Dialectical Proofs
for Constrained Argumentation

Caroline DEVRED \textsuperscript{a} Sylvie DOUTRE \textsuperscript{b} Claire LEFÈVRE \textsuperscript{a} and Pascal NICOLAS \textsuperscript{a}

\textsuperscript{a} LERIA, University of Angers, France
\textsuperscript{b} IRIT, University of Toulouse, France

Abstract. Constrained argumentation frameworks (CAF) generalize Dung’s framework by allowing additional constraints on arguments to be taken into account in the definition of acceptability of arguments. These constraints are expressed by means of a logical formula which is added to Dung’s framework. The resulting system captures several other extensions of Dung’s original system. To determine if a set of arguments is credulously inferred from a CAF, the notion of dialectical proof (alternating pros and cons arguments) is extended for Dung’s frameworks in order to respect the additional constraint. The new constrained dialectical proofs are computed by using Answer Set Programming.

Keywords. Formal models for argumentation, dialogue, implementation, ASP.

Introduction

Argumentation is an approach to nonmonotonic reasoning, based on the notions of argument, and interactions between arguments. Among all the frameworks which have been proposed for argumentation (see \cite{2,16} for a survey), the one by Dung \cite{8} has received a lot of attention. Actually, the high level of abstraction of this framework and its simplicity make it encompass many approaches to nonmonotonic inference and logic programming, and generalize several other approaches to argumentation.

Concretely, \cite{8}’s framework consists of a set of arguments and of an attack relation between arguments. Nothing is said neither on the nature of the arguments, nor on the nature of the attack relation.

Several notions of acceptable sets, namely extensions, are defined as sets of arguments that satisfy some properties. In \cite{8}’s framework, these properties are always defined in terms of attacks between arguments. Some works propose to refine this definition of acceptability by taking into account a notion of preference between arguments, or of values (e.g. \cite{22}). However, it may be necessary to express constraints between arguments which are neither attack-based nor preference-based. An attempt in this sense has been proposed in \cite{21,23}: these works allow the possibility for some arguments to belong to extensions only if necessary. However, these works do not offer the possibility for either an argument \(x\), or an argument \(y\) (\(x\) and \(y\) not attacking each other), to belong to extensions; or the possibility for an argument \(z\) to belong to extensions only if \(x\) and \(y\) both belong to them (\(z\) not being attack-related to \(x\) and \(y\)).
[5] have proposed an extension of Dung’s framework with a notion of constraint, which, without taking into account preferences between arguments, offers a way to specify further, non attack-based, requirements which have to be satisfied by the acceptable sets of arguments. The resulting constrained argumentation framework encompasses Dung’s framework and several of its existing extensions. For instance, as shown in [5], it captures Dung’s preferred extensions and the weakly preferred semantics defined for bipolar systems. The constrained argumentation framework has been applied to practical reasoning in [1].

A crucial problem in argumentation is the derivability (or acceptance) of arguments: is a set of arguments included in at least one, or into every extension of an argumentation framework? In other words, is it credulously, or sceptically acceptable? An answer to the credulous acceptance problem for one argument has been proposed by, for example, [20,4] for Dung’s framework. Such an answer takes the form of a dialogue (or argument game) between two players, the one trying to show why the argument is acceptable, the other one in turn trying to show why it is not. Such dialectical proofs not only answer the problem, but also explain why the answer is so, in terms of arguments attacking and defending the argument on stage. Another kind of answer, based on a labeling of arguments, has been proposed by [19]. This approach builds minimal answers, but it does not exhibit dialogues which explain the answer as in the previous approaches. [7] has recently exhibited a unified framework which aims at capturing several dialectical approaches, among which the last two ones, but the approach is limited to Dung’s framework. In this paper, we propose an extension of [4]’s dialectical proofs to the credulous acceptance problem for a set of arguments, in a constrained argumentation framework.

Another important issue about argumentation is computation. Some programs exist (e.g. [24,19,25,26,9]), among which some of them compute a kind of dialectical proofs, different from [4]. Furthermore, none of them takes into account the additional notion of constraint presented here. In this paper, we present not only a formal model for these proofs, but also a program which computes them. This program is written using a high level language which makes it rather easy to read and understand. This language is Answer Set Programming (ASP), a paradigm which has already been applied with success in many areas of knowledge representation and reasoning, and for which efficient solvers are available [6,10]. [9] has recently applied ASP to the computation of extensions in Dung’s framework. The ASP program we propose here computes the dialectical proofs for credulous acceptance in a constrained argumentation framework.

