CaSAPI: a system for credulous and sceptical argumentation

Dorian Gaertner and Francesca Toni

Department of Computing
Imperial College London
Email: {dg00,ft}@doc.ic.ac.uk

Abstract. We present the CaSAPI system, implementing (a generalisation of) three existing computational mechanisms [8–10] for determining argumentatively whether potential beliefs can be deemed to be acceptable and, if so, for computing supports for them. These mechanisms are defined in terms of dialectical disputes amongst two fictional agents: a proponent agent, eager to determine the acceptability of the beliefs, and an opponent agent, trying to undermine the existence of an acceptable support for the beliefs, by finding attacks against it that the proponent needs to counter-attack in turn. The three mechanisms differ in the level of scepticism of the proponent agent and are defined for (flat) assumption-based argumentation frameworks [3]. Thus, they can serve as decision-making mechanisms for all instances of these frameworks. In this paper we show how they can be used for logic programming, legal reasoning, practical reasoning, and agent reasoning.

1 Introduction

Assumption-based argumentation [3] has been proven to be a powerful mechanism to understand commonalities and differences amongst many existing frameworks for non-monotonic reasoning, including logic programming [3]. It has also been studied in the context of legal reasoning [14]. Furthermore, the computational complexity of several instances of assumption-based argumentation frameworks for non-monotonic reasoning has been studied in [7].

Assumption-based argumentation frameworks can be coupled with a number of different semantics, all defined in dialectical terms, some credulous and some sceptical, of various degrees. Different computational mechanisms can be defined to match these semantics. In this paper, we consider three existing such mechanisms: GB-dispute derivations for computing the sceptical grounded semantics [9], AB-dispute derivations for computing the credulous admissible semantics [8, 9] and IB-dispute derivations for computing the sceptical ideal semantics [9, 10].

All mechanisms are defined as "dialogues" between two fictional agents: the proponent and the opponent, trying to establish the acceptability of given beliefs with respect to the chosen semantics. The three mechanisms (and corresponding semantics) differ in the level of scepticism of the proponent agent: in GB-dispute derivations the agent is not prepared to take any chances and is completely

sceptical in the presence of seemingly equivalent alternatives; in AB-dispute derivations the agent would adopt any alternative that is capable of counter-attacking all attacks without attacking itself; in IB-dispute derivations, the agent is wary of alternatives, but is prepared to accept common ground between them.

In this paper we describe the CaSAPI [1] system implementating these mechanisms and we illustrate the system and its potential for application in the context of some application scenarios. The system relies upon a generalisation of the original assumption-based argumentation framework and of the computational mechanisms, whereby multiple contraries are allowed. This generalisation is useful to widen the applicability of assumption-based argumentation (*e.g.* for reasoning about decisions). We provide this generalisation explicitly for AB-dispute derivations. The application scenarios we consider are non-monotonic reasoning (using logic programming), legal reasoning (where different regulations need to be applied, taking into account dynamic preferences amongst them), practical reasoning (where decisions need to be made as to which is the appropriate course of action for a given agent), and reasoning to support autonomous agents (about their individual beliefs, desires and intentions, as well as relationships amongst them). Most of these application scenarios require a mapping from appropriate frameworks into assumption-based argumentation.

The paper is structured as follows: in the next section, we will briefly introduce the concept of assumption-based argumentation and describe the three dispute derivations upon which our system is based. Section 3 presents the generalised assumption-based framework and the generalised AB-dispute derivations that our system implements. Section 4 provides a brief description of the CaSAPI system. Applications of this system to the areas of non-monotonic reasoning and legal, practical and agent reasoning are given in Section 5. Finally, we conclude and discuss future work.

2 Background

The definitions and notions in this section are adapted from [3, 8, 10, 9].

Definition 1. *An* assumption-based argumentation framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ *where*

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

We will assume that the inference rules in $\mathcal{R}$ have the syntax

$$c_0 \leftarrow c_1, \ldots, c_n$$

with $n > 0$ or

$$c_0$$

[1] CaSAPI stands for Credulous and Sceptical Argumentation: Prolog Implementation.

where each $c_i \in \mathcal{L}$. c_0 is referred to as the *head* and $c_1, \ldots, c_n$ as the *body* of a rule $c_0 \leftarrow c_1, \ldots, c_n$. The body of a rule c_0 is considered to be empty.

As in [8], we will restrict attention to *flat assumption-based frameworks*, such that if $c \in \mathcal{A}$, then there exists no inference rule of the form $c \leftarrow c_1, \ldots, c_n \in \mathcal{R}$.

Example 1. $\mathcal{L} = \{p, a, \neg a, b, \neg b\}$, $\mathcal{R} = \{p \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a\}$, $\mathcal{A} = \{a, b\}$ and $\overline{a} = \neg a$, $\overline{b} = \neg b$.

