Argumentation and CP-Boolean games

Elise Bonzon* Caroline Devred†

Marie-Christine Lagasquie-Schiex‡

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.

1 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 $\langle$set of assumptions, conclusion$\rangle$, 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).

*LIPADE, Université Paris Descartes, France - elise.bonzon@parisdescartes.fr

†LERIA, Université d'Angers, France - caroline.devred@info.univ-angers.fr

‡IRIT, Université Paul Sabatier, France - lagasq@irit.fr

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 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 give a new way for computing preferred extensions, 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 2 and Section 3. Section 4 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 extensions of argumentation semantics. Section 5 gives an application of our approach on a classical example of decision making issued from [14]. Some related works are presented in Section 6 and Section 7 concludes.

Note that this paper is an extented version of [15]. This improvement essentially concerns the odd-length cycles which are taken into account if they are not minimal. So, new algorithms are given and new properties hold. We also apply our approach on a classical problem of decision making.

[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.

2 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:

Definition 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:

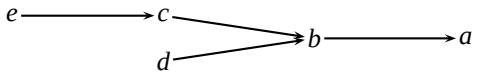
Definition 2 *S is* conflict-free *for* AF *iff there exists no* a_i, a_j *in S such that* $a_i \mathcal{R} a_j$. *An argument a is* acceptable w.r.t. *S for* AF *iff* $\forall b \in \mathcal{A}$ *such that* $b \mathcal{R} a$, $\exists c \in S$ *such that* $c \mathcal{R} b$. *S is* acceptable *for* AF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

Definition 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.

Example 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.

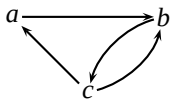
We will also need the notions of cycle and "elementary" cycle, as defined in [16]; in [16], an "elementary" cycle is a cycle which does not contain another cycle; however, this is not the classical definition of an "elementary" cycle in graph theory[2]. So, even if we use the same notion as that given in [16], we choose to use another word ("minimal" instead of "elementary") in order to avoid ambiguity with the concept given in graph theory:

[2] In graph theory, an elementary cycle is a cycle in which a vertex cannot appear twice (see [17]).

Definition 4 *Let $X = \{a_0, \ldots, a_n\}$ and $Y = \{b_0, \ldots, b_m\}$ be two non-empty sequences of arguments of $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$.*

- *X is a* cycle *if and only if $\forall i \leq n-1$, $a_i \mathcal{R} a_{i+1}$ and $a_n \mathcal{R} a_0$. $n+1$ is the length of X.*
- *Considering that X and Y are cycles, X* strictly contains *Y, denoted by $Y \subset X$, if and only if $m < n$ and $\exists i = 0 \ldots n$ such that $a_i = b_0$, $a_{(i+1) \text{ modulo } (n+1)} = b_1$, …, $a_{(i+m) \text{ modulo } (n+1)} = b_m$.*
- *X is a* minimal cycle *if and only if X is a cycle and $\nexists$ a cycle X' such that $X' \subset X$.*

Example 2 *In the following graph, the cycle $\{a, b, c\}$ is not minimal because it contains the minimal cycle $\{b, c\}$*[3].



We will need the following properties ([6, 12, 13, 18, 16]):

Proposition 1 *Let $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ such that $\mathcal{A} \neq \emptyset$.*

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, 18]).*
3. *If $\emptyset$ is the unique preferred extension then* AF *contains at least an odd-length cycle (see [12, 13, 18]).*
4. *If* AF *does not contain a minimal odd-length cycle then its preferred extensions are not empty (see [16])*[4]
5. *If* AF *does not contain a minimal odd-length cycle then each preferred extension is also stable*[5] *(see [16])*[6]

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, in particular, if these cycles are minimal. We agree with this opinion[7] and, in

[3] Note that, following the classical definition of an elementary cycle in graph theory, both cycles are elementary.

[4] Note that in [12, 13, 18], one can find a similar property concerning argumentation systems without odd-length cycles (minimal or not).

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

[6] Here too, a similar variant exists in [12, 13, 18] for argumentation systems without odd-length cycles (minimal or not).

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

this paper, we will consider that *argumentation frameworks should not contain minimal odd-length cycles*;[8] however, argumentation frameworks with even-length cycles will be taken into account.

