Argumentation Frameworks as Constraint Satisfaction Problems

Leila Amgoud[1] and Caroline Devred[2]

[1] IRIT, University of Toulouse, France
Leila.Amgoud@irit.fr
[2] LERIA, University of Angers, France
devred@info.univ-angers.fr

Abstract. This paper studies how to encode the problem of computing the extensions of an argumentation framework (under a given semantics) as a constraint satisfaction problem (CSP). Such encoding is of great importance since it makes it possible to use the very efficient solvers (developed by the CSP community) for computing the extensions. We focus on three families of frameworks: Dung's abstract framework, its constrained version and preference-based argumentation frameworks.

Keywords: Default Reasoning, Argumentation, CSP.

1   Introduction

Argumentation is a reasoning model based on the construction and evaluation of interacting arguments. An argument is a reason for believing in a statement, doing an action, pursuing a goal, etc.

Argumentation theory is gaining an increasing interest in Artificial Intelligence, namely for reasoning about defeasible/uncertain information, making decisions under uncertainty, learning concepts, and modeling agents' interactions (see [1]).

The most abstract argumentation framework has been proposed in the seminal paper [15] by Dung. It consists of a set of *arguments*, a *binary relation* representing *attacks* among arguments, and *semantics* for evaluating the arguments. A semantics describes when a set of arguments, called *extension*, is acceptable without bothering on how to compute that set. This framework has been extended in different ways in the literature. In [2,3], arguments are assumed to not have the same strength while in [9] an additional constraint on arguments may be available. In both works, Dung's semantics are used to evaluate arguments, thus to compute the extensions.

In [9,12,13,17], different decision problems related to the implementation of those semantics have been identified and the computational complexity of each problem studied. The results are a bit disappointing since they show that the most important decision problems (like for instance testing whether a framework has a stable set of arguments) are costly. Some algorithms that compute extensions under some semantics have been developed, for instance in [8,11,18].

However, the efficiency of those algorithms was not proved. They are neither tested on benchmarks nor compared to other algorithms developed for the same purpose.

Besides, there is a huge literature on *Constraints Satisfaction Problems* (CSP) since many real-world problems can be described as CSPs. A CSP consists of a set of *variables*, a (generally finite) *domain* for each variable and a set of *constraints*. Each constraint is defined over a subset of variables and limits the combination of values that the variables in this subset can take. The goal is to find an assignment to the variables which satisfies all the constraints. In some problems, the goal is to find all such assignments. Solving a constraint satisfaction problem on a finite domain is an NP-complete problem in general. In order to be solved in a reasonable time, different *solvers* have been developed. They use a form of search based on variants of backtracking, constraint propagation and local search [19].

Our aim is to be able to use those powerful solvers for computing the extensions of an argumentation framework. For that purpose, we study in this paper how to encode an argumentation framework as a CSP. We particularly focus on three families of frameworks: Dung's framework, constrained argumentation framework and preference-based argumentation framework (where arguments may have different strengths). For each family, we propose different CSPs which compute the extensions of the framework under different acceptability semantics. In each CSP, arguments play the role of variables and the attacks represent mainly the constraints.

This paper is organized as follows: Section 2 recalls the basic concepts of a CSP. Section 3 recalls Dung's argumentation framework and shows how it is encoded as a CSP. Section 4 recalls the constrained version of Dung's framework and presents its encoding as a CSP. Section 5 presents preference-based argumentation frameworks as well as their encoding as CSPs. In Section 6, we compare our approach to existing works on the topic. The last section is devoted to concluding remarks and perspectives. Due to space limitation, the proofs are not included in the paper.

2   Constraint Satisfaction Problems (CSPs)

Formally speaking, a *constraint satisfaction problem* (or CSP) is defined by a set of *variables*, $x_1, \ldots, x_n$, and a set of *constraints* $c_1, \ldots, c_m$. Each variable x_i takes its values from a *finite domain* $\mathcal{D}_i$, and each constraint c_i involves some subset of the variables and specifies the allowable combinations of values for that subset.

Definition 1 (CSP). *A* CSP *instance is a triple* $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ *where:*

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ *is a set of variables,*
- $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ *is a set of finite domains for the variables, and*
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ *is a set of constraints.*

Each constraint c_i is a pair (h_i, H_i) where

- $h_i = (x_{i1}, \ldots, x_{ik})$ *is a k-tuple of variables*
- H_i *is a k-ary relation over $\mathcal{D}$, i.e. H_i is a subset of all possible variable values representing the allowed combinations of simultaneous values for the variables in h_i.*

A state of the problem is defined by an *assignment* of values to some or all of the variables.