An *argument* in favour of a sentence or belief x in $\mathcal{L}$ supported by a set of assumptions $X \subseteq \mathcal{A}$ is a *backward* (or *tight*) *deduction* [8] from x to X, via the backward application of rules in $\mathcal{R}$. For the simple assumption-based framework in example 1, an argument in favour of p supported by $\{a\}$ may be obtained by applying $p \leftarrow a$ backwards.

In order to determine whether a belief is to be held, a set of assumptions needs to be identified that would provide an "acceptable" support for the belief, namely a "consistent" set of assumptions including a "core" support as well as assumptions that defend it. This informal definition of "acceptable" support can be formalised in many ways, using a notion of "attack" amongst sets of assumptions:

Definition 2. X attacks Y *iff there is an argument in favour of some* $\overline{y}$ *supported by (a subset of)* X*, where* y *is in* Y*.*

In Example 1 above, $\{b\}$ attacks $\{a\}$. In this paper we are concerned with the following formalisations of the notion of "acceptability":

- a set of assumptions is *admissible*, iff it does not attack itself and it counter-attacks every set of assumptions attacking it;
- *complete*, iff it is admissible and it contains all assumptions it can defend, by counter-attacking all attacks against them;
- *grounded*, iff it is minimally complete;
- *ideal*, iff it is admissible and contained in all maximally admissible sets.

In the remainder of this section we will illustrate, by means of examples, the three forms of dispute derivations presented in [8, 10, 9], for computing grounded, admissible and ideal sets of assumptions (respectively) in support of given beliefs. For any formal details and results see [8, 10, 9].

2.1 GB-dispute derivations

GB-dispute derivations compute grounded sets of assumptions supporting a given input belief. They are finite sequences of tuples

$$\langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i \rangle$$

where $\mathcal{P}_i$ and $\mathcal{O}_i$ represent (the set of sentences held by) the proponent and opponent in a dispute, A_i holds the set of assumptions generated by the proponent in support of its belief and to defend itself against the opponent, and C_i holds

the set of assumptions in attacks generated by the opponent that the proponent has chosen as "culprits" to be counter-attacked. Each derivation starts with a tuple

$$\langle \mathcal{P}_0 = \{x\}, \mathcal{O}_0 = \{\}, A_0 = \mathcal{A} \cap \{x\}, C_0 = \{\} \rangle$$

where x is the belief whose acceptability the derivation aims at establishing. Then, for every $0 \leq i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and [2]:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if σ is an assumption, then
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$
   (ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup R$
   $A_{i+1} = A_i \cup (\mathcal{A} \cap R)$.
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then
   (i) if σ is an assumption, then
   (a) either σ is ignored, i.e.
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\}\}$
   (b) or $\sigma \notin A_i$ and
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$
   $A_{i+1} = A_i \cup (\{\overline{\sigma}\} \cap \mathcal{A})$
   $C_{i+1} = C_i \cup \{\sigma\}$
   (ii) if σ is not an assumption, then
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\} \cup R \mid \sigma \leftarrow R \in \mathcal{R}\}$.

In Example 1, no GB-dispute derivation for p exists (and, indeed, p is not an acceptable belief according to the grounded semantics) as the search for any such derivation loops and hence no finite sequence of tuples can be found:

$\langle \mathcal{P}_0 = \{p\}, \mathcal{O}_0 = \{\}, A_0 = \{\}, C_0 = \{\} \rangle$,
$\langle \mathcal{P}_1 = \{a\}, \mathcal{O}_1 = \{\}, A_1 = \{a\}, C_1 = \{\} \rangle$, by using rule $p \leftarrow a$,
$\langle \mathcal{P}_2 = \{\}, \mathcal{O}_2 = \{\{\neg a\}\}, A_2 = \{a\}, C_2 = \{\} \rangle$, since $\overline{a} = \neg a$,
$\langle \mathcal{P}_3 = \{\}, \mathcal{O}_3 = \{\{b\}\}, A_3 = \{a\}, C_3 = \{\} \rangle$, by using rule $\neg a \leftarrow b$,
$\langle \mathcal{P}_4 = \{\neg b\}, \mathcal{O}_4 = \{\}, A_4 = \{a\}, C_4 = \{b\} \rangle$, since $\overline{b} = \neg b$,
$\langle \mathcal{P}_5 = \{a\}, \mathcal{O}_5 = \{\}, A_5 = \{a\}, C_5 = \{b\} \rangle$, by using rule $\neg b \leftarrow a$,
...

Note that $\mathcal{P}_1 = \mathcal{P}_5$ and $\mathcal{O}_1 = \mathcal{O}_5$, so both proponent and opponent are repeating their arguments.