As an argumentation framework without any minimal odd-length cycle is coherent, an interesting consequence occurs:

Corollary 1 *Let* $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an AF without minimal 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 $a \in E$. ■

3 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 $\bar{}$ 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 $M' \succeq M$.

3.1 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 [19, 20]. Although CP-nets generally consider variables with arbitrary finite domains, here we consider only "propositionalized" CP-nets, that is, CP-nets with binary variables.

Definition 5 *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* $x_1 y_2 z \succeq x_2 y_2 z$.

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

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:

Definition 6 *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 $p \in 2^{Pa(X_j)}$.*

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.

Example 3 *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 $D(W) = \{W_r, W_b\}$.*

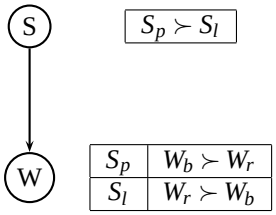


Figure 1: CP-net "My dinner"

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.

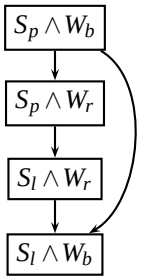


Figure 2: Preference graph induced by the CP-net "My dinner"

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.

3.2 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.

Definition 7 *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,[9] *i.e.*, $\{\pi_1, \ldots, \pi_n\}$ forms a partition of V.

Definition 8 *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.

Definition 9 *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].

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

Proposition 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 [19] 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.

4 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.

4.1 Translation of an argumentation framework into a CP-Boolean game

This transformation is done by Algorithm 1. However, the use of this algorithm assumes the existence of some other algorithms which make a kind of "precompilation" of the argumentation system to translate. The aim of this "precompilation" is the removal of the minimal odd-length cycles. This removal can be done in polynomial time if we choose to remove **all** the odd-length cycle, even if they are not minimal, or it will be done in exponential time if we remove **only** the minimal cycles:

- To remove all odd-length cycles, the two following algorithms can be used:
  - IsCyclic which returns true if there exists at least one cycle in the argumentation graph,[10]
  - RemOddCycles for removing the odd-length cycles if there are some of them in the AF.[11]

[10] 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).

[11] This algorithm is polynomial:

(Step 1) computation of the Boolean adjacency matrix corresponding to all shortest odd-length paths of attack (in term of length); 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 an elementary path – in the classical sense given by graph theory: a path in which each vertex appears only once – 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.

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.

- To remove only minimal odd-length cycles, Algorithm 4 given in Appendix can be used. This algorithm is less efficient than the previous ones (it is an exponential-time algorithm) but it allows the conservation of some odd-length cycles which make sense.

Let AF be an argumentation framework which does not contain minimal 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 $v \succ \bar{v}$ (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 $\bigvee_{w \in \mathcal{R}^{-1}(v)} w : \bar{v} \succ v$[12]); otherwise, if all variables $w \in \mathcal{R}^{-1}(v)$ are not satisfied, v can be satisfied (and so $\bigwedge_{w \in \mathcal{R}^{-1}(v)} \bar{w} : v \succ \bar{v}$).

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 all its odd-length cycles).

The use of this algorithm implies the following property:

Proposition 3 *Let* $\mathsf{AF} = \langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework without minimal odd-length cycle. Let* $G = (N, V, \pi, \Phi)$ *be the CP-Boolean game obtained from* AF *by applying Algorithm 1. s is a preferred extension of* AF *iff s is a SPNE for*[13] 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:

[12] The formula $w : \bar{v} \succ v$ (resp. $\bar{w} : \bar{v} \succ v$) 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.