Definition 2 (Assignment). *An* assignment *v for a CSP instance $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is a mapping that assigns to every variable $x_i \in \mathcal{X}$ an element $v(x_i) \in \mathcal{D}_i$. An assignment v satisfies a constraint $((x_{i1}, \ldots, x_{ik}), H_i) \in \mathcal{C}$ iff $(v(x_{i1}), \ldots, v(x_{ik})) \in H_i$.*

Finally, a solution of a CSP is defined as follows:

Definition 3 (Solution). *A* solution *of a CSP instance $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ is an assignment v that satisfies all the constraints in $\mathcal{C}$ and in which all the variables of $\mathcal{X}$ are assigned a value. We write $(v(x_1), \ldots, v(x_n))$ to denote the solution.*

3   Abstract Frameworks

This section recalls Dung's argumentation framework and presents the different corresponding CSPs which return its extensions under various semantics.

3.1   Dung's Framework

In [15], Dung has developed the most abstract argumentation framework in the literature. It consists of a set of arguments and an attack relation between them.

Definition 4 (Argumentation framework). *An* argumentation framework *(AF) is a pair $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$). The notations $a\mathcal{R}b$ or $(a, b) \in \mathcal{R}$ mean that the argument a attacks the argument b.*

Different *acceptability semantics* for evaluating arguments have been proposed in the same paper [15]. Each semantics amounts to define sets of acceptable arguments, called *extensions*. Before recalling those semantics, let us first introduce the two basic properties underlying them, namely *conflict-freeness* and *defence*.

Definition 5 (Conflict-free, Defence). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an AF and $\mathcal{B} \subseteq \mathcal{A}$.*

- *$\mathcal{B}$ is* conflict-free *iff $\nexists\, a, b \in \mathcal{B}$ s.t. $a\mathcal{R}b$.*
- *$\mathcal{B}$ defends* an argument a iff for all $b \in \mathcal{A}$ s.t. $b\mathcal{R}a$, there exists $c \in \mathcal{B}$ s.t. $c\mathcal{R}b$.*

The following definition recalls the acceptability semantics proposed in [15].

Definition 6 (Acceptability semantics). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an AF and $\mathcal{B} \subseteq \mathcal{A}$.*

- *$\mathcal{B}$ is an* admissible *set iff it is conflict-free and defends its elements.*
- *$\mathcal{B}$ is a* preferred *extension iff it is a maximal (for set $\subseteq$) admissible set.*
- *$\mathcal{B}$ is a* stable *extension iff it is a preferred extension that attacks any argument in $\mathcal{A} \setminus \mathcal{B}$.*
- *$\mathcal{B}$ is a* complete *extension iff it is conflict-free and it contains all the arguments it defends.*
- *$\mathcal{B}$ is a* grounded *extension iff it is a minimal (for set $\subseteq$) complete extension.*

Example 1. Let us consider the framework $\mathcal{F}_1 = (\mathcal{A}_1, \mathcal{R}_1)$ where $\mathcal{A}_1 = \{a, b, c, d\}$ and $\mathcal{R}_1 = \{(a, b), (b, c), (c, d), (d, a)\}$. $\mathcal{F}_1$ has two preferred and stable extensions: $\mathcal{B}_1 = \{a, c\}$ and $\mathcal{B}_2 = \{b, d\}$ while its grounded extension is the empty set.

3.2 Computing Dung's Semantics by CSPs

In this section, we propose four mappings of Dung's argumentation framework into CSP instances. The idea is: starting from an argumentation framework, we define a CSP instance whose solutions are the extensions of the framework under a given acceptability semantics. In the four instances, arguments play the role of variables that is, a variable is associated to each argument. Each variable may take two values 0 or 1 meaning that the corresponding argument is rejected or accepted. Thus, the domains of the variables are all *binary*. Things are different with the constraints. We show that according to the semantics that is studied, the definition of a constraint changes.

Let us start with a CSP instance that computes the conflict-free sets of arguments. In this case, each attack $(a, b) \in \mathcal{R}$ gives birth to a constraint which says that the two variables a and b cannot take value 1 at the same time. This means that the two corresponding arguments cannot belong to the same set. This constraint has the following form: $((a, b), ((0, 0), (0, 1), (1, 0)))$. Note that this is equivalent to the cases where the propositional formula $a \Rightarrow \neg b$ is true (i.e. gets value 1). For simplicity reasons, throughout the paper we will use propositional formulas for encoding constraints. Solving a CSP amounts thus to finding the models of the set of constraints.