[2] For brevity, we indicate here and in all the dispute derivations in the paper only the items of the $i+1$-th tuple that are different from the corresponding items in the i-th tuple: all other items are as in the i-th tuple. For full details, see [9].

2.2 AB-dispute derivations

AB-dispute derivations[3] are modifications of GB-dispute derivation to determine whether beliefs can be held according to the admissible semantics, as follows:

- at step 1.(ii): $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup (R - A_i)$
- step 2.(i)(b) becomes: $\sigma \notin A_i$ and $\sigma \in C_i$ and $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
- a new step 2.(i)(c) is added: $\sigma \notin A_i$ and $\sigma \notin C_i$ and

(c.1) if $\overline{\sigma}$ is not an assumption, then
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$
$C_{i+1} = C_i \cup \{\sigma\}$

(c.2) if $\overline{\sigma}$ is an assumption, then
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
$A_{i+1} = A_i \cup \{\overline{\sigma}\}$
$C_{i+1} = C_i \cup \{\sigma\}$

- at step 2. (ii): $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\} \cup R \mid \sigma \leftarrow R \in \mathcal{R}, \text{and} R \cap C_i = \{\}\}$.

For Example 1, an AB-dispute derivation exists, following up from the earlier GB-derivation with a terminating step

$$\langle \mathcal{P}_5 = \{\}, \mathcal{O}_5 = \{\}, A_5 = \{a\}, C_5 = \{b\} \rangle$$

computing an admissible support $\{a\}$ for p ($\{a\}$ is indeed admissible since it does not attack itself and it counter-attacks $\{b\}$, the only attack against it).

2.3 IB-dispute derivations

IB-dispute derivations are extensions of AB-dispute derivations, in that they are finite sequences of tuples

$$\langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i, \mathcal{F}_i \rangle$$

where the new component $\mathcal{F}_i$ holds the set of all (potential) attacks against $\mathcal{P}_i$. IB-dispute derivations deploy Fail-dispute derivations to check that no admissible extensions of any element in any $\mathcal{F}_i$ exists. For lack of space we simply exemplify IB-dispute derivations here (see [10, 9] for details).

Example 2. $\mathcal{L} = \{a, \neg a, b, \neg b, c, \neg c, d, \neg d\}$, $\mathcal{R} = \{\neg a \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a; \neg c \leftarrow d; \neg d \leftarrow c\}$, $\mathcal{A} = \{a, b, c, d\}$ and $\overline{x} = \neg x$, for all $x \in \mathcal{A}$.

Given the framework in Example 2, an IB-derivation for $\neg a$ is:

$\langle \mathcal{P}_0 = \{\neg a\}, \mathcal{O}_0 = \{\}, A_0 = \{\}, C_0 = \{\}, \mathcal{F}_0 = \{\} \rangle$
$\langle \mathcal{P}_1 = \{b\}, \mathcal{O}_1 = \{\}, A_1 = \{b\}, C_1 = \{\}, \mathcal{F}_1 = \{\} \rangle$,

[3] AB-dispute derivations are a slight modification of the dispute derivations of [8], presented in [9].

$\langle \mathcal{P}_2 = \{\}, \mathcal{O}_2 = \{\{\neg b\}\}, A_2 = \{b\}, C_2 = \{\}, \mathcal{F}_2 = \{\}\rangle$,
$\langle \mathcal{P}_3 = \{\}, \mathcal{O}_3 = \{\{a\}\}, A_3 = \{b\}, C_3 = \{\}, \mathcal{F}_3 = \{\}\rangle$,
$\langle \mathcal{P}_4 = \{\neg a\}, \mathcal{O}_4 = \{\}, A_4 = \{b\}, C_4 = \{a\}, \mathcal{F}_4 = \{\{a\}\}\rangle$,
$\langle \mathcal{P}_5 = \{\}, \mathcal{O}_5 = \{\}, A_5 = \{b\}, C_5 = \{a\}, \mathcal{F}_5 = \{\{a\}\}\rangle$,
$\langle \mathcal{P}_6 = \{\}, \mathcal{O}_6 = \{\}, A_6 = \{b\}, C_6 = \{a\}, \mathcal{F}_6 = \{\}\rangle$.

The transition between the penultimate and the last tuple in the sequence above requires the existence of a Fail-dispute derivation confirming that no admissible extension of $\{a\} \in \mathcal{F}_5$ exists.

The derivation succeeds in computing support $\{b\}$ for $\neg a$. The set $\{b\}$ is ideal as it is admissible and contained in every maximally admissible set of assumptions (there are two such sets: $\{b, c\}$ and $\{b, d\}$).