The paper is organised as follows: we start with presenting Dung’s argumentation framework and constrained argumentation frameworks. Then we define the dialectical proofs for the constrained argumentation frameworks. Finally, we present ASP and the program which computes the proofs, before concluding.

1. Argumentation frameworks

This section starts with briefly presenting [8]’s argumentation framework (Def. 1 and 2).

**Definition 1** An argumentation framework is a pair \( \langle A, R \rangle \) where \( A \) is a set of so-called arguments and \( R \) is a binary relation over \( A \) (\( R \subseteq A \times A \)). Given two arguments \( a \) and \( b \), \( (a, b) \in R \) means that \( a \) attacks \( b \) (\( a \) is said to be an attacker of \( b \)).
An argumentation framework is nothing but a directed graph, whose vertices are the arguments and edges correspond to the elements of $R$. For computational reasons, we restrict the argumentation frameworks considered in this article to those whose set of arguments is finite.

**Example 1** Let $AF = \langle A, R \rangle$ be an argumentation framework with $A = \{a, b, c, d, e, f, g, h, i, j, k\}$ and $R = \{(a, b), (b, d), (d, i), (i, h), (a, c), (c, e), (e, f), (f, e), (f, g), (g, h), (j, k), (k, j)\}$. The graph for $AF$ is depicted on Figure 1.

![Graphical representation of AF](image)

**Definition 2** Let $AF = \langle A, R \rangle$ be an argumentation framework. $S \subseteq A$ is a conflict-free set of arguments if and only if for every $a, b \in S$, we have $(a, b) \notin R$. An argument $a \in A$ is acceptable w.r.t. a subset $S$ of $A$ if and only if for every $b \in A$ s.t. $(b, a) \in R$, there exists $c \in S$ s.t. $(c, b) \in R$. $S \subseteq A$ is admissible if and only if $S$ is conflict-free and every argument in $S$ is acceptable w.r.t. $S$. $S \subseteq A$ is a preferred extension if and only if it is maximal w.r.t. $\subseteq$ among the set of admissible sets.

**Example 2 (contd.)** $\{a, b\}$ is not a conflict-free set, $\{a, d\}$ is. $d$ is acceptable w.r.t. $\{a\}$. $\{a, d, c\}$ is admissible. $\{a, d, f, h, j\}$ and $\{a, d, e, g, j\}$ are two of the preferred extensions of $AF$.

Dung’s argumentation framework has been extended by [5] in order to take into account constraints over arguments (definitions 3, 4 and 5, and proposition 1).

**Definition 3** Let $PROP_{PS}$ be a propositional language defined in the usual inductive way from a set $PS$ of propositional symbols, the constants $\top$, $\bot$ and the connectives $\neg$, $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$. A Constrained Argumentation Framework (CAF) is a triple $CAF = \langle A, R, C \rangle$ where $A$ is a finite set of arguments, $R$ is a binary relation over $A$, the attack relation, and $C$ is a propositional formula from $PROP_A$.

**Example 3 (contd.)** The argumentation framework $AF$ of Figure 1 is extended with a constraint $C = (k \Leftrightarrow d) \land ((d \Rightarrow (f \lor g)) \lor (\neg d \Rightarrow \neg f))$. This constraint means, regarding the extensions, that argument $k$ belongs to an extension if and only if argument $d$ belongs to it, and that, if $d$ belongs to an extension, then either $f$ or $g$ must belong to it, or if $d$ does not belong to an extension, then $f$ must not belong to it either.

In the following definitions of this section, we consider that a constrained argumentation framework $CAF = \langle A, R, C \rangle$ is given. Furthermore, each subset $S$ of $A$ corresponds to an interpretation over $A$ (i.e. a total function from $A$ to $\{\text{true, false}\}$), given by the completion of $S$.
Definition 4 Let $S \subseteq A$. $S$ satisfies $C$ if and only if the completion $\hat{S} = \{ a \mid a \in S \} \cup \{ \neg a \mid a \in A \setminus S \}$ of $S$ is a model of $C$ (denoted by $\hat{S} \models C$).

The notions of admissibility and preferred extensions have been restricted so that the constraint $C$ is satisfied.

Definition 5 A subset $S$ of $A$ is $C$-admissible if and only if $S$ is admissible for $\langle A, R \rangle$ and satisfies $C$. A $C$-admissible set $S \subseteq A$ is a preferred $C$-extension if and only if it is maximal w.r.t. $\subseteq$ among the set of $C$-admissible sets.