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

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 without minimal odd-length cycle */

/* OUTPUTS: $G = (N, V, \pi, \Phi)$ a CP-Boolean game */

/* LOCAL VARIABLES: i = current agent, a = current argument */

/* computation of the CPTs for each argument */

for $a \in \mathcal{A}$ **do**

  if $\mathcal{R}^{-1}(a) = \emptyset$ **then** $CPT(a) = a \succ \overline{a}$ /* unattacked argument */

  else /* case of the other arguments */

    $CPT(a) = \{\bigvee_{v \in \mathcal{R}^{-1}(a)} v : \overline{a} \succ a\} \cup \{\bigwedge_{v \in \mathcal{R}^{-1}(a)} \overline{v} : a \succ \overline{a}\}$

/* 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 of AF */ /* associated with the CPTs of each argument */

/* computation of N, V, π and Φ */

$i = 1$

$N = \emptyset$

$V = \mathcal{A}$ /* 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 $(G = (N, V, \pi, \langle \mathcal{N}_1, \ldots, \mathcal{N}_{|V|} \rangle), \mathsf{AF})$

end

- $\mathcal{R}^{-1}(x_i) = \emptyset$ (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 $\mathcal{R}^{-1}(x_i) = \emptyset$ then $x_i \in s$. We have a contradiction.
- $\mathcal{R}^{-1}(x_i) \neq \emptyset$ (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 the conclusion obtained applying Conseq. 1 which says that $x_i \in s$ (AF is an argumentation framework without odd-length minimal 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.

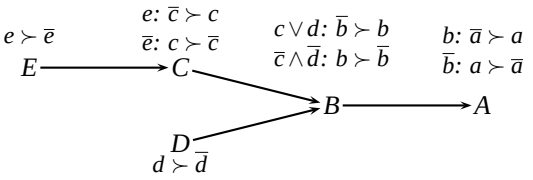
Let us study now the $\Leftarrow$ direction.

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. ■

Example 4 *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*[14]*:*



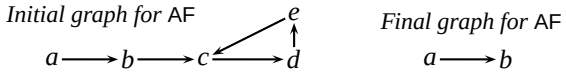
G has one SPNE $\{ed\overline{c}\overline{b}a\}$ *and* AF *has only one preferred extension* $\{e,d,a\}$.

[14] 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.

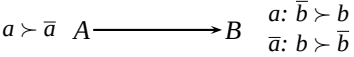
Example 5 *Consider* AF $= \langle \{a,b\}, \{(a,b), (b,a)\} \rangle$. *By applying Algorithm 1,* $V = \{a,b\}$ *and* $N = \{1,2\}$, *with* $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ *(*AF *is cyclic, but contains only even-length cycles, so Algorithm 1 can be applied). The following CP-net represents the preferences of all players:*

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

Example 6 *Consider* 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 a minimal odd-length cycle; so before to apply Algorithm 1, this cycle is removed. The final* AF *will contain only a and b.*

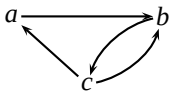


So, by applying Algorithm 1, we have $V = \{a,b\}$, $N = \{1,2\}$, *with* $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ *and the following CP-net which represents the preferences of all players:*

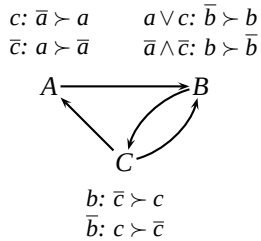


G has one SPNE $\{a\overline{b}\}$ *and the final* AF *(*after removal of minimal odd-length cycles*) has one preferred extension* $\{a\}$.

Example 7 *Consider* AF $= \langle \{a,b,c\}, \{(a,b), (b,c), (c,a), (c,b)\} \rangle$. *The initial* AF *is cyclic, and contains one not minimal odd-length cycle and one minimal even-length cycle. As the odd-length cycle is not minimal, we do not have to remove any cycle.*



So, by applying Algorithm 1, we have $V = \{a,b,c\}$, $N = \{1,2,3\}$, *with* $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$ *and the following CP-net which represents the preferences of all players:*



G has one SPNE $\{\overline{a}\overline{b}c\}$ *and* AF *has one preferred extension* $\{c\}$.

4.2 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:

Proposition 4 *Let* AF *be an argumentation framework without minimal odd-length cycles. Let G be the CP-Boolean game 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. ■

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 cycles except minimal odd-length cycles.

These algorithms assume the existence of:

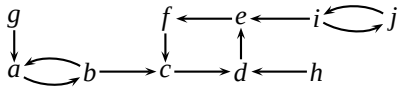
- Algorithm COMPINTCYCLEFORPROP which returns the cycle (or one of the cycles if there are several) in a given set of variables which permits to reach more variables as possible[15]
- Algorithm COMPINTVARFORPROP which returns a variable of the cycle which permits to reach more variables as possible[16].