Note that there is no GB-dispute derivation for $\neg a$ (which indeed is not supported by any grounded set of assumptions). Also, note that there exists an AB-dispute derivation for $\neg a$, as well as for $\neg c$ and $\neg d$, but no GB- or IB-dispute derivation exists for the latter two beliefs. Thus, the proponent agent in GB-derivations is the most sceptical, followed by the proponent agent in IB-derivations. The proponent agent in AB-derivations on the other hand is completely credulous.

3 Generalisation of assumption-based argumentation frameworks

In order to widen their applicability (*e.g.* for practical reasoning), assumption-based argumentation frameworks need to be generalised as follows:

Definition 3. *A* generalised assumption-based framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{C}on\rangle$ *where* $\mathcal{L}$, $\mathcal{R}$, $\mathcal{A}$ *are as in conventional assumption-based frameworks, and* $\mathcal{C}on$ *is a (total) mapping from assumptions in* $\mathcal{A}$ *into* sets of sentences *in* $\mathcal{L}$.

Intuitively, in this generalised framework, assumptions admit multiple contraries. Given a generalised assumption-based argumentation framework, the notion of attack between sets of assumptions becomes:

Definition 4. X attacks Y *iff there is an argument in favour of some* x *supported by (a subset of)* X *where* $x \in \mathcal{C}on(y)$ *and* y *is in* Y.

All dispute derivations defined in previous works [8, 10, 9] can thus be modified for generalised assumption-based frameworks. In what follows, we show how this can be done for AB-dispute derivations (the modifications are typeset in bold font), as this is the "core" form of dispute derivation (GB-dispute derivations are a simplification and IB-dispute derivation an extension of AB-dispute derivations):

Definition 5. *Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{C}on \rangle$ be a generalised assumption-based argumentation framework. Given a selection function, a* generalised AB-dispute derivation of a defence set *A for a sentence α is a finite sequence of* quadruples

$$\langle \mathcal{P}_0, \mathcal{O}_0, A_0, C_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, A_n, C_n \rangle$$

where $\mathcal{P}_0 = \{\alpha\}$ $\quad A_0 = \mathcal{A} \cap \{\alpha\}$ $\quad \mathcal{O}_0 = C_0 = \{\}$
$\mathcal{P}_n = \mathcal{O}_n = \{\}$ $\quad A = A_n$

and for every $0 \leq i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and:

1. *If $\sigma \in \mathcal{P}_i$ is selected then*

   (i) *if σ is an assumption, then*
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{x\} \mid x \in \mathcal{C}on(\sigma)\}$
   *(***In the original AB-dispute derivation,** $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$.*)*

   (ii) *if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and*
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup (R - A_i)$
   $A_{i+1} = A_i \cup (\mathcal{A} \cap R)$.

2. *If S is selected in $\mathcal{O}_i$ and σ is selected in S then*

   (i) *if σ is an assumption, then*
   (a) *either σ is ignored, i.e.*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\}\}$
   (b) *or $\sigma \notin A_i$ and $\sigma \in C_i$ and*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   (c) *or $\sigma \notin A_i$ and $\sigma \notin C_i$ and* **chosen some** *$x \in \mathcal{C}on(\sigma)$ (***in the original AB-dispute derivations,** $x = \overline{\sigma}$*)*
   (c.1) *if x is not an assumption, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{x\}$
   $C_{i+1} = C_i \cup \{\sigma\}$
   (c.2) *if x is an assumption* **and it does not belong to** *C_i, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $A_{i+1} = A_i \cup \{x\}$
   $C_{i+1} = C_i \cup \{\sigma\}$

   (ii) *if σ is not an assumption, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\} \cup R \mid \sigma \leftarrow R \in \mathcal{R}, \text{ and } R \cap C_i = \{\}\}$.

Note that in step 2(i)(c) the choice of counter-attack is based upon the choice of contrary of the selected culprit. This choice is made randomly, but can be customised if necessary.

The definitions of GB- and IB-dispute derivations can be modified in a similar fashion, by considering all contraries (of a given assumption) when extending $\mathcal{O}$ (to find attacks against the assumption) and by choosing one contrary (of a given "culprit" assumption) when extending $\mathcal{P}$ (to counter-attack the assumption).

4 System Description

In this section, we will describe the CaSAPI system, a Prolog implementation for credulous and sceptical argumentation based upon the computation of dispute derivations for grounded beliefs (GB-dispute derivations), admissible beliefs (AB-dispute derivations) and ideal beliefs (IB-dispute derivations) for the generalised assumption-based frameworks described in the previous section. The latest version of CaSAPI can be downloaded from `www.doc.ic.ac.uk/~dg00/casapi.html`. The system is developed in Sicstus Prolog but runs on most variants of Prolog[4].

4.1 How to use CaSAPI

After invoking a Prolog process and loading the CaSAPI program, users need to load the input assumption-based framework[5] and the beliefs to be proved. These are best specified in a separate file, prior to invoking Prolog.

