Translation of an argumentation framework into a CP-boolean game

E. Bonzon
CRIP5, Université Paris Descartes
45 rue des Saints Pères
75006 Paris, France
elise.bonzon@mi.parisdescartes.fr

C. Devred
LERIA, Université d'Angers
2 Boulevard Lavoisier
49045 Angers Cedex 01, France
caroline.devred@info.univ-angers.fr

M-C. Lagasquie-Schiex
IRIT, Université Paul Sabatier
118 route de Narbonne
F-31062 Toulouse Cedex 9, France
lagasq@irit.fr

***Abstract*—There already exist some links between argumentation and game theory. For instance, dynamic games can be used for simulating interactions between agents in an argumentation process. In this paper, we establish a new link between these domains in a static framework: we show how an argumentation framework can be translated into a CP-Boolean game and how this translation can be used for computing extensions of argumentation semantics. We give formal algorithms to do so.**

***Index Terms*—AI and games, Knowledge representation and reasoning, Argumentation**

I. Introduction

Argumentation has become an influential approach to treat AI problems including defeasible reasoning and some forms of dialogue between agents (see e.g, [1], [2], [3], [4], [5]).

Argumentation is basically concerned with the exchange of interacting arguments. Usually, the interaction takes the form of a conflict, called attack. For example, a logical argument can be a pair ⟨set of assumptions, conclusion⟩, where the set of assumptions entails the conclusion according to some logical inference schema; then a conflict occurs for instance if the conclusion of an argument contradicts an assumption of another argument.

The main issue for any theory of argumentation is the selection of acceptable sets of arguments, based on the way arguments interact. Intuitively, an acceptable set of arguments must be in some sense "coherent" (*e.g.*, no conflict in this set) and "strong enough" (*e.g.*, able to defend itself against all attacking arguments). This concept of acceptability can be explored through argumentation frameworks (AF), and especially Dung's framework ([6]), which abstracts from the nature of the arguments, and represents interaction under the form of a binary relation "attack" on a set of arguments. However, even in the abstract framework of Dung, the complexity of the associated problems remains prohibitive in the general case (for instance, "verifying if a given set of arguments is a preferred extension" is a coNP-complete problem).

In another way, game theory attempts to formally analyze strategic interactions between agents. Roughly speaking, a non-cooperative game consists of a set of agents (or players), and for each agent, a set of possible strategies and a utility function mapping every possible combination of strategies to a real value (in this paper, we consider only *one-shot* games, where agents choose their strategies in parallel, without observing the others' choices).

A problem of this approach is the difficulty to express efficiently this utility function.[1] A way out consists in using a language for representing agents' preferences in a *structured* and *compact* way. An interesting use of these languages in Game theory is given by Boolean games: each agent has control over a set of *boolean* (binary) variables and her preferences consist in a specific *propositional formula* that she wants to be satisfied (a propositional formula is a very compact representation of preferences – see [7], [8]); unfortunately, this framework is also very restricted because of the dichotomy of preferences (a formula can only be true or false). However, some recent works have shown the possibility to extend these games by the use of *CP-nets* in place of a simple propositional formula (see [9]). In this context, some interesting results about complexity exist (for instance, "verifying if a given strategy profile is a pure Nash equilibrium" is a polynomial problem when each CP-net is acyclic).

Argumentation and game theory already have some common points. For instance, in [6], [10], games are used for defining proof theories and algorithms for some acceptability problems in argumentation. However, the considered games are always dynamic and there is no specific work concerning static games and argumentation. The aim of this paper is to identify a possible translation of an argumentation framework into a particular static game (a CP-Boolean game) in order to compute preferred extensions using solution concepts of game theory (pure strategy Nash equilibrium: PNE). So, we want to establish a new link between argumentation and games and then to give a new way for computing preferred extension, even if this new way is not more efficient than the existing algorithms (see for instance [11], [12], [13], [10]).

The paper is organized as follows : Dung's abstract framework and CP-Boolean games are respectively recalled in Section II and Section III. Section IV presents the core of this paper: how to translate an argumentation framework into a CP-Boolean game and how to use this game for computing

[1] Because the number of possible combinations of strategies is exponential in the number of players and in the number of variables controlled by each player.

extensions of argumentation semantics. Some related works are presented in Section V and Section VI concludes.

II. Argumentation frameworks (AF)

In [6], Dung has proposed an abstract framework for argumentation in which he focuses only on the definition of the status of arguments. For that purpose, he assumes that a set of arguments is given, as well as the different conflicts among them. We briefly recall that abstract framework:

Def. 1: An *argumentation framework (AF)* is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$ of a set $\mathcal{A}$ of arguments and a binary relation $\mathcal{R}$ on $\mathcal{A}$ called the *attack relation*. $\forall a_i, a_j \in \mathcal{A}$, $a_i \mathcal{R} a_j$ means that a_i *attacks* a_j (or a_j is *attacked* by a_i). An AF may be represented by a directed graph, called the *interaction graph*, whose nodes are arguments and edges represent the attack relation.

In Dung's framework, the *acceptability of an argument* depends on its membership to some sets, called extensions. These extensions characterize collective acceptability. Let $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$, $S \subseteq \mathcal{A}$. The main characteristic properties are:

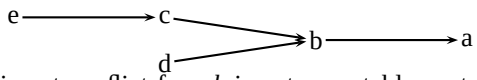
Def. 2: S is *conflict-free* for AF iff there exists no a_i, a_j in S such that $a_i \mathcal{R} a_j$. An argument a is *acceptable w.r.t. S* for AF iff $\forall b \in \mathcal{A}$ such that $b \mathcal{R} a$, $\exists c \in S$ such that $c \mathcal{R} b$. S is *acceptable* for AF iff $\forall a \in S$, a is acceptable w.r.t. S for AF.

Then several *semantics for acceptability* have been defined in [6]. For instance:

Def. 3: S is an *admissible set* for AF iff S is conflict-free and acceptable for AF. S is a *preferred extension* of AF iff S is $\subseteq$-maximal among the admissible sets for AF. S is a *stable extension* of AF iff S is conflict-free and S attacks each argument which does not belong to S.

These notions are illustrated on the following example.