Example 4 (contd.) The set $\{ a, d, e \}$ is admissible, but it is not $C$-admissible, because it does not satisfy the constraint $C$. $\{ a, e \}$ is $C$-admissible. $\{ a, d, e, g, k \}$ and $\{ a, e, g, j \}$ are two of the preferred $C$-extensions of CAF.

Obviously, a constrained argumentation framework $\langle A, R, C \rangle$ such that $C \equiv \top$ comes down to a Dung’s argumentation framework. Its preferred $C$-extensions are the preferred extensions of $\langle A, R \rangle$.

An important issue in argumentation is to be able to decide, given some definition of acceptability, that is, some semantics, which arguments can be derived from the framework. Usually, an argument is considered as derivable, hence acceptable, when it belongs to one extension (credulous consequence), or to every extension (sceptical consequence) under the semantics.

In this paper, we focus on the credulous derivability of a set of arguments $S$. Notice that two arguments may be individually derivable, while the set which contains both may not be included in any extension of the framework. Hence, considering the derivability problem for a set of arguments is more general than considering it for a single argument. Notice that it is always possible to come down to a single argument by considering the singleton which contains it. We will say that a set $S$ is credulously accepted under the $C$-preferred semantics if it is included in at least one preferred $C$-extension of CAF. To answer the credulous acceptance problem, that is, to determine whether a set of arguments is credulously accepted, the following result will be helpful.

Proposition 1 For each $C$-admissible set $X$ of CAF, there exists a preferred $C$-extension $E$ of CAF such that $X \subseteq E$.

Hence, determining whether a set of arguments is included into a preferred $C$-extension comes down to determine whether it is included into a $C$-admissible set.

From a computational point of view, [5] showed that the problem that consists in determining whether a set of arguments $S$ is credulously accepted is NP-complete. To answer this problem, they suggest to use a translation of the constrained argumentation framework into a formula in propositional logic which encodes the preferred $C$-extensions of the framework. This translation allows the use of SAT provers to answer the problem. However, such provers return a yes/no answer; they do not explain why the answer is so, what are the reasons that make the set of arguments credulously acceptable or not. This is a major drawback of this solution, since the argumentation process aims at explaining the answer in terms of pros and cons, by showing how the arguments on stage are attacked, how they are defended against these attacks, how they respect the constraint.
This process has been captured by [4] and subsequent works, in a dialectical framework for Dung’s preferred semantics without constraint.

In the next section, we extend the framework to constrained argumentation frameworks and to the preferred C-extensions, and then we will show how to compute them by using ASP.

2. Defining constrained dialectical proofs

This section starts with introducing a new general dialectical framework, adapted from [4], in the context of which a proof theory for the credulous acceptance problem under the C-preferred semantics is defined.

A dialectical proof is formalised by a dialogue between two players, PRO (the proponent) and OPP (the opponent). The dialogue takes place in a constrained argumentation framework. Its moves are governed by rules expressed in a so-called legal-move function.

Given a set \( A \), \( A^\ast \) denotes the set of finite sequences of elements from \( A \). For a syntactical purpose which will be explained later on, the set of arguments \( A \) of the framework is extended with an “empty” argument, denoted by \( \_ \). The set \( A \cup \{ \_ \} \) is denoted by \( A^{-} \).

**Definition 6** Let \( A \) be a set of arguments. A dialogue is a finite sequence \( d = \langle a_0, a_1, a_2, \ldots, a_n \rangle \) of arguments from \( A^{-} \). The player of \( a_i \), \( i \in \{0 \ldots n\} \), in \( d \) is PRO if \( i \) is even and is OPP if \( i \) is odd.

**Notation 1** Let \( d = \langle a_0, \ldots, a_i, \ldots, a_n \rangle \) be a dialogue. \( PRO(d) \) (resp. \( OPP(d) \)) denotes the set of arguments played by PRO (resp. OPP) in \( d \).

\( \forall i, 1 \leq i \leq n + 1 \), \( d_i = \langle a_0, \ldots, a_{i-1} \rangle \) is the length \( i \) prefix of \( d \). If \( i = 0 \), \( d_0 = \langle \_ \rangle \) is the empty dialogue. For a non empty dialogue \( d = \langle a_0 \ldots a_n \rangle \), \( a_n \) is denoted by \( last(d) \).