Rules in $\mathcal{R}$ are represented as facts of a binary relation `myRule/2` consisting of a left- and right-hand side. The first argument holds the head of the rule and the second argument a list containing the body of the rule. Assumptions in A and beliefs to be proved are represented as unary predicates `myAsm/1` and `toBeProved/1` (respectively) using a list notation for their respective argument. The latter predicate allows queries about more than one belief to be expressed. The notion of contrary can also be customised using an binary relation `contrary/2`. In order to illustrate the representation of assumption-based frameworks, Example 1 from Section 2 is represented as follows:

```
myRule(p,[a]).
myRule(not(a),[b]).
myRule(not(b),[a]).

myAsm([a,b]).

toBeProved([p]).
```

[4] CaSAPI has been successfully tested using SWI Prolog, for example.
[5] The language $\mathcal{L}$ does not need to be specified explicitly.

```
contrary(a,not(a)).
contrary(b,not(b)).
```

The users can then control the kind of dispute derivation they want to employ (GB, AB or IB), the amount of output to the screen (silent, compact or noisy) and the number of supports computed (one or all). They specify their choices as arguments to the command `run/3` and CaSAPI will begin the argumentation process in a manner dictated by the users' choices. For example, in order to run AB-dispute derivations in silent mode and asking for only one answer, one needs to specify: `run(ab,s,1)`. Furthermore, for running GB-dispute derivations in noisy mode asking for all answers, one needs to execute: `run(gb,n,a).` Note that *all answers* here refers to all answers that can be computed using the dispute derivation in question.

4.2 Design Choices

We have picked Sicstus Prolog as the implementation language of choice since we intend to employ some of its constraint solving features in future versions of CaSAPI. In the current version 2.0 we do not make use of any Sicstus specific code and hence it should run on most standard Prolog engines

One of the interesting properties of Prolog is its handling of variables. Instantiation takes place when a binding can be made, but backtracking allows new instantiations to override old ones where possible. We made use of this feature in that we allow variables in the definition of rules, assumptions and contraries in CaSAPI. This can be seen as a shortcut to writing out all the ground instances of the predicate in question.

A further design choice, that paves the way to interesting experimental research, is the fact that the selection strategies of the agent are not hard-wired into CaSAPI. Different selection strategies do not affect the result of the argumentation process, but have a significant impact on efficiency. Indeed, these strategies control how the dispute trees are generated and hence can lead to early pruning for certain trees. One simple example for a selection strategy is:

```
selFunc([HeadProponent|_],_,HeadProponent,[]).
selFunc([],[[OppHead|OppTail]|_],OppHead,[OppHead|OppTail]).
selFunc([],[],_,_).    % Finished.
```

The first two arguments are the beliefs held by the proponent and opponent, respectively. The third argument is used to return the chosen element and the fourth one returns the set of beliefs of the opponent that this element was chosen from – if applicable. If both the proponent and the opponent have no further beliefs to investigate, the argumentation process terminates. In this simple example, all beliefs of the proponent are handled before the opponent gets to reply. More sophisticated selection strategies can easily be imagined and have been used as defaults in the CaSAPI system.

As we have hinted to before, CaSAPI allows queries to involve sets of beliefs to be proved, rather than individual beliefs as in the original formulation of dispute derivations. But the biggest innovation is the extension of the argumentation framework to allow multiple contraries. The theoretical aspects of this extension have been discussed in the previous section. An example where this extension is employed will be given in Section 5.3 on practical reasoning.

4.3 Worked Example

We illustrate an exemplary execution trace of the CaSAPI system in the case of Example 2 from Section 2. Here and in the remainder of the paper, we represent negative literals $\neg p$ as `not(p)`. In this example, basically `a`, `b` and `c`, `d` are (pairwise) mutually exclusive. Intending to prove the belief `not(a)`, after feeding the following input program:

```
myRule(not(a),[a]).
myRule(not(a),[b]).
myRule(not(b),[a]).
myRule(not(c),[d]).
myRule(not(d),[c]).

myAsm([a,b,c,d]).

toBeProved([not(a)]).

contrary(X,not(X)) :- myAsm(L), member(X,L).
```

into CaSAPI, one needs to choose the execution options. Deciding to use admissible belief semantics, demanding verbose output and requesting only one solution, the following will happen: `not(a)` can only be proved by either the first or second of the rules given above.

```
Step 0:
- Content of this quadruple:
- PropNods: [not(a)]
- OppoNods: []
- DfnceAss: []
- Culprits: []

CASE 1ii
Step 1:
- Content of this quadruple:
- PropNods: [a]
- OppoNods: []
- DfnceAss: [a]
- Culprits: []
```

```
CASE 1i
Step 2:
- Content of this quadruple:
- PropNods: []
- OppoNods: [[not(a)]]
- DfnceAss: [a]
- Culprits: []
```

After some backtracking, the output ends with the final answer:

```
Step 5:
- Content of this quadruple:
- PropNods: []
- OppoNods: []
- DfnceAss: [b,b]
- Culprits: [a]

FINISHED, the defence set is: [b,b]
Without duplicates it is: [b]

yes
```

The defence set `[b]` indicates which assumption(s) need to be made and are sufficient to defend the belief `not(a)` against all possible attacks.