Definition 7 (Free CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework. A free CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{a \Rightarrow \neg b \mid (b, a) \in \mathcal{R}\}$.*

It can be checked that $|\mathcal{C}| = |\mathcal{R}|$. The following result shows that the solutions of this CSP are the conflict-free sets of arguments of the corresponding AF.

Theorem 1. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be the CSP instance associated with the AF $\mathcal{F} = (\mathcal{A}, \mathcal{R})$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is conflict-free (with $i = j \ldots, k$).*

Let us consider the argumentation framework $\mathcal{F}_1$ defined in Example 1.

Example 1 (Cont):   The CSP corresponding to $\mathcal{F}_1$ is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ s.t. $\mathcal{X} = \{a, b, c, d\}$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$, $\mathcal{C} = \{a \Rightarrow \neg d, b \Rightarrow \neg a, c \Rightarrow \neg b, d \Rightarrow \neg c\}$. This CSP has the following solutions: $(0, 0, 0, 0)$, $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, $(0, 0, 1, 0)$, $(0, 0, 0, 1)$, $(1, 0, 1, 0)$ and $(0, 1, 0, 1)$. Thus, the sets $\{\}$, $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{a, c\}$ and $\{b, d\}$ are conflict-free.

Let us now study the case of stable semantics. Stable extensions are computed by a CSP which considers that an argument and its attackers cannot have the same value.

Definition 8 (Stable CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework. A stable CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, $\forall a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{a \Leftrightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} \neg b \mid a \in \mathcal{A}\}$.*

It is worth mentioning that the previous definition is inspired from [10].

Theorem 2. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a stable CSP associated with $\mathcal{F} = (\mathcal{A}, \mathcal{R})$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is a stable extension of $\mathcal{F}$.*

Let us illustrate this result on the following example.

Example 1 (Cont):   The stable CSP corresponding to $\mathcal{F}_1$ is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ s.t. $\mathcal{X} = \{a, b, c, d\}$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$, and $\mathcal{C} = \{a \Leftrightarrow \neg d, b \Leftrightarrow \neg a, c \Leftrightarrow \neg b, d \Leftrightarrow \neg c\}$. This CSP has two solutions: $(1, 0, 1, 0)$ and $(0, 1, 0, 1)$. The sets $\{a, c\}$ and $\{b, d\}$ are the two stable extensions of $\mathcal{F}_1$.

The two previous CSPs are simple since attacks are directly transformed into constraints. The notion of defence is not needed in both cases. However, things are not so obvious with admissible semantics. The following definition shows that a CSP which computes the admissible sets of an AF should consider both the attacks and the defence in its constraints.

Definition 9 (Admissible CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework. An admissible CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{(a \Rightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} \neg b) \wedge (a \Rightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} (\bigvee_{c:(c,b) \in \mathcal{R}} c)) \mid a \in \mathcal{A}\}$.*

The following result shows that the solutions of an admissible CSP provide the admissible extensions of the corresponding argumentation framework.

Theorem 3. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be an admissible CSP associated with an AF $\mathcal{F}$. $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is an admissible set of $\mathcal{F}$.*

Let us illustrate the notion of admissible CSP with a simple example.

Example 2. Let us consider the framework $\mathcal{F}_2 = (\mathcal{A}_2, \mathcal{R}_2)$ where $\mathcal{A}_2 = \{a, b, c, d\}$ and $\mathcal{R}_2 = \{(c, b), (d, b), (b, a)\}$. The admissible CSP associated with $\mathcal{F}_2$ is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where: $\mathcal{X} = \mathcal{A}_2$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$ and $\mathcal{C} = \{d \Rightarrow \top, c \Rightarrow \top, b \Rightarrow \neg c \wedge \neg d, a \Rightarrow \neg b, a \Rightarrow c \vee d\}$. This CSP has the following solutions: $(0, 0, 0, 0)$, $(0, 0, 1, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 1)$ $(1, 0, 1, 0)$, $(1, 0, 0, 1)$, $(1, 0, 1, 1)$. These solutions return the admissible sets of $\mathcal{F}_2$, that is: $\{\}$, $\{c\}$, $\{d\}$, $\{c, d\}$, $\{a, c\}$, $\{a, d\}$ and $\{a, c, d\}$.