**Definition 7** Let \( \langle A, R, C \rangle \) be a constrained argumentation framework and \( \phi : A^{-\ast} \to 2^{A^{-}} \) a function called legal-move function. A \( \phi \)-dialogue for a set of arguments \( S \subseteq A \) is a dialogue \( d \) such that \( \forall i \geq 0, a_i \in \phi(d_i) \) and \( S \subseteq PRO(d) \).

\( \phi \) defines the moves of the dialogue; when it returns an empty set, it means that the dialogue cannot be continued.

We define now a specific legal-move function \( \phi_C \) in order to answer the credulous acceptance problem under the C-preferred semantics. Our proposal follows Proposition 1 that shows that determining whether a set of arguments \( S \) is included in at least one preferred C-extension comes down to find a C-admissible set in which \( S \) is included.

The set of arguments \( S \), the arguments that defend this set, and the ones necessary to satisfy the constraint, will be played by PRO; the attackers will be played by OPP. If the dialogue successfully shows that \( S \) is credulously accepted, then the set of arguments played by PRO is a model of the constraint: its arguments are the positive elements of the model, the arguments played by OPP are negative elements, and the rest of the arguments of \( A \) not played in the dialogue are also negative elements of the model. The legal-move function \( \phi_C \) is formally introduced in Definition 8. It is defined by respecting the following general principles.
The first move is played by PRO and follows the same rules as if it replies to an empty argument (see below). A move by OPP contains either an argument which attacks an argument played by PRO in a previous move, if such an argument exists (2), or the empty argument otherwise (1) to indicate that there is no argument in this case. The value of this empty argument is purely syntactical. This argument will be played each time OPP will be in such a case.

A move by PRO replies to the last move by OPP. As long as this move is not the empty argument, the argument played by PRO should attack OPP’s argument and should respect a number of constraints defined later in the set CPOSS(d) (6), in order to show that PRO can defend itself against the attacker exhibited by OPP. If there is no such argument, then it means that the set of arguments played so far by PRO cannot be included in any C-admissible set. The dialogue ends; the construction of a C-admissible set which contains S fails.

If OPP’s argument is the empty argument, then the set of arguments played so far by PRO is admissible. However, it is not guaranteed that it contains S. If it is not the case, PRO should go on playing the dialogue by moving an argument that belongs to S and to CPOSS(d) (3). If there is no such argument, then it means that S cannot be included in any C-admissible set. The dialogue ends, the construction of such a set fails.

Now if S ⊆ PRO(d), it is not guaranteed that PRO(d) satisfies the constraint (and hence is C-admissible). If it is not the case, PRO should go on playing the dialogue by moving an argument from CPOSS(d) (4). If there is no such argument, that is, if CPOSS(d) is empty, then PRO cannot continue the dialogue; the dialogue ends. The construction of a C-admissible set which contains S fails.

Now if OPP’s argument is the empty argument, if the constraint is satisfied and if S is included in PRO(d), then PRO does not have any more argument to play (5): the set of arguments played by PRO is C-admissible and contains S. The dialogue ends, the construction succeeds.

In order to present the set CPOSS(d) and the legal-move function φC, several notations are introduced:

Notation 2 Let ⟨A, R, C⟩ be a constrained argumentation framework. Let x ∈ A and S ⊆ A. Refl = {x ∈ A | (x, x) ∈ R}. R+(S) = {y ∈ A | ∃x ∈ S such that (x, y) ∈ R}. R+(S) = {y ∈ A | ∃x ∈ S such that (y, x) ∈ R}. R+(S) = R+(S) ∪ R+(S).

If a dialogue d would lead to a C-admissible set, then the set of arguments played by PRO should be conflict-free, that is, it should not contain any self-attacking argument (i.e. in Refl), nor any argument attacked or which attacks an argument already played by PRO (i.e. in R+(PRO(d))). Moreover, PRO should not have to repeat an argument it has already played (i.e. in PRO(d)). The set: POSS(d) = A \ (PRO(d) ∪ Refl ∪ R+(PRO(d))) formalizes these rules.

Moreover, we also want to play an argument that does not prevent us from satisfying the constraint. But, when we consider a C-admissible set, the semantics of C is given by evaluating as true all arguments in the set and as false all the other ones. But, during the building of a dialogue, this approach is quite too strong because the status (PRO or OPP) of some arguments (outside the dialogue) is not yet determined. That is why we introduce the Kleene’s three-valued logic in order to deal with C more precisely. Let us recall that a Kleene-interpretation is a mapping i from the propositional symbols to the
set of truth values \{f, t, u\} meaning false, true and undetermined. The truth tables are:

<table>
<thead>
<tr>
<th></th>
<th>~A</th>
<th>A &amp; B</th>
<th>A \lor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>t</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

and, as in classical logic, \(\iota(A \Rightarrow B) = \iota(\neg A \lor B)\).

