COMMA'06*, pages 247–258. IOS Press, 2006.
2. P. Besnard and A. Hunter. *Elements of Argumentation*. MIT Press, 2008.
3. A. Bondarenko, P. Dung, R. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101, 1997.
4. A. Bondarenko, F. Toni, and R. Kowalski. An assumption-based framework for non-monotonic reasoning. In *Proc. LPRNR'93*, pages 171–189. MIT Press, 1993.
5. Y. Dimopoulos, B. Nebel, and F. Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence*, 141:57–78, 2002.
6. P. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning and logic programming. In *Proc. IJCAI'93*, pages 852–859. Morgan Kaufmann, 1993.
7. P. Dung. An argumentation theoretic foundation of logic programming. *Journal of Logic Programming*, 22:151–177, 1995.
8. P. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.
9. P. Dung, R. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170:114–159, 2006.
10. P. M. Dung. Negations as hypotheses: An abductive foundation for logic programming. In *Proc. ICLP*, pages 3–17. MIT Press, 1991.
11. P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *Proc. ICLP*, pages 616–630. MIT Press, 1993.
12. P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.
13. P. M. Dung and P. M. Thang. Towards an argument-based model of legal doctrines in common law of contracts. In *Proc. CLIMA IX*, 2008.
14. P. M. Dung, P. M. Thang, and N. D. Hung. Argument-based decision making and negotiation in e-business: Contracting a land lease for a computer assembly plant. In *Proc. CLIMA IX*, 2008.

[7] `http://www.doc.ic.ac.uk/ dg00/casapi.html`

[8] `www.argugrid.eu`

15. P. M. Dung, P. M. Thang, and F. Toni. Towards argumentation-based contract negotiation. In *Proc. COMMA'08*. IOS Press, 2008.
16. P. M. Dung, P. M. Thang, F. Toni, N. D. Hung, P.-A. Matt, J. McGinnis, and M. Morge. Towards argumentation-based contract negotiation. *ARGUGRID Deliverable D.4.1*, 2008.
17. K. Eshghi and R. Kowalski. Abduction compared with negation as failure. In *Proc. ICLP*. MIT Press, 1989.
18. D. Gaertner, J. Rodriguez, and F. Toni. Agreeing on institutional goals for multi-agent societies. In *Proc. COIN*, pages 94–113, 2008.
19. D. Gaertner and F. Toni. CaSAPI: A system for credulous and sceptical argumentation. In *Proc. ArgNMR*, 2007.
20. D. Gaertner and F. Toni. Computing arguments and attacks in assumption-based argumentation. *IEEE Intelligent Systems*, 22(6):24–33, 2007.
21. D. Gaertner and F. Toni. Preferences and assumption-based argumentation for conflict-free normative agents. In *Proc. ArgMAS'07*. Springer, 2007.
22. D. Gaertner and F. Toni. Hybrid argumentation and its computational properties. In *Proc. COMMA'08*. IOS Press, 2008.
23. A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Journal of Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.
24. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.
25. A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. OUP, 1998.
26. A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9:515–562, 1999.
27. R. A. Kowalski and F. Toni. Abstract argumentation. *Journal of Artificial Intelligence and Law*, 4(3-4):275–296, 1996.
28. P.-A. Matt and F. Toni. Basic influence diagrams and the liberal stable semantics. In *Proc. COMMA'08*. IOS Press, 2008.
29. P.-A. Matt, F. Toni, T. Stournaras, and D. Dimitrelos. Argumentation-based agents for eprocurement. In *Proc. AAMAS 2008*, 2008.
30. J. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.
31. D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36(1):27–47, 1988.
32. H. Prakken and G. Sartor. The role of logic in computational models of legal argument: a critical survey. In *Computational Logic: Logic Programming and Beyond – Essays in Honour of Robert A. Kowalski*, pages 342–381. Springer, 2002.
33. F. Toni. Assumption-based argumentation for selection and composition of services. In *Proc. CLIMA VIII*, 2007.
34. F. Toni. Assumption-based argumentation for closed and consistent defeasible reasoning. In *Proc. JSAI 2007*, pages 390–402. Springer, 2008.
35. F. Toni. Assumption-based argumentation for epistemic and practical reasoning. In *Computable Models of the Law*, pages 185–202. Springer, 2008.
36. F. Toni and A. Kakas. Computing the acceptability semantics. In *Proc. LPNMR'95*, pages 401–415. Springer, 1995.
37. D. Walton, C. Reed, and F. Macagno. *Argumentation Schemes*. Cambridge Univ. Press, 2008.