Ex. 1: Consider $\mathsf{AF} = \langle \{a,b,c,d,e\}, \{(b,a), (c,b), (d,b), (e,c)\} \rangle$ represented by:



$\{e,c\}$ is not conflict-free; b is not acceptable w.r.t. $\{e\}$; a is acceptable w.r.t. $\{e,d\}$; $\{d,a\}$ is an admissible set and $\{e,d,a\}$ is the preferred and stable extension of AF.

We will need the following properties ([6], [12], [13], [14]):

Prop. 1: Let $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ such that $\mathcal{A} \neq \varnothing$.

1) Each unattacked argument belongs to every preferred extension of AF (see [6]).
2) An acyclic argumentation framework AF contains only one preferred extension (see [12], [13], [14]).
3) If $\varnothing$ is the unique preferred extension then AF contains at least an odd-length cycle (see [12], [13], [14]).
4) If AF does not contain an odd-length cycle then its preferred extensions are not empty (see [12], [13], [14]).
5) If AF does not contain an odd-length cycle then each preferred extension is also stable[2] (see [12], [13], [14]).

Note that many works in argumentation domain consider that odd-length cycles may be considered as paradoxes, as they are a generalization of an argument attacking himself. We agree with this opinion[3] and, in this paper, we will consider that *argumentation frameworks should not contain odd-length cycles*;[4] however, argumentation frameworks with even-length cycles will be taken into account. When there is no odd-length cycle, an interesting consequence occurs:

Conseq. 1: Let $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF without odd-length cycle. Let E be a preferred extension of AF. Let a be an argument of AF. If there is no attacker of a in E then $a \in E$.

Proof: We can apply Prop. 1.5. So E is a stable extension. So, if we assume that $a \notin E$ then, because E is stable, $\exists b \in E$ such that $b \mathcal{R} a$. But this fact is impossible because there is no attacker of a in E. So $a \in E$. ■

III. CP-Boolean games

Let us start by introducing some background. Let V be a finite set of propositional variables and L_V be the propositional language built from V and the usual connectives as well as the Boolean constants $\top$ (true) and $\bot$ (false). Formulas of L_V are denoted by φ, ψ, etc. 2^V is the set of the interpretations for V, with the usual convention that for $M \in 2^V$ and $x \in V$, M gives the value true to x if $x \in M$ and false otherwise. Let $X \subseteq V$. 2^X is the set of *X-interpretations*. X-interpretations are denoted by listing all variables of X, with a $^-$ symbol when the variable is set to false: for instance, let $X = \{a,b,d\}$, then the X-interpretation $M = \{a,d\}$ is denoted $a\bar{b}d$. A *preference relation* $\succeq$ is a reflexive and transitive binary relation (not necessarily complete) on 2^V. Consider $M, M' \in 2^V$. The strict preference $\succ$ associated with $\succeq$ is defined as usual by $M \succ M'$ iff $M \succeq M'$ and not $M' \succeq M$.

A. CP-net

In this section, we consider a very popular language for compact preference representation on combinatorial domains, namely CP-nets. This graphical model exploits conditional preferential independence in order to structure the decision maker's preferences under a *ceteris paribus* assumption [15], [16]. Although CP-nets generally consider variables with arbitrary finite domains, here we consider only "propositionalized" CP-nets, that is, CP-nets with binary variables.

Def. 4: Let V be a set of propositional variables and $\{X,Y,Z\}$ be a partition of V. X is *conditionally preferentially independent* of Y given Z iff $\forall z \in 2^Z$, $\forall x_1, x_2 \in 2^X$ and $\forall y_1, y_2 \in 2^Y$ we have: $x_1 y_1 z \succeq x_2 y_1 z$ iff $x_1 y_2 z \succeq x_2 y_2 z$.

For each variable X, the agent specifies a set of *parent variables* $Pa(X)$ that can affect her preferences over the values of X. Formally, X and $V \setminus (\{X\} \cup Pa(X))$ are conditionally preferentially independent given $Pa(X)$. This information is used to create the CP-net:

Def. 5: Let V be a set of propositional variables. $\mathcal{N} = \langle \mathcal{G}, \mathcal{T} \rangle$ is a *CP-net on V*, where $\mathcal{G}$ is a directed graph over V, and $\mathcal{T}$ is a set of conditional preference tables $CPT(X_j)$ for each $X_j \in V$. Each $CPT(X_j)$ associates a linear order $\succ_p^j$ with each instantiation $p \in 2^{Pa(X_j)}$.

[2] One says that the argumentation framework is coherent (see [6]).

[3] Even if some odd-length cycles may make sense.

[4] And, if it is not the case, these odd-length cycles will be removed of the argumentation framework.

The preference information captured by a CP-net $\mathcal{N}$ can be viewed as a set of logical assertions about a user's preference ordering over complete assignments to variables in the network. These statements are generally not complete, that is, they do not determine a unique preference ordering.

Ex. 2: Consider the CP-net given by Figure 1 about my preferences for the dinner. Variables S and W correspond respectively to the soup and the wine. I strictly prefer to eat a fish soup (S_p) rather than a vegetable soup (S_l), and about wine, my preferences depend on the soup I eat: I prefer red wine (W_r) with vegetable soup ($S_l : W_r \succ W_b$) and white wine (W_b) with fish soup ($S_p : W_b \succ W_r$). So $D(S) = \{S_p, S_l\}$ and $D(W) = \{W_r, W_b\}$.

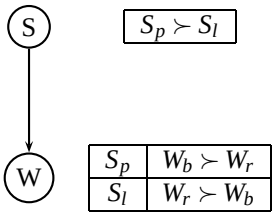


Fig. 1. CP-net "My dinner"

Figure 2 represents the preference relation induced by this CP-net. The bottom element ($S_l \wedge W_b$) is the worst case and the top element ($S_p \wedge W_b$) is the best case.

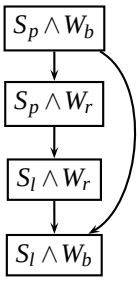


Fig. 2. Preference graph induced by the CP-net "My dinner"

There is an arrow between the nodes ($S_p \wedge W_b$) and ($S_l \wedge W_b$) because we can compare these states, every other thing being equal.

In this case, we can completely order the possible states (from the most preferred one to the least preferred one) :

$$(S_p \wedge W_b) \succ (S_p \wedge W_r) \succ (S_l \wedge W_r) \succ (S_l \wedge W_b)$$

This relation $\succ$ is the only ranking that satisfies this CP-net.