**Proposition 2** Let \(\varphi\) be a formula, \(\{a_1, \ldots, a_n\}\) the set of propositional symbols occurring in \(\varphi\) and \(\iota\) a Kleene-interpretation such that \(\exists k \in \{1, \ldots, n\}, \iota(a_k) = u\).

Let \(k_0 \in \{1, \ldots, n\}\) such that \(\iota(a_{k_0}) = u\), we define \(\iota_f\) and \(\iota_t\) two Kleene-interpretations by:

\[
\begin{align*}
\iota_f(a_{i}) &= \iota_t(a_{i}) = \iota(a_{i}), \forall i \neq k_0, i \in \{1, \ldots, n\} \\
\iota_f(a_{k_0}) &= t \\
\iota_t(a_{k_0}) &= t
\end{align*}
\]

Then, we have:

\[
\begin{align*}
\iota_f(\varphi) &= f \Rightarrow \iota_f(\varphi) = \iota_t(\varphi) = f \\
\iota_t(\varphi) &= t \Rightarrow \iota_f(\varphi) = \iota_t(\varphi) = t
\end{align*}
\]

Let \(d\) be a \(\varphi\)-dialogue. It defines a Kleene-interpretation \(\iota_d\):

\[
\begin{align*}
\iota_d(x) &= t \Leftrightarrow x \in \text{PRO}(d) \\
\iota_d(x) &= f \Leftrightarrow x \in \text{OPP}(d) \\
\iota_d(x) &= u \text{ otherwise}
\end{align*}
\]

By proposition 2, if for a given \(\varphi\)-dialogue \(d\) the constraint \(C\) is satisfied, \(\iota_d(C) = t\), (resp. not satisfied, \(\iota_d(C) = f\)) then \(C\) is satisfied (resp. not satisfied) by \(\iota_d\) for every \(\varphi\)-dialogue \(d'\) extending \(d\).

Let \(d\) be a \(\phi_C\)-dialogue whose last move is by OPP. When we want to extend it with a PRO argument, we have to play some \(x\) from \(\text{POSS}(d)\) that does not prevent us from satisfying the constraint\(^1\), i.e. \(\iota_d(x)(C) \neq f\).

To sum up, the set:

\[
\text{CPOSS}(d) = \text{POSS}(d) \cap \{x \mid \iota_d(x)(C) \neq f\}
\]

expresses all these elements. Note that, for an even-length dialogue \(d\), \(\text{CPOSS}(d) = \emptyset\) means that \(\text{PRO}(d)\) cannot be extended into a larger \(C\)-admissible set.

The following legal-move function is going to be used to define the dialogues for the credulous acceptance problem.

**Definition 8** Given a constrained argumentation framework \(\langle A, R, C\rangle\), a set \(S \subseteq A\), let \(\phi_C : A^+ \rightarrow 2^A\) be the function defined by:

- if \(d\) is an odd-length dialogue (next move is by OPP),
  \[
  \phi_C(d) = \begin{cases} 
  \bot & \text{if } R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d)) = \emptyset \quad (1) \\
  R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d)) & \text{otherwise} \quad (2)
  \end{cases}
  \]

- if \(d\) is an even-length dialogue (next move is by PRO),
  \[
  \phi_C(d) = \begin{cases} 
  \text{If } d = \emptyset \text{ or last}(d) = \bot \text{ then if } S \subseteq \text{PRO}(d) \\
  \text{then } \text{CPOSS}(d) \cap S \\
  \text{else if } \text{PRO}(d) \not\subseteq C \text{ then } \text{CPOSS}(d) \text{ (4)} \\
  \text{else } \emptyset \\
  \text{else } \text{CPOSS}(d) \cap R^-(\text{last}(d)) \quad (6)
  \end{cases}
  \]

\(^1\)\(d(x)\) denotes the dialogue \(d\) extended with argument \(x\).
Definition 9 Let \((A, R, C)\) be a constrained argumentation framework and \(S \subseteq A\) be a set of arguments. A \(\phi_C\)-proof for \(S\) is a \(\phi_C\)-dialogue \(d\) for \(S\) such that 
\[
(d = \emptyset) \lor \text{last}(d) = \_ \land \text{PRO}(d) = C.
\]
We say that \(d\) is won by PRO.

The following result\(^2\) establishes the correctness and the completeness of \(\phi_C\)-proofs.