5 Applications

In this section we give examples of how assumption-based argumentation in general and CaSAPI in particular can be applied. First we consider logic programming as an instance of non-monotonic reasoning, and then look at legal, practical and agent reasoning. Note that non-monotonic reasoning using default logic could also be modelled, following [3].

5.1 Non-monotonic reasoning: Logic programming.

A logic program P can be seen as an assumption-based framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where $\mathcal{R}{=}P$, $\mathcal{L}$ is the Herbrand base of P together with the set of all negations of atoms in such Herbrand base, $\mathcal{A}$ is the set of all negative literals in $\mathcal{L}$, and $\overline{not\, p} = p$ for all negative literals $not\, p$ in $\mathcal{A}$. Logic programming queries correspond to sets (conjunctions) of beliefs for which we want to compute dispute derivations.

In this instance of assumption-based frameworks, the admissible, grounded, and ideal semantics correspond (see [3] and [9]) to partial stable models [17], well-founded model [13], and ideal semantics [1], respectively. Although the theoretical framework is propositional, our Prolog implementation allows us to deal with variables, both in the rules of the deductive system and in the beliefs to be proved, as shown below.

Example 3. $P = \{p(X) \leftarrow not\, p(X);$
$p(X) \leftarrow not\, q(X);$
$q(X) \leftarrow not\, p(X);$
$r(X) \leftarrow not\, t(X);$
$t(X) \leftarrow not\, r(X)\}$

Given the logic program from Example 3 and queries $Q_1 = p(a)$, $Q_2 = q(a)$, $Q_3 = r(a)$, $Q_4 = t(a)$, the system computes the following answers, respectively:
GB-dispute derivations: loops, `no`, loops, loops
AB-dispute derivations: $\{not\, q(a)\}$, `no`, $\{not\, t(a)\}$, $\{not\, r(a)\}$
IB-dispute derivations: $\{not\, q(a)\}$, `no`, `no`, `no`

Example 4. $P = \{p \leftarrow not\, q;$
$q \leftarrow not\, r;$
$r \leftarrow not\, s;$
$s \leftarrow not\, q\}$

Given the logic program from Example 4 — with an odd-loop via negation — and query $Q_1 = p$, the system computes `no` for all three kinds of derivations.

5.2 Legal reasoning.

This kind of reasoning often requires dealing with defeasible rules and facts (possibly under dispute), strict rules and facts (beyond dispute) and preferences amongst defeasible rules and facts (possibly under dispute). We show here how a concrete example of legal reasoning from [15] can be dealt with by means of our CaSAPI system, following the formalisation of the problem given in [14].

Example 5. Consider the following set of defeasible rules, including rules defining preferences between rules[6]:

$r_1(X)$: *X's exterior may not be modified if X is a protected building.*
$r_2(X)$: *X's exterior may be modified if X needs restructuring.*
$r_3(X,Y)$: $R_1(X) > R_2(Y)$ *if* $R_1(X)$ *concerns artistic buildings and* $R_2(Y)$ *concerns town planning.*
$t(X,Y)$: $R_1(X) > R_2(Y)$ *if* $R_1(X)$ *is later than* $R_2(Y)$.

and the following six facts/strict rules:

$r_1(X)$ *concerns artistic buildings.*
$r_2(X)$ *concerns town planning.*
$r_2(X)$ *is later than* $r_1(X)$.
$r_3(X,Y)$ *is later than* $t(X,Y)$.
villa is a protected building.

[6] For all kinds of rules, we adopt a representation in pseudo-natural language, with variables implicitly universally quantified with scope the rules.

villa needs restructuring.

Intuitively, the conclusion that the exterior of the villa may not be modified should be drawn. Both rules r_1 and r_2 apply and the *meta-rules* r_3 and t deciding the priorities between r_1 and r_2 also apply both, but according to meta-rule t, the importance of r_3 is higher than its own importance. Hence, r_3 should be applied which gives r_1 priority over r_2. Following [14], this problem can be represented as a logic program (and thus as an assumption-based framework, as explained above):

$$
\begin{array}{l}
villa\text{'s exterior may not be modified} \leftarrow not\ defeated(r_1(villa)) \\
villa\text{'s exterior may be modified} \leftarrow not\ defeated(r_2(villa)) \\
defeated(r_1(villa)) \leftarrow not\ defeated(t(villa, villa)),\ not\ defeated(r_2(villa)) \\
defeated(r_2(villa)) \leftarrow not\ defeated(r_3(villa, villa)), not\ defeated(r_1(villa)) \\
defeated(t(villa, villa)) \leftarrow not\ defeated(t((villa, villa), (villa, villa))), \\
\qquad\qquad\qquad\qquad not\ defeated(r_3(villa, villa))
\end{array}
$$

GB-, AB- and IB-dispute derivations for the belief *villa's exterior not modified* all give the following defence set as an answer: $\{not\ defeated(r_1(villa)),$ $not\ defeated(r_3(villa, villa)),\ not\ defeated(t((villa, villa), (villa, villa)))\}$.