B. CP-Boolean games

Boolean games [7], [8] yield a compact representation of 2-player zero-sum static games with binary preferences. In [9], Boolean games are generalized with non dichotomous preferences: they are coupled with propositionalized CP-nets.

Def. 6: A *CP-Boolean game* is a 4-uple $G = (N, V, \pi, \Phi)$, where N is a set of players, V is a set of propositional variables, $\pi : N \mapsto V$ is a control assignment function which defines a partition of V, and $\Phi = \langle \mathcal{N}_1, \ldots, \mathcal{N}_n \rangle$. Each $\mathcal{N}_i$ is a CP-net on V whose graph is denoted by $\mathcal{G}_i$, and $\forall i \in N$, $\succeq_i = \succeq_{\mathcal{N}_i}$.

The control assignment function π maps each player to the variables she controls and each variable is controlled by one and only one agent,[5] *i.e.*, $\{\pi_1, \ldots, \pi_n\}$ forms a partition of V.

Def. 7: Let $G = (N, V, \pi, \Phi)$ be a CP-Boolean game. A *strategy* s_i for a player i is a π_i-interpretation. A *strategy profile* s is a n-tuple $s = (s_1, \ldots, s_n)$ where for all i, $s_i \in 2^{\pi_i}$.

In other words, a strategy for i is a truth assignment for all the variables i controls. As $\{\pi_1, \ldots, \pi_n\}$ forms a partition of V, a strategy profile defines an (unambiguous) interpretation for V. Slightly abusing notation and words, we write $s \in 2^V$, to refer to the value assigned by s to some variable.

In the rest of the paper, we make use of the following notations which are standard in game theory: let $G = (N, V, \pi, \Phi)$ be a Boolean game with $N = \{1, \ldots, n\}$, and $s = (s_1, \ldots, s_n)$, $s' = (s'_1, \ldots, s'_n)$ be two strategy profiles; s_{-i} denotes the projection of s onto $N \setminus \{i\}$: $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$; similarly, π_{-i} denotes the set of the variables controlled by all players except i: $\pi_{-i} = V \setminus \pi_i$; finally, (s'_i, s_{-i}) denotes the strategy profile obtained from s by replacing s_i with s'_i without changing the other strategies: $(s'_i, s_{-i}) = (s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n)$.

A pure strategy Nash equilibrium (PNE) is a strategy profile such that each player's strategy is a best response to the other players' strategies. PNEs are classically defined for games where preferences are complete, which is not necessarily the case here. So we introduce the notion of *strong* PNE.

Def. 8: Let $G = (N, V, \pi, \Phi)$ and $Pref_G = \langle \succeq_1, \ldots, \succeq_n \rangle$ the collection of preference relations on 2^V induced from Φ. Let $s = (s_1, \ldots, s_n) \in 2^V$. s is a *strong PNE* (SPNE) for G iff $\forall i \in \{1, \ldots, n\}$, $\forall s'_i \in 2^{\pi_i}, (s'_i, s_{-i}) \preceq_i (s_i, s_{-i})$.

The following proposition has been shown in [9].

Prop. 2: Let $G = (N, V, \pi, \Phi)$ be a CP-Boolean game such that graphs $\mathcal{G}_i$ are all identical ($\forall i, j \in N$, $\mathcal{G}_i = \mathcal{G}_j$) and acyclic. Then G has one and only one strong PNE.

The proof of this result makes use of the *forward sweep* procedure [15] for outcome optimization (this procedure consists in instantiating variables following an order compatible with the graph, choosing for each variable its preferred value given the value of its parents). Moreover, as shown in [9], this SPNE can be built in polynomial time.

IV. Argumentation and CP-Boolean games

Our objective here is to transform an AF into a CP-Boolean game G, and thus to use well-known tools of game theory, and more specifically properties of CP-Boolean games, in order to find the preferred extensions of AF. By this work, we mainly want to establish a new link between argumentation and games.

A. Translation of an argumentation framework into a CP-Boolean game

This transformation is done by Algorithm 1. This algorithm assumes the existence of two others algorithms:

[5] The set of all variables controlled by i will be written π_i instead of $\pi(i)$.

- IsCyclic which returns true if there exists at least one cycle in the argumentation graph,[6]
- RemOddCycles for removing the odd-length cycles if there are some of them in the AF.[7]

The execution of these two algorithms can be viewed as a precompilation step of Algorithm 1. One can say that as Algorithm IsCyclic does not directly detect odd-length cycles, it is useless in the precompilation of Algorithm 1. However, as IsCyclic is a linear-time algorithm whereas RemOddCycles is only a polynomial-time one, we think that it is interesting to avoid an unnecessary execution of RemOddCycles when AF is acyclic.

Let AF be an argumentation framework which does not contain odd-length cycles, the principles of Algorithm 1 are the following:

- each argument of AF is a variable of G;
- each variable is controlled by a different player (so we have as many players as variables);
- the CP-nets of all players are defined in the same way:
  - the graph of the CP-net is exactly the directed graph of AF;
  - the preferences over each variable v which is not attacked are $v \succ \overline{v}$ (if an argument is not attacked, we want to protect it; so the value true of the variable v is preferred to its value false),
  - the preferences over each variable v which is attacked by the set of variables $\mathcal{R}^{-1}(v)$ depends on these variables: if at least one variable $w \in \mathcal{R}^{-1}(v)$ is satisfied, v cannot be satisfied (so we have $\bigvee_{w\in\mathcal{R}^{-1}(v)} w : \overline{v} \succ v$[8]); otherwise, if all variables $w \in \mathcal{R}^{-1}(v)$ are not satisfied, v can be satisfied (and so $\bigwedge_{w\in\mathcal{R}^{-1}(v)} \overline{w} : v \succ \overline{v}$).

The construction of a CP-Boolean game G from an argumentation framework AF is made in polynomial time (even if AF is cyclic and if we have to remove its odd-length cycles).

The use of this algorithm implies the following property:

Prop. 3: Let $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. Let $G = (N, V, \pi, \Phi)$ be the CP-Boolean game and AF' be the argumentation framework both obtained from AF by applying Algorithm 1.

s is a preferred extension of AF' iff s is a SPNE for[9] G.

Proof: Let us first study the $\Rightarrow$ direction.