Proposition 3 Let \(CAF = \langle A, R, C \rangle\) be a constrained argumentation framework. If \(d\) is a \(\phi_C\)-proof for a set of arguments \(S \subseteq A\), then PRO\((d)\) is a \(C\)-admissible set of \(CAF\) that contains \(S\). If a set of arguments \(S\) is included in a \(C\)-admissible set of \(CAF\) then there exists a \(\phi_C\)-proof for \(S\).

Notice that in the case where the constraint is a tautology, the dialectical proofs come down to the dialectical proofs defined in [4].

Example 5 (contd.) Consider the set \(S = \{e, k\}\). A \(\phi_C\)-dialogue for \(S\) is:
\[
\begin{align*}
d_0 & = \emptyset, \phi_C(d_0) = \{e, k\}; \\
a_0 & = e, d_1 = \langle e \rangle, \phi_C(d_1) = \{e\}; \\
a_1 & = e, d_2 = \langle e.e \rangle, \phi_C(d_2) = \{a\}; \\
a_2 & = a, d_3 = \langle e.a.a \rangle, \phi_C(d_3) = \{\_\}; \\
a_3 & = a, d_4 = \langle e.a.a.a \rangle, S \nsubseteq \text{PRO}(d_4), \phi_C(d_4) = \{k\}; \\
a_4 & = k, d_5 = \langle e.a.a.a.k \rangle, \phi_C(d_5) = \{\_\}; \\
a_5 & = k, d_6 = \langle e.a.a.a.k,_{\_} \rangle, S \subseteq \text{PRO}(d_6), \text{PRO}(d_6) \nsubseteq C, \phi_C(d_6) = \{d, i, h, g\}; \\
a_6 & = g, d_7 = \langle e.a.a.a.k,_{\_}g \rangle, \phi_C(d_7) = \{\_\}; \\
a_7 & = g, d_8 = \langle e.a.a.a.k,_{\_}g,_{\_}g \rangle, S \subseteq \text{PRO}(d_8), \text{PRO}(d_8) \nsubseteq C, \phi_C(d_8) = \{d, i\}; \\
a_8 & = d, d_9 = \langle e.a.a.a.k,_{\_}g,_{\_}d \rangle, \phi_C(d_9) = \{\_\}; \\
a_9 & = d, d_{10} = \langle e.a.a.a.k,_{\_}g,_{\_}d,_{\_} \rangle, S \subseteq \text{PRO}(d_{10}), \text{PRO}(d_{10}) \nsubseteq C, \phi_C(d_{10}) = \emptyset; \\
d_{10} & is a \phi_C\)-dialogue won by PRO. \text{PRO}(d_{10}) = \{e, a, k, g, d\} is a \(C\)-admissible set. Thus \(\{e, k\}\) is included into at least one preferred \(C\)-extension; it is then credulously accepted.
\]

3. Computing constrained dialectical proofs by ASP

Answer Set Programming (ASP) is a declarative paradigm having its root in the stable model semantics [12] for normal logic programs. It has already been applied with success in many areas of knowledge representation and reasoning in Artificial Intelligence (default reasoning, semantic web, planification, causal reasoning,...). Furthermore, because of the availability of efficient solvers as Clasp [11], Smodels [17] or DLV [14], ASP is also a very good framework to encode and solve combinatorial problems coming from various domains (graph theory, configuration, bio informatics,...). We invite the reader to consult [6] and [10] for a recent, theoretical and practical overview of ASP and we recall in the following the main theoretical notions that we use in this work.

A normal logic program is a set of rules like

\[
\begin{align*}
c & \leftarrow a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m, n \geq 0, m \geq 0,
\end{align*}
\]

\(^2\)The proof of Proposition 3 can be downloaded at http://www.info.univ-angers.fr/pub/ claire/asperix/Argumentation
where \( c, a_1, \ldots, a_n, b_1, \ldots, b_m \) are ground atoms and \( \text{not} \) represents a default negation. For a rule \( r \) (or by extension for a rule set), we note \( \text{head}(r) = c \) its head, \( \text{body}^+(r) = \{a_1, \ldots, a_n\} \) its positive body, \( \text{body}^-(r) = \{b_1, \ldots, b_m\} \) its negative body.