For instance, on the following graph:

[15] This algorithm uses the notion of Boolean adjacency matrix as Algorithm REMODDCYCLES:

- computation of the Boolean adjacency matrix $\mathcal{M}^{ap}$ corresponding to all shortest paths in the graph reduced to the given set of variables: $\mathcal{M}^{ap} = \mathcal{M} + \mathcal{M}^2 + \mathcal{M}^3 + \ldots + \mathcal{M}^{2n}$ with $n = |V|$; $\mathcal{M}^{ap}(i)$ will denote $(\mathcal{M}^{ap}(i,1), \ldots, \mathcal{M}^{ap}(i,n))$;
- ToSee $= V$; $C = \emptyset$; end? =false;
- loop: while NOT(end?) do
  $v =$ top(ToSee); ToSee $=$ ToSee $\setminus \{v\}$;
  if ($\nexists w \in$ ToSee s.t. $\mathcal{M}^{ap}(v) \subset \mathcal{M}^{ap}(w)$) then
  /* no var. permitting to reach more var. than v */
  $C = C \cup \{v\}$;
  $\forall w \in$ ToSee do if $\mathcal{M}^{ap}(v) = \mathcal{M}^{ap}(w)$ then $C = C \cup \{w\}$;
  end? =true;
  else if ToSee is empty then end? =true;
- Return C

[16] This algorithm can be very simple: return the variable whose leaving degree is the greatest.



$\{a,b\}$ 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). And among the variables of the cycle $\{i,j\}$, the variable i (leaving degree[17] of $i = 2$) is more interesting than the variable j (leaving degree of $j = 1$).

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 (chosen by Algorithm CompIntVarForProp in order to make the propagation more efficient) instantiated to true, the second one contains this same variable instantiated to false; note that, when odd-length cycles exist in the graph, some instantiations are impossible (see Example 9); However, at this stage of the algorithm, we are not able to know what are the impossible instantiations; so, a checking process is needed at the end of the propagation process;
  - propagation of these instantiations for each one of these SPNEs as long as possible.
- checking process of each obtained instantiation in order to detect impossible instantiations due to the existence of non minimal odd-length cycles; the points which must be checked are exactly the same ones which are taken into account for propagating values:
  - either a variable and one of its parents belonging to the same instantiation,
  - or a variable not belonging to an instantiation without any of its parents belong to this instantiation.

[17] The leaving degree of a vertex in a graph is the number of edges leaving the vertex.

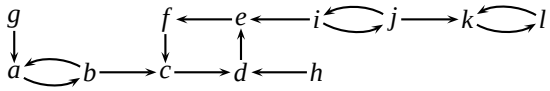
These problems can be appeared neither with an acyclic graph (because of the simplicity of the propagation process), nor with a graph containing even-length cycles (because in an even-length cycle, if a variable is instantiated to true, its parents are always instantiated to false and vice-versa).

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
```

Example 8 *Using the following graph:*



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); then we select in this cycle the most interesting variable (that which has the largest leaving degree); in this example and following this point of view, i and j are equivalent;*
- *the propagation process is restarted with the following current states of two SP-NEs: gh$\bar{a}\bar{d}$b$\bar{c}$i and gh$\bar{a}\bar{d}$b$\bar{c}\bar{i}$;*

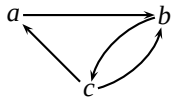
Algorithm 3: CompSPNERec: Recursive computation of SPNEs of a CP-Boolean game obtained from an argumentation framework

```
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, res = Boolean variable to check if the instanciation is
    correct */