Let s be a preferred extension of AF'. Assume than s is not a SPNE of the CP-Boolean game associated. So, $\exists i \in N$, $\exists s'_i \in 2^{\pi_i}$, $\exists s_{-i} \in 2^{\pi_{-i}}$, such that $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$. Let x_i be the variable in V such that $\pi_i = \{x_i\}$ (x_i is also an argument of AF'). We have several cases:

- $\mathcal{R}^{-1}(x_i) = \varnothing$ (x_i is unattacked). We know from Algorithm 1 that we have $CPT(x_i) = x_i \succ \overline{x}_i$. As $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, we know than $s_i = \overline{x}_i$ ($x_i \notin s$). But, we know from Prop. 1.1 than if $\mathcal{R}^{-1}(x_i) = \varnothing$ then $x_i \in s$. We have a contradiction.
- $\mathcal{R}^{-1}(x_i) \neq \varnothing$ (there exists at least one attacker of x_i). We know from Algorithm 1 that we have $CPT(x_i) = \{\bigvee_{w\in\mathcal{R}^{-1}(x_i)} w : \overline{x}_i \succ x_i\} \cup \{\bigwedge_{w\in\mathcal{R}^{-1}(x_i)} \overline{w} : x_i \succ \overline{x}_i\}$. There are two cases:
  - $\forall x_j$ such that $x_j \mathcal{R} x_i$, $x_j \notin s$. So, $\bigwedge_{w\in\mathcal{R}^{-1}(x_i)} \overline{w}$ holds and using $CPT(x_i)$, we can deduce that $x_i \succ \overline{x}_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, $s_i = \overline{x}_i$ and $s'_i = x_i$. Thus, x_i is not in s. But this is in contradiction with

Algorithm 1: Translation of an argumentation system into a CP-Boolean game

begin

/* INPUT: $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ an argumentation system */
/* OUTPUTS: $G = (N, V, \pi, \Phi)$ a CP-Boolean game, AF after removal of odd-length cyles */
/* LOCAL VARIABLES: i = current agent, a = current argument */

/* if necessary, removal of the odd-length cycles */
if IsCyclic(AF) **then** AF = RemOddCycles(AF)
/* no more odd-length cycle in AF */
/* computation of the CPTs for each argument */
for $a \in \mathcal{A}$ **do**
  if $\mathcal{R}^{-1}(a) = \varnothing$ **then** $CPT(a) = a \succ \overline{a}$ /* unattacked argument */
  else
    $CPT(a) =$ /* case of the other arguments */
    $\{\bigvee_{v\in\mathcal{R}^{-1}(a)} v : \overline{a} \succ a\}$
    $\cup$ $\{\bigwedge_{v\in\mathcal{R}^{-1}(a)} \overline{v} : a \succ \overline{a}\}$
/* computation of the CP-net $\mathcal{N}$ */
$\mathcal{N} = \langle \mathsf{AF}, \cup_{a\in\mathcal{A}} CPT(a) \rangle$ /* it is the attack graph*/ /* after removal of odd-length cycles, */ /* associated with the CPTs of each argument */
/* computation of N, V, π and Φ */
$i = 1$
$N = \varnothing$
$V = \mathcal{A}$ /* each argument is a variable */
for $a \in \mathcal{A}$ **do**
  $N = N \cup \{i\}$ /* an agent per each argument */
  $\pi_i = \{a\}$ /* i controls only this argument */
  $\mathcal{N}_i = \mathcal{N}$ /* the same CP-net for each agent */
  $i = i + 1$
return ($G = (N, V, \pi, \langle \mathcal{N}_1, \ldots, \mathcal{N}_{|V|} \rangle)$, AF)

end

[6]This algorithm is linear:
(Step 1) removing all the vertices which do not have predecessors;
(Step 2) iterating Step 1 until either all the remained vertices have at least one predecessor (there is a cycle in the initial graph), or the graph is empty (there is no cycle in the initial graph).

[7]This algorithm is polynomial:
(Step 1) computation of the Boolean adjacency matrix corresponding to all the minimal odd-length paths of attack; it is sufficient to take the Boolean adjacency of the graph $\mathcal{M}$ ($\mathcal{M}(i,j) = 1$ if there is an edge from i to j in AF) and to compute $\mathcal{M}^{\text{olc}} = \mathcal{M}^1 + \mathcal{M}^3 + \ldots + \mathcal{M}^{2n-1}$ with $n = |\mathcal{A}|$ (the bound $2n-1$ is obtained using a general result given by graph theory: if a directed graph contains a path from a to b then there exists a simple path – a path in which each vertex appears only one – from a to b);
(Step 2) removal of all the arguments for which the diagonal element of $\mathcal{M}^{\text{olc}}$ is 1;
(Step 3) removal of all the edges having one removed argument as end point or as start point.

[8]The formula $w : \overline{v} \succ v$ (resp. $\overline{w} : \overline{v} \succ v$) means that, for the value true (resp. false) of the variable w, the value false of the variable v is preferred to its value true.

[9]Recall that s denotes a V-interprétation, that is if $s = a\overline{b}c$ for example, this corresponds to the set $\{a, c\}$.

the conclusion obtained applying Conseq. 1 which says that $x_i \in s$ (AF' is an argumentation framework without odd-length cycle and there is no attacker of x_i in the preferred extension s). So this case is impossible.

– At least one argument x_j in $\mathcal{R}^{-1}(x_i)$ belongs to s. So, $\bigvee_{w \in \mathcal{R}^{-1}(x_i)} w$ holds and using $CPT(x_i)$ we can deduce that $\overline{x}_i \succ x_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, $s_i = x_i$. But, this is in contradiction with the fact that s must be conflict-free ($x_j \in s$ and $x_i \in s$). So this case is also impossible.

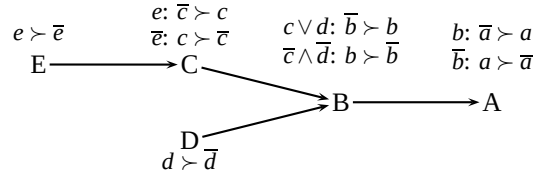
In conclusion, each case is impossible if we assume that s is not a SPNE. So, s is a SPNE.

Let us study now the $\Leftarrow$ direction.
Let $s = (s_1, \ldots, s_n)$ be a SPNE for G. Assume that s is not a preferred extension of AF'; so, either s is not conflict-free ($\exists x_i, x_j \in s$ such that $x_i \mathcal{R} x_j$ or $x_j \mathcal{R} x_i$), or s is not acceptable ($\exists x_i \in s$, $\exists x_j \in \mathcal{A}$ such that $x_j \mathcal{R} x_i$ and $\nexists x_k \in s$ such that $x_k \mathcal{R} x_j$).