The Gelfond-Lifschitz reduct of a program \( P \) by an atom set \( X \) is the program \( P^X = \{ \text{head}(r) ← \text{body}^+(r). \mid \text{body}^-(r) ∩ X = \emptyset \} \). Since it has no default negation, such a program is definite and then it has a unique minimal Herbrand model denoted with \( Cn(P) \). By definition, an answer set (originally called a stable model [12]) of \( P \) is an atom set \( S \) such that \( S = Cn(P^S) \).

As usual in ASP, a rule with an empty body \( x ← . \) is called a fact and is simplified as \( x \). And a headless rule is called a constraint. For instance, \( ← x, \text{not} y, \text{not} \text{new}.. \) is a shortcut for the rule \( \text{new} ← x, \text{not} y, \text{not} \text{new}.. \), where \( \text{new} \) is an atom appearing nowhere else in the program. Such a rule forbids to any atom set containing \( x \) and not containing \( y \) to be an answer set of the program. The use of constraints in a program illustrates the declarative nature of ASP. We just have to describe the situations that we reject in order to exclude them from the potential solutions.

As it is the case in our work, many problems are firstly encoded in ASP by means of a first order logic program containing atoms like \( p(X, 3, f(Y)) \). Since answer set definition is given for propositional programs, \( P \) has to be seen as an intensional version of the propositional program \( \text{ground}(P) \) defined as follows. Given a rule \( r \), \( \text{ground}(r) \) is the set of all fully instantiated rules that can be obtained by substituting every variable in \( r \) by every constant of the Herbrand universe of \( P \) and then, \( \text{ground}(P) = \bigcup_{r ∈ P} \text{ground}(r) \).

The relationships between argumentation frameworks and ASP have been studied in some previous works. For instance, considering that \( \text{att}(X, Y) \) stands for “argument \( X \) attacks argument \( Y \)”, \( \text{acc}(X) \) for “argument \( X \) is acceptable” and \( \text{def}(X) \) for “argument \( X \) is defeated”, [8] shows that the program

\[
\{ \text{att}(a_1, a_2), \ldots, \text{att}(a_m, a_n), \\
\text{acc}(X) ← \text{not} \text{def}(X), \\
\text{def}(X) ← \text{att}(Y, X), \text{acc}(Y) \}
\]

has a stable model iff the corresponding argumentation framework has a stable extension. Moreover, the atoms \( \text{acc}(X) \) occurring in the stable model represent the set of acceptable arguments. Pursuing this line, [9] presents an ASP system able to compute the extensions of a Dung’s argumentation framework under various notions of acceptability.

But, up to our knowledge no system has already been developed in order to answer the credulous acceptance problem with constraints. Furthermore, one goal of the framework of dialectical proofs is to exhibit explicitly the argumentative fight between pros and cons arguments and again this point has not already been studied via ASP.

In our present work, ASP is also used to deal with the constraint of the argumentation framework. The links between classical logic and ASP as already been studied in [15, 18] whose goals are to compute (if it exists) a model of a formula by means of a normal logic program. In our setting, given an interpretation, we are concerned by the computation of the truth-value (in classical logic or in 3-valued logic) of a formula. Clearly this is a polynomial problem when the previous one, computing a model, is an NP-hard one.

Because of lack of space we give only the main lines of the program that we have written in order to compute the dialectical proofs of a given constrained argumentation framework. The whole program is available at [Link]
We mention that our program contains symbolic functions, in particular lists, and then only two ASP solvers, ASPeRiX [13] and DLV-complex [3], can be used to compute its answer sets.

The ASP program consists of facts encoding an argumentation system and of rules encoding what a $\phi_C$-proof is: the legal-move function, POSS and CPOSS sets... The models of the program (the answer sets), if there are, are all the $\phi_C$-proofs of the given argumentation system.

The argumentation system is represented via predicates $\text{argu}(X)$ for the arguments, and $\text{att}(X,Y)$ for attacks. The set $S$, for which we want to prove its acceptability, is represented by extension of predicate $\text{inS}(X)$, and the constraint by $\text{constraint}(C)$. The constraint is expressed by a functional term where the logic constants $\top$ and $\bot$ are true and false, the logical connectives $\neg$, $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$ are unary functor $\neg$ and binary functors $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$. The argumentation system from the example developed in previous sections is encoded by the following facts.

\begin{verbatim}
argu(a). argu(b). argu(c). ... argu(k).
att(a,b). att(a,c). att(b,d). ... att(k,j).
inS(e). inS(k).
constraint(and(or(imp(d, or(f,g)), impl(neg(d),neg(f))), equ(k,d))).
\end{verbatim}

A dialectical proof is represented by a set of atoms of the form $\text{rank}(X,N)$ meaning that argument $X$ is played at rank $N$ in the dialogue. Term $\text{empty}$ encodes the special empty argument _. For instance, the dialogue $\langle e.c.a._ \rangle$ is represented by the set of atoms \{$\text{rank}(e,0)$, $\text{rank}(c,1)$, $\text{rank}(a,2)$, $\text{rank}(empty,3)$\}.