As preferred extensions are maximal (for set inclusion) admissible sets, then they are computed by an admissible CSP.

Theorem 4. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be an admissible CSP associated with an AF $\mathcal{F}$. Each maximal (for set inclusion) set $\{x_j, \ldots, x_k\}$, s.t. $v(x_i) = 1$ and $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP, is a preferred extension of $\mathcal{F}$.*

Let us come back to Example 2.

Example 2 (Cont): It is clear that the last solution $(1, 0, 1, 1)$ is the one which returns the only preferred extension of $\mathcal{F}_2$, i.e. $\{a, c, d\}$.

Complete extensions are also computed by a CSP which takes into account the notion of defence in the constraints.

Definition 10 (Complete CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework. A* complete CSP *associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{(a \Rightarrow \bigwedge\limits_{b:(b,a)\in\mathcal{R}} \neg b) \wedge (a \Leftrightarrow \bigwedge\limits_{b:(b,a)\in\mathcal{R}} (\bigvee\limits_{c:(c,b)\in\mathcal{R}} c)) \mid a \in \mathcal{A}\}.$*

Note that there is a slight difference between the constraints of an admissible CSP and those of a complete CSP. Since a complete extension should contain all the arguments it defends, then an argument and all its defenders should be in the same set. However, the only requirement on an admissible set is that it defends its arguments. This is encoded by a simple logical implication.

Theorem 5. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a complete CSP associated with an AF $\mathcal{F}$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$, s.t. $v(x_i) = 1$ is a complete extension of $\mathcal{F}$.*

Example 2 (Cont): The complete CSP associated with $\mathcal{F}_2$ is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where: $\mathcal{X} = \mathcal{A}_2$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$ and $\mathcal{C} = \{d \Rightarrow \top, c \Rightarrow \top, b \Rightarrow \neg c \wedge \neg d, a \Rightarrow \neg b, a \Leftrightarrow c \vee d\}$. This CSP has one solution which is $(1, 0, 1, 1)$. Thus, $\mathcal{F}_2$ has the set $\{a, c, d\}$ as its unique complete extension.

Since grounded extension is a minimal (for set inclusion) complete extension, then it is computed by a complete CSP as follows.

Theorem 6. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a complete CSP associated with an AF $\mathcal{F}$. The grounded extension of $\mathcal{F}$ is the minimal (for set inclusion) set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ and $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP.*

Example 2 (Cont): The grounded extension of $\mathcal{F}_2$ is $\{a, c, d\}$ which is returned by the unique solution of the complete CSP corresponding to $\mathcal{F}_2$.

4   Constrained Frameworks

This section recalls the constrained version of Dung's argumentation framework and proposes its mappings to CSPs.

4.1   Basic Definitions

The basic argumentation framework of Dung has been extended in [9] by adding a *constraint* on arguments. This constraint should be satisfied by Dung's extensions (under a given semantics). For instance, in Example 1, one may imagine a constraint which requires that the two arguments a and c belong to the same extension. Note that this constraint is satisfied by $\mathcal{B}_1$ but not by $\mathcal{B}_2$. Thus, $\mathcal{B}_1$ would be the only extension of the framework.

The constraint is a formula of a propositional language $\mathcal{L}_\mathcal{A}$ whose alphabet is exactly the set $\mathcal{A}$ of arguments. Thus, each argument in $\mathcal{A}$ is a literal of $\mathcal{L}_\mathcal{A}$. $\mathcal{L}_\mathcal{A}$ contains all the formulas that can be built using the usual logical operators ($\wedge$, $\vee$, $\Rightarrow$, $\neg$, $\Leftrightarrow$) and the constant symbols ($\top$ and $\bot$).

Definition 11 (Constraint, Completion). *Let $\mathcal{A}$ be a set of arguments and $\mathcal{L}_\mathcal{A}$ its corresponding propositional language.*

- *$\mathcal{C}$ is a constraint on $\mathcal{A}$ iff $\mathcal{C}$ is a formula of $\mathcal{L}_\mathcal{A}$.*
- *The completion of a set $\mathcal{B} \subseteq \mathcal{A}$ is $\widehat{\mathcal{B}} = \{a \mid a \in \mathcal{B}\} \cup \{\neg a \mid a \in \mathcal{A} \setminus \mathcal{B}\}$.*
- *A set $\mathcal{B} \subseteq \mathcal{A}$ satisfies $\mathcal{C}$ iff $\widehat{\mathcal{B}}$ is a model of $\mathcal{C}$ ($\widehat{\mathcal{B}} \models \mathcal{C}$).*

The completion of a set $\mathcal{B}$ of arguments is a set in which each argument of $\mathcal{A}$ appears either as a positive literal if the argument belongs to $\mathcal{B}$ or as a negative one otherwise. Thus, $|\widehat{\mathcal{B}}| = |\mathcal{A}|$.