- s is not conflict-free:
  – $\exists x_i, x_j \in s$ such that $x_i \mathcal{R} x_j$. As $x_i \in s$, we know that $\bigvee_{w \in \mathcal{R}^{-1}(x_j)} w$ holds and using $CPT(x_j)$ we can conclude that $\overline{x}_j \succ x_j$. As we know than s is a SPNE, we have for the player j who controls x_j, $\forall s'_j$, $\forall s_{-j}$, $(s_j, s_{-j}) \succ_j (s'_j, s_{-j})$. So, $s_j = \overline{x}_j$ and $x_j \notin s$, which is a contradiction.
  – $\exists x_i, x_j \in s$ such that $x_j \mathcal{R} x_i$. As $x_i \in s$ and s is a SPNE, we know that $x_i \succ \overline{x}_i$; so, using $CPT(x_i)$, we can conclude that $\bigwedge_{w \in \mathcal{R}^{-1}(x_i)} \overline{w}$ holds. So, $x_j \notin s$, which is a contradiction.
- s is not acceptable: $\exists x_i \in s$, $\exists x_j \in \mathcal{A}$ such that $x_j \mathcal{R} x_i$ and $\nexists x_k \in s$ such that $x_k \mathcal{R} x_j$. As $x_i \in s$ and s is a SPNE, we know that $x_i \succ \overline{x}_i$, and using $CPT(x_i)$, we can deduce that $\bigwedge_{w \in \mathcal{R}^{-1}(x_i)} \overline{w}$ holds. As $x_j \mathcal{R} x_i$, we know that $x_j \notin s$. So, $\overline{x}_j \succ x_j$ and using $CPT(x_j)$ we can deduce that $\bigvee_{w \in \mathcal{R}^{-1}(x_j)} w$ holds. That means that there exists $x_k \in s$ such that $x_k \in \mathcal{R}^{-1}(x_j)$, which is a contradiction.

In conclusion, each case is impossible if we assume that s is not a preferred extension. So, s is a preferred extension. ■

Ex. 3: Consider $\mathsf{AF} = \langle \{a,b,c,d,e\}, \{(b,a), (c,b), (d,b), (e,c)\} \rangle$ (AF is acyclic) and transform it in a CP-Boolean game $G = (N, V, \pi, \Phi)$. By applying Algorithm 1, $V = \{a,b,c,d,e\}$ and $N = \{1,2,3,4,5\}$, with $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$, $\pi_4 = \{d\}$ and $\pi_5 = \{e\}$. The following CP-net represents the preferences of all players[10]:

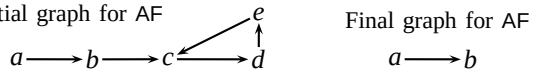


G has one SPNE $\{ed\overline{c}\overline{b}a\}$ and AF has only one preferred extension $\{e,d,a\}$.

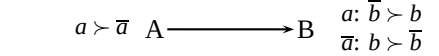
Ex. 4: Consider $\mathsf{AF} = \langle \{a,b\}, \{(a,b), (b,a)\} \rangle$. By applying Algorithm 1, $V = \{a,b\}$ and $N = \{1,2\}$, with $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ (AF is cyclic, but contains only even-length cycles). The following CP-net represents the preferences of all players:

G has two SPNEs $\{a\overline{b}, \overline{a}b\}$ and AF has two preferred extensions $\{a\}$, $\{b\}$.

Ex. 5: Consider $\mathsf{AF} = \langle \{a,b,c,d,e\}, \{(a,b), (b,c), (c,d), (d,e), (e,c)\} \rangle$. The initial AF is cyclic, and contains an odd-length cycle, which has to be removed. The final AF will contain only a and b.



So, by applying Algorithm 1, we have $V = \{a,b\}$, $N = \{1,2\}$, with $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ and the following CP-net which represents the preferences of all players:



G has one SPNE $\{a\overline{b}\}$ and the final AF (*after removal of odd-length cycles*) has one preferred extension $\{a\}$.

B. Computation of preferred extensions

Since preferred extensions correspond exactly to SPNEs, the main properties about computation of SPNE in CP-Boolean games can be applied. The first interesting case concerns the acyclic argumentation frameworks:

Prop. 4: Let AF be an argumentation framework. Let G be the CP-Boolean game and AF' be the argumentation framework both obtained from AF by applying Algorithm 1.
If AF' is acyclic, AF' has one and only one preferred extension which is computable in polynomial time using G.

Proof: The transformation of AF in the CP-Boolean game by applying Algorithm 1 is done in polynomial time. Then, the computation of the SPNE of this game using the forward sweep procedure is also computable in polynomial time (see Prop. 2 in [9]) and Prop. 3 shows that this SPNE corresponds to the preferred extension of AF' (AF after removal of odd-length cycles). ■

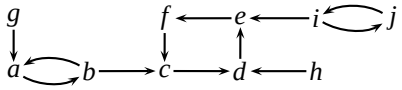
This proposition holds for the simple case of acyclic argumentation frameworks. The computation of SPNE(s) for cyclic argumentation frameworks is much more complex. However, Algorithms 2 and 3 allow to compute such solution concept when the argumentation framework contains even-length cycles.

These algorithms assume the existence of Algorithm CompIntCycleForProp which returns the cycle (or one of the cycles if there are several) in a given set of variables which permits to reach more variables as possible.[11] For instance, on the following graph:

[10] In order to distinguish the CP-net to the AF, nodes in the CP-net are in uppercase, whereas nodes in the AF are in lowercase.

[11] This algorithm uses the notion of Boolean adjacency matrix as Algorithm RemOddCycles:

- computation of the Boolean adjacency matrix $\mathcal{M}^{\text{ap}}$ corresponding to all minimal paths in the graph reduced to the given set of variables: $\mathcal{M}^{\text{ap}} = \mathcal{M} + \mathcal{M}^2 + \mathcal{M}^3 + \ldots + \mathcal{M}^{2n}$ with $n = |V|$; $\mathcal{M}^{\text{ap}}(i)$ will denote $(\mathcal{M}^{\text{ap}}(i,1), \ldots, \mathcal{M}^{\text{ap}}(i,n))$;
- ToSee $= V$; $C = \varnothing$; end? =false;



$\{a,b\}$ permits to reach the variables a, b, c, d, e, f, and $\{i,j\}$ permits to reach the variables i, j, c, d, e, f. These cycles are more interesting than the other ones for the propagation of values over the graph (if they are the starting point of a propagation process then this propagation is more efficient).