The dialogue can be seen as built incrementally, one rank after another. When several arguments can be played at a rank, one of them is chosen in a non deterministic way. Predicate $\text{nrank}$ prevents to play more than one argument at a rank: when an argument is chosen for rank $N$, all other are prohibited for the same rank. The construction of the dialogue begins by choosing an argument for rank 0. Then the dialogue continues until no move can be played at a step. If some final conditions are realized, a model of the program has been built, which represents a dialogue, and which is the output of the solver. Otherwise, the solver tries to build another dialogue.

The legal-move function $\phi_C$ is represented in the program by rules that define arguments that can be played at each rank of the dialogue. For instance, rules for cases (1), (2) and (6) of Definition 8 are given below. In these rules, $\text{strongAttack}(X,N)$ means that $X$ attacks the proof but is not attacked by the proof, that is $X \in R_-(\text{PRO}(d)) \setminus R_+(\text{PRO}(d))$ and $\text{imposs}(X,N)$ represents the complementary set of $\text{POSS}(d)$.

\begin{verbatim}
% X is played by PRO, next move is played by OPP (1)
rank(empty,N+1) :- rank(X,N), N mod 2==0, not weak(N).
weak(N) :- strongAttack(X,N).
% X is played by PRO, next move is played by OPP (2)
rank(Y,N+1) :- rank(X,N), N mod 2==0, strongAttack(Y,N), not nrank(Y,N+1).
...
% X is played by OPP, next move is played by PRO (6)
rank(Y,N+1) :- rank(X,N), X!=empty, N mod 2==1, att(Y,X),
    not imposs(Y,N+1), not nrank(Y,N+1).
% only one argument is played at a rank
nrank(Y,N) :- rank(X,N), argu(Y), X!=Y.
nrank(empty,N) :- rank(X,N), X!=empty.
\end{verbatim}
But this program builds too many dialogues: some of them do not correspond to $C$-admissible sets and thus must be rejected. First, when PRO is playing, he has to play an argument from $\text{CPOSS}(d)$ and not only from $\text{POSS}(d)$ as we do. So we have to discard the dialogues for which a PRO move makes the constraint false. Second, a dialogue is a successful proof only if some final conditions are realized: the dialogue ends with the empty argument (unless the dialogue is empty), the set $S$ is included in PRO’s arguments, and the constraint is satisfied. The “wrong” dialogues are excluded from the models of the program by adding some constraints (headless rules) that prohibit them. In these rules, $\text{false}_3(C,N)$ means that propositional formula $C$ is false in the 3-valued interpretation defined by the dialogue at rank $N$, i.e. $\iota_d(C) = f$. $\text{non_satisfied}_C(N)$ means that the constraint is not satisfied in the 2-valued interpretation defined by the dialogue at rank $N$, i.e. $\widehat{\text{PRO}}(d) \not\models C$. And $\text{non_included}_S(N)$ means that $S \not\subseteq \text{PRO}(d)$ at rank $N$ of the dialogue.

Recall that the constraint is represented as a functional term. The truth values, in 2 and 3-valued logics, of arguments are inferred from the part of the dialogue already built. Then a simple propagation suffices to deduce the value of the whole constraint.

4. Conclusion

In this paper, we propose a new notion of dialectical proofs for constrained argumentation frameworks. These proofs allow to verify the credulous acceptability of a set of arguments under the $C$-preferred semantics. Such proofs explain why and how the set of arguments is inferred. Thus, it is very easy and natural to associate to the theoretical framework an effective computation of the proof. This last part is realized in ASP. These two contributions can be used to develop any automated reasoning system based on the $C$-preferred semantics of a constrained argumentation framework (e.g., as pointed out in [5], Dung’s preferred semantics, bipolar systems’ weakly preferred semantics...).

Another conclusion to draw is that, once again, ASP has proved its very good flexibility to represent and solve a reasoning problem. On one side, we see that the notion of rules is very well adapted to encode the iterating and alternating roles of pros and cons in the building of the proof. On another side, the ASP constraints are a very convenient way to discard unwanted potential solutions.

As a future work, we plan to compare the recent approach by [28] with the one presented in this paper.
References