A constrained argumentation framework (CAF) is defined as follows:

Definition 12 (CAF). *A constrained argumentation framework (CAF) is a triple $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathcal{C})$ where $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is an attack relation and $\mathcal{C}$ is a constraint on the set $\mathcal{A}$.*

Dung's semantics are extended to the case of CAFs. The idea is to compute Dung's extensions (under a given semantics), and to keep among those extensions only the ones that satisfy the constraint $\mathcal{C}$.

Definition 13 (*C*-admissible set). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a CAF and $\mathcal{B} \subseteq \mathcal{A}$. The set $\mathcal{B}$ is $\mathscr{C}$-admissible in $\mathcal{F}$ iff:*

1. *$\mathcal{B}$ is admissible,*
2. *$\mathcal{B}$ satisfies the constraint $\mathscr{C}$.*

In [15], it has been shown that the empty set is always admissible. However, it is not always $\mathscr{C}$-admissible since the set $\widetilde{\varnothing}$ does not always imply $\mathscr{C}$.

Definition 14 (*C*-preferred, *C*-stable extension). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a CAF and $\mathcal{B} \subseteq \mathcal{A}$.*

– *$\mathcal{B}$ is a $\mathscr{C}$-preferred extension of $\mathcal{F}$ iff $\mathcal{B}$ is maximal for set-inclusion among the $\mathscr{C}$-admissible sets.*
– *$\mathcal{B}$ is a $\mathscr{C}$-stable extension of $\mathcal{F}$ iff $\mathcal{B}$ is a $\mathscr{C}$-preferred extension that attacks all arguments in $\mathcal{A} \backslash \mathcal{B}$.*

The following result summarizes the links between the extensions of a CAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ and those of its basic version $\mathcal{F}' = (\mathcal{A}, \mathcal{R})$.

Theorem 7. *[9] Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a CAF and $\mathcal{F}' = (\mathcal{A}, \mathcal{R})$ be its basic version.*

– *For each $\mathscr{C}$-preferred extension $\mathcal{B}$ of $\mathcal{F}$, there exists a preferred extension $\mathcal{B}'$ of $\mathcal{F}'$ s.t. $\mathcal{B} \subseteq \mathcal{B}'$.*
– *Every $\mathscr{C}$-stable extension of $\mathcal{F}$ is a stable extension of $\mathcal{F}'$. The converse does not hold.*

It is worth noticing that when the constraint of a CAF is a tautology, then the extensions of this CAF coincide with those of its basic version (i.e. the argumentation framework without the constraint).

Let us illustrate this notion of CAFs through a simple example.

Example 1 (Cont): Assume an extended version of the argumentation framework $\mathcal{F}_1$ where we would like to accept the two arguments a and c. This is encoded by a constraint $\mathscr{C} : a \wedge c$. It can be checked that the CAF $(\mathcal{A}_1, \mathcal{R}_1, \mathscr{C})$ has one $\mathscr{C}$-stable extension which is $\mathcal{B}_1 = \{a, c\}$. Note that $\mathcal{B}_2 = \{b, d\}$ is a stable extension of $\mathcal{F}_1$ but not a $\mathscr{C}$-stable extension of its constrained version.

4.2   Mappings into CSPs

Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a given CAF. In order to compute its $\mathscr{C}$-extensions under different semantics, we follow the same line of research as in the previous section. The only difference is that in addition to the constraints defined in Section 3.2, there is an additional constraint which is $\mathscr{C}$.

Let us start with $\mathscr{C}$-stable extensions. They are computed by the stable CSP given in Def. 8 augmented by the constraint $\mathscr{C}$ in its set $\mathcal{C}$.

Definition 15 ($\mathscr{C}$-stable CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a constrained argumentation framework. A $\mathscr{C}$−stable CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{\mathscr{C}\} \cup \{a \Leftrightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} \neg b \mid a \in \mathcal{A}\}$.*

Note that the constraints in $\mathcal{C}$ are all propositional formulas built over a language $\mathcal{L}_\mathcal{A}$ whose alphabet is the set $\mathcal{A}$ of arguments. We show next that the solutions of a $\mathscr{C}$-stable CSP return all the $\mathscr{C}$-stable extensions of the corresponding CAF.