Let $\mathcal{N}$ be the CP-net representing goals of players of a CP-Boolean game, the principles of Algorithms 2 and 3 are:

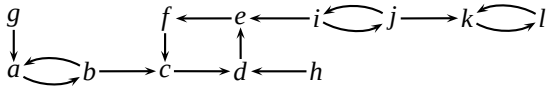
- instantiation of all unattacked variables (which have no parents in $\mathcal{N}$ and are satisfied in the SPNE);
- propagation of these instantiations as long as possible;
- once all feasible instantiations have been done, loop:
  - if all variables have been instantiated, the SPNE can be returned;
  - else, with Algorithm CompIntCycleForProp, the more interesting cycle C remaining is computed (there is one, otherwise all variables would have been instantiated);
  - using the current state of the current SPNE, create two new SPNEs; the first one contains a variable of C instantiated to true, the second one contains this same variable instantiated to false
  - propagation of these instantiations for each one of these SPNEs as long as possible.

Algorithm 2: Computation of SPNEs of a CP-Boolean game obtained from an argumentation framework

```
begin
    /* INPUTS: a CP-Boolean game G = (N,V,π,Φ), where
    Φ = ⟨N_1,...,N_n⟩ */
    /* OUTPUTS: a set of SPNEs SP */
    /* LOCAL VARIABLES: v = current variable, In = (resp.
    Out =) set of variables instantiated to true (resp. false),
    R = set of variables remaining to be instantiated */

    In = ∅, Out = ∅, R = V                    /* Initialization */
    /* Instantiation of all variables without parents */
    for v ∈ R do
        if Pa(v) = ∅ then
            R = R \ {v}
            In = In ∪ {v}
    /* propagation by a recursive process */
    return CompSPNERec(G, R, In, Out)
end
```

Ex. 6: Using the following graph:



- loop: while NOT(end?) do
  $v = \texttt{top}(\texttt{ToSee})$; $\texttt{ToSee} = \texttt{ToSee} \setminus \{v\}$;
  if $(\nexists w \in \texttt{ToSee}$ s.t. $\mathcal{M}^{\text{ap}}(v) \subset \mathcal{M}^{\text{ap}}(w))$ then
  /* no var. permitting to reach more var. than v */
  $C = C \cup \{v\}$;
  $\forall w \in \texttt{ToSee}$ do if $\mathcal{M}^{\text{ap}}(v) = \mathcal{M}^{\text{ap}}(w)$ then $C = C \cup \{w\}$;
  end? =true;
  else if ToSee is empty then end? =true;
- Return C

Algorithm 3: CompSPNERec: Recursive computation of SPNEs of a CP-Boolean game obtained from an argumentation framework

```
begin
    /* INPUTS: a CP-Boolean game G = (N,V,π,Φ),
        R = set of variables remaining to be instantiated,
        In = set of variables already instantiated to true,
        Out = set of variables already instantiated to false*/
    /* OUTPUTS: a set of SPNEs SP */
    /* LOCAL VARIABLES: v = current variable, n =
    cardinal of R, C = set of variables forming a cycle*/

    if R = ∅ then
        /* all variables are instantiated: a SPNE is found */
        return {(In Out̄)}
    else
        n = |R|   /* n = number of variables remaining to be
        instantiated */
        for v ∈ R do
            /* simple propagation process */
            if Pa(v) ⊆ Out then
                /* all parents are instantiated to false */
                In = In ∪ {v}
                R = R \ {v}
            else
                if (Pa(v) ∩ In) ≠ ∅ then
                    /* at least 1 parent instantiated to true */
                    Out = Out ∪ {v}
                    R = R \ {v}
        if n = |R| then
            /* none variable instantiated in For instruction */
            C = CompIntCycleForProp(G, R)
            v = Top(C)
            return(
              CompSPNERec(G, R \ {v}, In ∪ {v}, Out) ∪
              CompSPNERec(G, R \ {v}, In, Out ∪ {v}))
        else
            /* at least 1 variable instantiated in For instr. */
            return CompSPNERec(G, R, In, Out)
end
```

the steps of the computation process are:

- g and h are instantiated to true (current state of SPNE $= gh$);
- then a and d are instantiated to false (current state of SPNE $= gh\bar{a}\bar{d}$);
- then b is instantiated to true (current state of SPNE $= gh\bar{a}\bar{d}b$);
- then c is instantiated to false (current state of SPNE $= gh\bar{a}\bar{d}b\bar{c}$);
- at this point the simple propagation stops; so we must compute the interesting cycles in the remaining set of variables (e,f,i,j,k,l) and the result is (i,j);
- the propagation process is restarted with the following current states of two SPNEs: $gh\bar{a}\bar{d}b\bar{c}i$ and $gh\bar{a}\bar{d}b\bar{c}\bar{i}$;

- so, at the end of the propagation process, three SPNEs are obtained $gh\overline{a}\overline{d}b\overline{c}i\overline{e}f\overline{j}k\overline{l}$, $gh\overline{a}\overline{d}b\overline{c}\overline{i}ef\overline{j}k\overline{l}$ and $gh\overline{a}\overline{d}b\overline{c}\overline{i}e\overline{f}j\overline{k}l$ and $gh\overline{a}\overline{d}b\overline{c}\overline{i}e\overline{f}\overline{j}\overline{k}l$. These SPNEs correspond to the three preferred extensions $\{g,h,b,e,j,l\}$, $\{g,h,b,f,i,k\}$ and $\{g,h,b,i,f,l\}$.

The following proposition shows that Algorithms 2 and 3 allow to exactly compute the set of SPNEs of the CP-Boolean game.

Prop. 5: Let G be a CP-Boolean game given by Algorithm 1. Let SP be the set of strategy profiles of G given by Algorithms 2 and 3. $s \in SP$ iff s is a SPNE for G.