This can be understood as follows: the villa's exterior should not be modified since rule r_1 is not defeated (stating that artistic buildings should not be modified) and rule r_3 is not defeated (stating that rules concerning artistic buildings override rules concerning town planning) and the temporal ordering rule t is not defeated either.

5.3 Practical reasoning.

This form of reasoning requires making decisions in order to achieve certain properties/objectives, having only partial information. We show how to deal with the concrete example in [2], requiring multiple contraries.

Example 6. A judge needs to decide how best to punish a criminal found guilty, while deterring the general public, rehabilitating the offender, and protecting society from further crime. The judge can choose amongst three forms of punishment: (i) imprisonment, (ii) a fine, or (iii) community service. The judge believes that: (i) promotes deterrence and protection to society, but it demotes rehabilitation; (ii) promotes deterrence but has no effect on rehabilitation and protection of society; (iii) promotes rehabilitation but demotes deterrence.

We can represent the problem as a generalised assumption-based framework:

- $\mathcal{A} = \{prison, fine, service, \alpha, \beta, \gamma, \delta\}$,
- $\mathcal{C}on(prison) = \{fine, service\}$, $\mathcal{C}on(fine) = \{prison, service\}$, $\mathcal{C}on(service) = \{prison, fine\}$, $\mathcal{C}on(\alpha) = \{\neg deter\}$, $\mathcal{C}on(\beta) = \{deter\}$, $\mathcal{C}on(\gamma) = \{\neg rehabilitate\}$, $\mathcal{C}on(\delta) = \{rehabilitate\}$,

– $\mathcal{R}$ consists of nine rules:

$punish \leftarrow prison$	$deter \leftarrow prison, \alpha$	$rehabilitate \leftarrow service, \gamma$
$punish \leftarrow fine$	$deter \leftarrow fine, \alpha$	$\neg rehabilitate \leftarrow prison, \delta$
$punish \leftarrow service$	$\neg deter \leftarrow service, \beta$	$protect \leftarrow prison$

Then, given the goal (belief) *punish*, AB dispute derivations compute the defence set $\{prison\}$, for example. Given also goal *rehab* the defence set $\{service\}$ is computed. One cannot have all goals *punish*, *deter*, *rehabilitate* and *protect* provable (AB dispute derivations return `no`) and it would be interesting to give preferences amongst these, as suggested in [2]. We leave this for future research.

5.4 Agent reasoning

Finally, we will give an example involving a traditional BDI agent [16] that reasons about its beliefs, desires and intentions. We chose the ballroom scenario from [11] and the following setup: picture a traditional ballroom with several male and female dancers; the rules of etiquette state among other things that two dancers agreeing to dance together should be of opposite sex and that female dancers should wait to be approached by a male dancer (with the exception of *ladies' choice night*).

Imagine a female dancer called *anna*, who considers both *bob* and *charlie* to be pretty and who generally intends whatever she desires. This information can be expressed with the following rules in an assumption-based argumentation framework[7]:

$$intend(X) \leftarrow desire(X)$$
$$desire(danceWith(X)) \leftarrow belief(pretty(X)),\ \beta(X)$$
$$\neg desire(danceWith(X)) \leftarrow belief(sameSex(self, X)),\ \alpha(X)$$
$$intend(danceWith(X)) \leftarrow belief(approachedBy(X))$$

$$belief(pretty(bob))$$
$$belief(pretty(charlie))$$

Note that the first rule is domain-independent, whereas the other rules are domain-dependent.

Let $\mathcal{A}$ be the set of (all ground instances of) $\neg belief(X)$ together with $\alpha(X)$ and $\beta(X)$. The latter two assumptions are needed to relate $desire(danceWith(X))$ and $\neg desire(danceWith(X))$ as opposite notions. One cannot directly make them contraries of one another, since neither of them is an assumption.

The $\mathcal{C}on$ relation defines the contrary of any $\neg belief(X)$ as $belief(X)$ and the contrary of $\alpha(X)$ as $desire(danceWith(X))$ and finally, the contrary of $\beta(X)$ as $\neg desire(danceWith(X))$.