Theorem 8. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a $\mathscr{C}$-stable CSP associated with a CAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ such that $v(x_i) = 1$ is a $\mathscr{C}$-stable extension of $\mathcal{F}$.*

Example 1 (Cont): The $\mathscr{C}$-stable CSP associated with the CAF extending $\mathcal{F}_1$ with the constraint $\mathscr{C} : a \wedge c$ is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ s.t. $\mathcal{X} = \{a, b, c, d\}$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$, and $\mathcal{C} = \{a \wedge c, a \Leftrightarrow \neg d, b \Leftrightarrow \neg a, c \Leftrightarrow \neg b, d \Leftrightarrow \neg c\}$. This CSP has one solution which is $(1, 0, 1, 0)$. It returns the $\mathscr{C}$-stable extension $\{a, c\}$ of the CAF.

A CSP which computes the $\mathscr{C}$-admissible sets of a CAF is grounded on the admissible CSP introduced in Definition 9.

Definition 16 ($\mathscr{C}$-admissible CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$ be a constrained argumentation framework. A $\mathscr{C}$-admissible CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{\mathscr{C}\} \cup \{(a \Rightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} \neg b) \wedge (a \Rightarrow \bigwedge_{b:(b,a) \in \mathcal{R}} (\bigvee_{c:(c,b) \in \mathcal{R}} c)) \mid a \in \mathcal{A}\}$.*

We show that the solutions of this CSP are $\mathscr{C}$-admissible extensions of the corresponding CAF.

Theorem 9. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a $\mathscr{C}$-admissible CSP associated with a CAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is a $\mathscr{C}$-admissible set of the CAF $\mathcal{F}$.*

$\mathscr{C}$-preferred extensions are maximal (for set inclusion) admissible sets, then the following result follows from the previous one.

Theorem 10. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a $\mathscr{C}$-admissible CSP associated with a CAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \mathscr{C})$. Each maximal (for set inclusion) set $\{x_j, \ldots, x_k\}$, s.t. $v(x_i) = 1$ and $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP, is a $\mathscr{C}$-preferred extension of $\mathcal{F}$.*

5   Preference-Based Frameworks

Is is well acknowledged in argumentation literature that arguments may not have the same strength. For instance, arguments built from certain information are stronger than arguments built from uncertain information. Consequently, in [2] Dung's framework has been extended in such a way to take into account the

strengths of arguments when evaluating them. The idea is to consider in addition to the attack relation, another binary relation $\succeq$ which represents preferences between arguments. This relation can be instantiated in different ways. Writing $a \succeq b$ means that a is at least as good as b. Let $\succ$ be the strict relation associated with $\succeq$. It is defined as follows: $a \succ b$ iff $a \succeq b$ and   not $b \succeq a$. In Dung's framework, an attack always succeeds (if the attacked argument is not defended). In preference-based frameworks, an attack may fail if the attacked argument is stronger than its attacker.

Definition 17 (PAF). *A* preference-based argumentation framework *(PAF) is a tuple $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$ where $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is an attack relation and $\succeq$ is (partial or total) preorder on $\mathcal{A}$ ($\succeq \subseteq \mathcal{A} \times \mathcal{A}$).*
The extensions of $\mathcal{F}$ (under any semantics) are those of the AF $(\mathcal{A}, \mathtt{Def})$ where $(a, b) \in \mathtt{Def}$ iff $(a, b) \in \mathcal{R}$ and $not(b \succ a)$.

Let us now show how to compute the extensions of a PAF with a CSP. The following CSP computes the conflict-free sets of arguments in a PAF.

Definition 18. *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$ be a PAF. A CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{a \Rightarrow \neg b \; s.t. \; (a, b) \in \mathcal{R} \; and \; not(b \succ a)\}$.*

Theorem 11. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP instance associated with a PAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is conflict-free in PAF $\mathcal{F}$.*

Stable extensions of a PAF are computed by a slightly modified version of stable CSP.