Proof: Let us first study the $\Rightarrow$ direction.
Let $s \in SP$. Assume than s is not a SPNE of the CP-Boolean game associated. So, $\exists i \in N$, $\exists s'_i \in 2^{\pi_i}$, $\exists s_{-i} \in 2^{\pi_{-i}}$, such that $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$.
Let x_i be the variable in V such that $\pi_i = \{x_i\}$. We have several cases:

- $Pa(x_i) = \varnothing$. We know from Algorithm 1 that we have $CPT(x_i) = x_i \succ \overline{x}_i$. As $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, we know than $s_i = \overline{x}_i$. But, we know from Algorithm 2 than $x_i \in s$. We have a contradiction.
- $Pa(x_i) \neq \varnothing$. We know from Algorithm 1 that we have $CPT(x_i) = \{\bigvee_{v \in Pa(x_i)} v : \overline{x}_i \succ x_i\} \cup \{\bigwedge_{v \in Pa(x_i)} \overline{v} : x_i \succ \overline{x}_i\}$.
  - All variables in $Pa(x_i)$ are not satisfied: $\bigwedge_{v \in Pa(x_i)} \overline{v}$ holds. In this case, we know that $x_i \succ \overline{x}_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, we have $s_i = \overline{x}_i$. But, we know from Algorithm 2 than if $Pa(x_i) \subseteq Out$, that is if all variables in $Pa(x_i)$ are not satisfied, then we have $x_i \in In$, so $x_i \in s$, which leads to a contradiction.
  - At least one variable in $Pa(x_i)$ is satisfied: $\bigvee_{v \in Pa(x_i)} v$ holds. In this case, we know that $\overline{x}_i \succ x_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, $s_i = x_i$. But, we know from Algorithm 2 than if $Pa(x_i) \cap In \neq \varnothing$, that is if at least one variable in $Pa(x_i)$ is satisfied, then we have $x_i \in Out$, so $\overline{x}_i \in s$, which leads to a contradiction.

So s is a SPNE for G.
Let us study now the $\Leftarrow$ direction.
Let $s = (s_1, \ldots, s_n)$ be a SPNE for G. We have to show than $s \in SP$, that is $\forall i$, if $s_i = x_i$ then $x_i \in In$ else $x_i \in Out$ (case corresponding to $s_i = \overline{x}_i$).

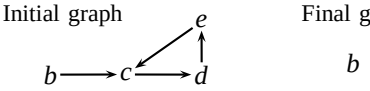
- $Pa(x_i) = \varnothing$. We know from Algorithm 1 that we have $CPT(x_i) = x_i \succ \overline{x}_i$. As s is a SPNE, we know than $s_i = x_i$. Moreover, we know from Algorithm 2 than $x_i \in In$.
- $Pa(x_i) \neq \varnothing$. We know from Algorithm 1 that we have $CPT(x_i) = \{\bigvee_{v \in Pa(x_i)} v : \overline{x}_i \succ x_i\} \cup \{\bigwedge_{v \in Pa(x_i)} \overline{v} : x_i \succ \overline{x}_i\}$.
  - All variables in $Pa(x_i)$ are not satisfied: $\bigwedge_{v \in Pa(x_i)} \overline{v}$ holds and, in this case, we know that $x_i \succ \overline{x}_i$. So, as s is a SPNE, we know than $s_i = x_i$. Moreover, we know from Algorithm 3 than if $Pa(x_i) \subseteq Out$, that is if all variables in $Pa(x_i)$ are not satisfied, then we have $x_i \in In$.
  - At least one variable in $Pa(x_i)$ is satisfied: $\bigvee_{v \in Pa(x_i)} v$ holds and, in this case, we know that $\overline{x}_i \succ x_i$. So, as s is a SPNE, we know than $s_i = \overline{x}_i$. Moreover, we know from Algorithm 3 than if $Pa(x_i) \cap In \neq \varnothing$, that is if at least one variable in $Pa(x_i)$ is satisfied, then we have $x_i \in Out$.

So $s \in SP$. ■

C. Managing odd-length cycles

Of course, the removal of odd-length cycles has an important influence on the computation of the SPNE(s) and this point could be considered as problematic in some cases if one does not agree with our initial assumption: in general, an odd-length cycle may be considered as a paradox. Of course, we know that some odd-length cycles make sense, in particular when they are not strict odd-length cycles. But the work presented in this paper is preliminary and the removal of this kind of cycles guarantees some important properties (see Prop. 1 and Conseq. 1). The treatment of odd-length cycles will be the subject of a future work.

Ex. 7: Consider $\mathsf{AF} = \langle \{b,c,d,e\}, \{(b,c), (c,d), (d,e), (e,c)\}\rangle$. The initial AF is cyclic, and it contains an odd-length cycle which will be removed and the final AF will contain only b.

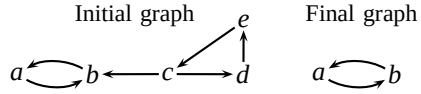


So, by applying Algorithm 1, we have $V = \{b\}$, $N = \{1\}$, with $\pi_1 = \{b\}$ and the following CP-net which represents the preferences of all players:

$b \succ \overline{b}$ B

So, G has one SPNE $\{b\}$ which corresponds to the preferred extension of the final AF. However, it does not correspond to the preferred extension of the initial AF which was the set $\{b,d\}$. If we consider that the odd-length cycle generally is a paradox, so that its arguments are not significant, we can consider that $\{b\}$ is a more realistic extension than $\{b,d\}$ (however this is not the approach chosen by the main semantics for acceptability).

Ex. 8: Consider $\mathsf{AF} = \langle \{a,b,c,d,e\}, \{(a,b), (b,a), (c,b), (c,d), (d,e), (e,c)\}\rangle$. The initial AF is cyclic, and it contains an odd-length cycle which will be removed and the final AF will contain only a and b.



By applying Algorithm 1, we have $V = \{a,b\}$, $N = \{1,2\}$, with $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ and the following CP-net which represents the preferences of all players:

b: $\overline{a} \succ a$
$\overline{b}$: $a \succ \overline{a}$
A B
a: $\overline{b} \succ b$
$\overline{a}$: $b \succ \overline{b}$

So, G has two SPNEs $\{a\overline{b}, \overline{a}b\}$ which correspond to the two preferred extensions of the final AF. However, they do not correspond to the preferred extension of the initial AF which was only the set $\{a\}$. In this case, to take into account the extension $\{b\}$ means that the attack $c \rightarrow b$ is considered as not significant (because an odd-length cycle generally is a paradox and its arguments are not significant; so, they cannot be able to provide a realistic attack against other arguments).