Then, asking CaSAPI whether *anna* should intend to dance with *charlie*, the system returns that **y**es, she should intend to dance with that person, provided

[7] We ignore here nested beliefs, desires and intentions for simplicity's sake.

that *anna* believes that *charlie* is not of the same gender. Thus CaSAPI replies with the following defence set: `beta(charlie)`. This is the assumption needed to defend the belief in question and it ensures that $\neg desire(danceWith(charlie))$ does not hold.

The reasoning goes roughly as follows: using the fourth rule, *anna* should intend to dance with *charlie* if she beliefs to have been approached by *charlie*. However, this is not the case. Using the first rule, *anna* should intend to dance with *charlie* if she desires it. She does desire it, since she believes *charlie* is pretty. Now, the fictional opponent who plays devil's advocate in *anna*'s mind may argue that she should not desire (and hence not intend) to dance with *charlie* because *charlie* may be female, too. Therefore, the fictional proponent who defends the query needs to make the additional assumption that *anna* and *charlie* are of opposite gender in order to render the third rule inapplicable.

Note that this is just one simple example of agent reasoning, and more complex and sophisticated forms of reasoning may be afforded by CaSAPI. For example, in case conflicts may arise, *e.g.* due to intending and not intending the same action, the use of preferences, as modelled in legal reasoning, can provide an effective means of conflict resolution. We leave this for future work.

6 Conclusions

In this paper, we have presented a generalisation of computational mechanisms for assumption-based argumentation that allows multiple contraries of assumptions to be expressed. This generalisation enables this kind of argumentation to handle a broader class of applications. Furthermore, we have described the CaSAPI system which implements credulous and (two forms of) sceptical argumentation for this generalisation of assumption-based argumentation and shown how to use the system in some application areas.

Two of these application areas (legal and practical reasoning) assumed a translation (by-hand) from a given formalism into assumption-based argumentation [18]. Future work includes providing appropriate front-ends to our system in order to automate this translation.

We have implemented a number of extensions to theoretical assumption-based argumentation (*e.g.* variables in rules) that would also be worthwhile to formalise in the future.

A number of other argumentation systems exist, for example GORGIAS [6], for credulous argumentation in argumentation frameworks with preferences amongst defeasible rules, the ASPIC system (`http://aspic.acl.icnet.uk/`) [5] dealing with quantitative uncertainty, DeLP [12] for defeasible logic programming, and the system by Krause et al. [4]. These systems are defined for different frameworks for argumentation than ours. It would be interesting to provide a mapping from these various frameworks onto assumption-based argumentation (possibly extended) in order to carry out a full comparison.

Acknowledgements

This research was partially funded by the EC-funded ARGUGRID project. The second author has also been supported by a UK Royal Academy of Engineering/Leverhulme Trust senior fellowship.

References

1. José Júlio Alferes, Phan Minh Dung, and Luís Moniz Pereira. Scenario semantics of extended logic programs. In *LPNMR*, 1993.
2. T. Bench-Capon and H. Prakken. Justifying actions by accruing arguments. In *COMMA*, 2006.
3. A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93(1-2), 1997.
4. D. Bryant and P. Krause. An implementation of a lightweight argumentation engine for agent applications. In *JELIA*, 2006.
5. M. Caminada, S. Doutre, S. Modgil, H. Prakken, and G.A.W. Vreeswijk. Implementations of argument-based inference. In *Review of Argumentation Tech.*, 2004.
6. N. Demetriou and A. C. Kakas. Argumentation with abduction. In *Proceedings of the fourth Panhellenic Symposium on Logic*, 2003.
7. Y. Dimopoulos, B. Nebel, and F. Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence*, 141, 2002.
8. P.M. Dung, R.A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170, 2006.
9. P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. Technical report, Imperial College London, 2006.
10. P.M. Dung, P. Mancarella, and F. Toni. A dialectic procedure for sceptical, assumption-based argumentation. In *COMMA*, 2006.
11. Dorian Gaertner, Keith Clark, and Marek Sergot. Ballroom etiquette: a case study for norm-governed multi-agent systems. In *Proceedings of the 1st International Workshop on Coordination, Organisation, Institutions and Norms*, 2006.
12. A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2), 2004.
13. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3), 1991.
14. R. A. Kowalski and F. Toni. Abstract argumentation. *Journal of AI and Law, Special Issue on Logical Models of Argumentation*, 4(3-4), 1996.
15. H. Prakken and G. Sartor. On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. In *ICAIL*, 1995.
16. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
17. Domenico Saccà and Carlo Zaniolo. Partial models and three-valued models in logic programs with negation. In *LPNMR*, 1991.
18. F. Toni. Assumption-based argumentation for epistemic and practical reasoning. Technical report, Imperial College London, 2007.