Definition 19 (Pref-stable CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$ be a PAF. A pref stable CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{a \Leftrightarrow \bigwedge_{b:(b,a)\in\mathcal{R} \; and \; not(a\succ b)} \neg b \mid a \in \mathcal{A}\}$.*

Theorem 12. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a pref stable CSP associated with a PAF $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of this CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is a stable extension of $\mathcal{F}$.*

Example 1 (Cont):   Assume a PAF with $\mathcal{A}_1$ as its set of arguments, $\mathcal{R}_1$ its attack relation and that $b \succ a$ and $d \succ c$. Its corresponding pref stable CSP is $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ s.t. $\mathcal{X} = \{a, b, c, d\}$, $\mathcal{D} = \{\{0, 1\}, \{0, 1\}, \{0, 1\}, \{0, 1\}\}$, and $\mathcal{C} = \{a \Leftrightarrow \neg d, \; b \Leftrightarrow \top, \; c \Leftrightarrow \neg b, \; d \Leftrightarrow \top\}$. This CSP has one solution: $(0, 1, 0, 1)$. Thus, the set $\{b, d\}$ is the unique stable extensions of this PAF.

A CSP which computes the admissible sets of a PAF is an extended version of admissible CSP.

Definition 20 (Pref-admissible CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$ be a PAF. A pref-admissible CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{(a \Rightarrow \bigwedge\limits_{b:(b,a)\in\mathcal{R} \ and \ not(a\succ b)} \neg b) \wedge (a \Rightarrow$*

$\bigwedge\limits_{b:(b,a)\in\mathcal{R} \ and \ not(a\succ b)} (\bigvee\limits_{c:(c,b)\in\mathcal{R} \ and \ not(b\succ c)} c)) \mid a \in \mathcal{A}\}$.

We show that the solutions of this CSP are admissible extensions of the corresponding PAF.

Theorem 13. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a pref-admissible CSP associated with a PAF $\mathcal{F}$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of this CSP iff the set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ is an admissible set of $\mathcal{F}$.*

As preferred extensions are maximal (for set inclusion) admissible sets, then the following result follows from the previous one.

Theorem 14. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a pref-admissible CSP associated with a PAF $\mathcal{F}$. Each maximal (for set inclusion) set $\{x_j, \ldots, x_k\}$, s.t. $v(x_i) = 1$ and $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP, is a preferred extension of $\mathcal{F}$.*

Complete extensions are computed by a revised version of complete CSP.

Definition 21 (Pref-complete CSP). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R}, \succeq)$ be a PAF. A pref-complete CSP associated with $\mathcal{F}$ is a tuple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X} = \mathcal{A}$, for each $a_i \in \mathcal{X}$, $\mathcal{D}_i = \{0, 1\}$ and $\mathcal{C} = \{(a \Rightarrow \bigwedge\limits_{b:(b,a)\in\mathcal{R} \ and \ not(a\succ b)} \neg b)$*

$\wedge (a \Leftrightarrow \bigwedge\limits_{b:(b,a)\in\mathcal{R} \ and \ not(a\succ b)} (\bigvee\limits_{c:(c,b)\in\mathcal{R} \ and \ not(b\succ c)} c)) \mid a \in \mathcal{A}\}$.

Theorem 15. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a pref-complete CSP associated with a PAF $\mathcal{F}$. The tuple $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP iff the set $\{x_j, \ldots, x_k\}$, s.t. $v(x_i) = 1$ is a complete extension of $\mathcal{F}$.*

The grounded extension of a PAF is computed by the pref-complete CSP as follows.

Theorem 16. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a pref-complete CSP associated with a PAF $\mathcal{F}$. The grounded extension of $\mathcal{F}$ is the minimal (for set inclusion) set $\{x_j, \ldots, x_k\}$ s.t. $v(x_i) = 1$ and $(v(x_1), \ldots, v(x_n))$ is a solution of the CSP.*

6 Related Work

There are very few attempts in the literature for modeling argumentation frameworks as a CSP. To the best of our knowledge, the only works on the topic are [4,5].

In [5], the authors have studied the problem of encoding *weighted* argumentation frameworks by semirings. In a weighted framework, attacks do not necessarily have the same weights. Thus, a weight (i.e. a value between 0 and 1) is

associated with each attack between two arguments. When all the attacks have weight 1, the corresponding framework collapses with Dung's abstract framework recalled in Section 3.

In [5], it has been shown how to compute stable and complete extensions by semirings. In our paper, we have proposed an alternative approach for computing those semantics and other semantics (like preferred and grounded semantics). The approach is simpler and more natural. While in [5], the authors have used soft CSP, in our paper we have used simple CSP. Moreover, we have studied more semantics and two extended versions of Dung's framework: the constrained version proposed in [9] and the preferred version proposed in [2].