V. Some related works

The main related work for our paper concerns the link between argumentation and games identified by Dung: in [6],

an AF is used to solve a classical cooperative game (the stable marriage problem). He uses the arguments of AF to represent the possible issues of the game, and the attack relation to express the conflicts between issues. This link has also been used in [17] in order to prove the acceptability of an argument: a special dynamic game has been exhibited in which a player is the proponent and the second one is the opponent. The main differences with our work are first the static nature of our game, and secondly the number and the role of the players.

Mentioned in the introduction of this paper, the second important related work is the computation of preferred extensions. Some algorithms already exist (see for instance [11], [12], [13], [10]). It is important to note that our algorithms are not more efficient than the existing ones.

Another related work refers to the use of an argument as a literal in a propositional formula. This idea can also be found in [18], [19], [20] (for instance, in [19], a characterization of a preferred extension is given under the form of a propositional formula).

The last kind of related works concerns the treatment of the odd-length cycles. In the literature, distinct approaches exist: these cycles can appear in the AF, but are forbidden in the extensions (see for instance [21]), or these cycles can be accepted and treated as even-length cycles for computing the extensions (see for instance [22]). Our approach corresponds to the first case: odd-length cycles can appear in the AF but they will be removed for the computation of the extensions (see Algorithm 1).

VI. Conclusion

In this paper, we show how to translate an argumentation framework AF into a CP-Boolean game, and how this game allows to compute preferred extensions of the original AF using pure strategy Nash equilibria. We give three formal algorithms allowing respectively to transform the AF into a CP-Boolean game, and to compute the preferred extensions of AF. Moreover, we show that once odd-cycles are removed from AF, if the resulting argumentation framework AF' is acyclic, then the preferred extensions of AF' are computable in polynomial time.

Clearly, a limitation of our results is that we consider argumentation framework containing no odd-length cycles. We explained this choice by the fact that our study is preliminary and that such argumentation frameworks have some important properties. However it would be interesting to study these argumentation frameworks, because some odd-length cycles may make sense. So a future work will be to see if our results still apply in a more general case, and if it is not the case, how we can modify our algorithms to do so.

References

[1] P. Krause, S. Ambler, M. Elvang, and J. Fox, "A logic of argumentation for reasoning under uncertainty," *Computational Intelligence*, vol. 11 (1), pp. 113–131, 1995.

[2] H. Prakken and G. Vreeswijk, "Logics for defeasible argumentation," in *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, Eds. Kluwer Academic, 2002, vol. 4, pp. 218–319.

[3] A. Bondarenko, P. Dung, R. Kowalski, and F. Toni, "An abstract, argumentation-theoretic approach to default reasoning," *Artificial Intelligence*, vol. 93, pp. 63–101, 1997.

[4] L. Amgoud, N. Maudet, and S. Parsons, "Modelling dialogues using argumentation," in *Fourth International Conference on MultiAgent Systems (ICMAS'2000), Boston, MA, USA*, Jul. 2000, pp. 31–38.

[5] C. I. Chesñevar, A. G. Maguitman, and R. P. Loui, "Logical models of argument," *ACM Computing surveys*, vol. 32, no. 4, pp. 337–383, 2000.

[6] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, pp. 321–357, 1995.

[7] P. Harrenstein, W. van der Hoek, J.-J. Meyer, and C. Witteveen, "Boolean Games," in *Proceedings of the 8th International Conference on Theoretical Aspects of Rationality and Knowledge (TARK'01)*, J. van Benthem, Ed., vol. Theoretical Aspects of Rationality and Knowledge, San Francisco. Morgan Kaufmann, 2001, pp. 287–298.

[8] P. Harrenstein, "Logic in Conflict," Ph.D. dissertation, Utrecht University, 2004.

[9] E. Bonzon, M. Lagasquie-Schiex, J. Lang, and B. Zanuttini, "Compact preference representation and boolean games," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 1–35, 2009.

[10] C. Cayrol, S. Doutre, and J. Mengin, "On decision problems related to the preferred semantics for argumentation frameworks," *Journal of logic and computation*, vol. 13, pp. 377–403, 2003.

[11] S. Doutre and J. Mengin, "Preferred Extensions of Argumentation Frameworks: Computation and Query Answering," in *IJCAR 2001*, ser. LNAI, A. L. R. Goré and T. Nipkow, Eds., vol. 2083. Springer-Verlag, 2001, pp. 272–288.

[12] P. E. Dunne and T. J. Bench-Capon, "Complexity and combinatorial properties of argument systems," University of Liverpool, Department of Computer Science (U.L.C.S.), Technical report, 2001.

[13] ——, "Coherence in finite argument system," *Artificial Intelligence*, vol. 141, no. 1-2, pp. 187–203, 2002.

[14] S. Doutre, "Autour de la sémantique préférée des systèmes d'argumentation," Thèse, Université Paul Sabatier, IRIT, 2002.

[15] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets : A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements," *Journal of Artificial Intelligence Research*, vol. 21, pp. 135–191, 2004.

[16] ——, "Preference-Based Constrained Optimization with CP-nets," *Computational Intelligence*, vol. 20, no. 2, pp. 137–157, 2004, special Issue on Preferences.

[17] C. Cayrol, S. Doutre, and J. Mengin, "Dialectical Proof Theories for the Credulous Preferred Semantics of Argumentation Frameworks," in *Proc of ECSQARU*, 2001, pp. 668–679.

[18] N. Creignou, "The class of problems that are linearly equivalent to satisfiability or a uniform method for proving NP-completeness," *Theoretical Computer Science*, vol. 145, pp. 111–145, 1995.

[19] P. Besnard and S. Doutre, "Characterization of semantics for argument systems," in *Proc. of KR*, 2004, pp. 183–193.

[20] S. Coste-Marquis, C. Devred, and P. Marquis, "Constrained argumentation frameworks," in *Proc. of KR*, 2006, pp. 112–122.

[21] ——, "Prudent semantics for argumentation frameworks," in *Proc. of ICTAI*, 2005, pp. 568–572.

[22] P. Baroni, M. Giacomin, and G. Guida, "Scc-recursiveness: a general schema for argumentation semantics," *Artifical Intelligence*, vol. 168, pp. 162–210, 2005.