The works presented in [4,9] are closer to our. In these papers, the authors have encoded Dung's framework as a satisfiability problem (SAT). In [7], it has been shown that SAT is a particular case of CSPs and a mapping from SAT to CSP has been given. In our paper, we took advantage of that mapping and we presented different CSPs which encode Dung's semantics not only for Dung's framework, but also for constrained frameworks and preference-based frameworks.

In [18] an implementation of Dung's semantics using answer set programming (ASP) has been provided. Thus, it is complementary to our work. Moreover, the ASP literature has shown that there are links between ASP and CSP.

7 Conclusion

In this paper, we have expressed the problem of computing the extensions of an argumentation framework under a given semantics as a CSP. We have investigated three types of frameworks: Dung's argumentation framework [15], its constrained version proposed in [9], and its extension with preferences [2]. For each of these frameworks, we have proposed different CSPs which compute their extensions under various semantics, namely admissible, preferred, stable, complete and grounded.

Such mappings are of great importance since they allow the use of the *efficient solvers* that have been developed by CSP community. Thus, the efficiency of our different CSPs depend on that of the solver that is chosen to solve them. Note also that the CSP version of Dung's argumentation framework is as simple as this latter since a CSP can be represented as a graph.

It is worth mentioning that in the particular case of grounded semantics, there is an additional test of minimality that is required after computing the solutions of the corresponding CSP. This increases thus the complexity of computing the grounded extension of an argumentation framework. Consequently, this particular extension should be computed using existing algorithms in argumentation literature [1] and not by a CSP.

There are a number of ways to extend this work. One future direction consists of proposing the CSPs that return other semantics like semi-stable [6] and ideal [14]. Another idea consists of encoding weighted argumentation frameworks [16] as CSPs. In a weighted framework, attacks may not have the same

importance. Such framework can be encoded by valued CSP in which constraints are associated with weights.

References

1. Rahwan, I., Simari, G. (eds.): Argumentation in Artificial Intelligence. Springer, Heidelberg (2009)
2. Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence 34, 197–216 (2002)
3. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation 13(3), 429–448 (2003)
4. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: NMR, pp. 59–64 (2004)
5. Bistarelli, S., Santini, F.: A common computational framework for semiring-based argumentation systems. In: ECAI, pp. 131–136 (2010)
6. Caminada, M.: Semi-stable semantics. In: Proceedings of the 1st International Conference on Computational Models of Argument (COMMA 2006), pp. 121–130 (2006)
7. Castell, T., Fargier, H.: Propositional satisfaction problems and clausal csps. In: ECAI, pp. 214–218 (1998)
8. Cayrol, C., Doutre, S., Mengin, J.: On decision problems related to the preferred semantics for argumentation frameworks. Journal of Logic and Computation 13(3), 377–403 (2003)
9. Coste-Marquis, S., Devred, C., Marquis, P.: Constrained argumentation frameworks. In: KR, pp. 112–122 (2006)
10. Creignou, N.: The class of problems that are linearly equivalent to satisfiability or a uniform method for proving np-completeness. Theor. Comput. Sci. 145(1&2), 111–145 (1995)
11. Devred, C., Doutre, S., Lefèvre, C., Nicolas, P.: Dialectical proofs for constrained argumentation. In: COMMA, pp. 159–170 (2010)
12. Dimopoulos, Y., Nebel, B., Toni, F.: Preferred arguments are harder to compute than stable extensions. In: IJCAI 1999, pp. 36–43 (1999)
13. Dimopoulos, Y., Nebel, B., Toni, F.: Finding admissible and preferred arguments can be very hard. In: KR 2000, pp. 53–61 (2000)
14. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal skeptical argumentation. Artificial Intelligence Journal 171, 642–674 (2007)
15. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artificial Intelligence Journal 77, 321–357 (1995)
16. Dunne, P., Hunter, A., McBurney, P., Parsons, S., Wooldridge, M.: Inconsistency tolerance in weighted argument systems. In: AAMAS, pp. 851–858 (2009)
17. Dunne, P., Wooldridge, M.: Complexity of abstract argumentation. In: Rahwan, I., Simari, G. (eds.) Argumentation in Artificial Intelligence. ch.4, pp. 85–104 (2009)
18. Egly, U., Gaggl, S., Woltran, S.: Answer-set programming encodings for argumentation frameworks. In: Technical report DBAI-TR-2008-62, Technische Universitat Wien (2008)
19. Kumar, V.: Depth-first search. Encyclopaedia of Artificial Intelligence 2, 1004–1005 (1987)