    if R = ∅ then
        /* all variables are instantiated: a SPNE is perhaps found */
        /* checking process of the instantiation */
        res = true
        for v ∈ In do
            if Pa(v) ∩ In ≠ ∅ then res = false
        for v ∈ Out do
            if Pa(v) ⊆ Out then res = false
        if res then return {(In, complement(Out))}        /* a correct SPNE has been found */
    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 one 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 = CompIntVarForProp(G, R, 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
```

- *so, at the end of the propagation process, three instantiations are obtained* $gh\overline{a}\overline{d}b\overline{c}i\overline{e}f\overline{j}k\overline{l}$, $gh\overline{a}\overline{d}b\overline{c}i\overline{e}\overline{f}j\overline{k}l$ *and* $gh\overline{a}\overline{d}b\overline{c}i\overline{e}f\overline{j}\overline{k}l$. *Moreover, all these instantiations are considered correct by the checking process*[18]*, so they are SPNEs and these SPNEs correspond to the three preferred extensions* $\{g,h,b,e,j,l\}$, $\{g,h,b,f,i,k\}$ *and* $\{g,h,b,i,f,l\}$.

Example 9 *Using the following graph:*



the steps of the computation process are:

- *No variable is without parents, no propagation is possible. We must compute the most interesting cycle then the most interesting variable of this cycle; in this example, both cycles are interesting but the interest of variables differ:*
  - *a allows to reach the variable b*
  - *b allows to reach the variable c*
  - *c allows to reach the variable a and b*
  - *so, in terms of the effectiveness of the propagation, the most interesting variable will be c*
- *the propagation process is started with the following current states of two sets:* $\{c\}$ *and* $\{\overline{c}\}$*;*
- *so, at the end of the propagation process, two sets are obtained* $\{c\overline{a}\overline{b}\}$ *and* $\{\overline{c}a\overline{b}\}$. *The checking process begins:*
  - *For the first set, we have* $In = \{c\}$, $Out = \{a,b\}$. *We have* $Pa(c) \cap In = \emptyset$*;* $Pa(a) \not\subseteq Out$ *and* $Pa(b) \not\subseteq Out$. *We return* $\{\overline{a}\overline{b}c\}$
  - *For the second set, we have* $In = \{a\}$, $Out = \{b,c\}$. *But,* $Pa(c) \subseteq Out$. *So, we do not return anything.*

Thus, we obtain one SPNE $\overline{a}\overline{b}c$ *which corresponds to the preferred extension* $\{c\}$,

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

Proposition 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.*

[18] Note that, in this example, the checking process was useless because there is no odd-length cycle.

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:

- $Pa(x_i) = \emptyset$. We know from Algorithm 1 that we have $CPT(x_i) = x_i \succ \bar{x}_i$. As $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, we know than $s_i = \bar{x}_i$. But, we know from Algorithm 2 than $x_i \in s$. We have a contradiction.
- $Pa(x_i) \neq \emptyset$. We know from Algorithm 1 that we have $CPT(x_i) = \{\bigvee_{v \in Pa(x_i)} v : \bar{x}_i \succ x_i\} \cup \{\bigwedge_{v \in Pa(x_i)} \bar{v} : x_i \succ \bar{x}_i\}$. Several cases appear:
  - All variables in $Pa(x_i)$ are not satisfied: $\bigwedge_{v \in Pa(x_i)} \bar{v}$ holds and $Pa(x_i) \subseteq Out$. By $CPT(x_i)$, we know that $x_i \succ \bar{x}_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, we have $s_i = \bar{x}_i$ which is equivalent to $x_i \in Out$. But, we know from the checking test of Algorithm 3 that this case ($Pa(x_i) \subseteq Out$ and $x_i \in Out$) does not produce a strategy profile; so $s \notin SP$; that is in contradiction with the initial assumption.
  - At least one variable in $Pa(x_i)$ is satisfied: $\bigvee_{v \in Pa(x_i)} v$ holds and $Pa(x_i) \cap In \neq \emptyset$. By $CPT(x_i)$, we know that $\bar{x}_i \succ x_i$. So, as $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, $s_i = x_i$ which is equivalent to $x_i \in In$. But, we know from the checking test of Algorithm 3 that this case ($Pa(x_i) \cap In \neq \emptyset$ and $x_i \in In$) does not produce a strategy profile; so $s \notin SP$; that is in contradiction with the initial assumption.

So s is a SPNE for G.

Let us study now the $\Leftarrow$ direction.
Let s be a SPNE of the CP-Boolean game associated. Assume that $s \notin SP$. So, either $\exists x_i \in In$ such that $Pa(x_i) \cap In \neq \emptyset$ or $\exists x_i \in Out$ such that $Pa(x_i) \subseteq Out$. Let's study these both cases:

- $\exists x_i \in In$ (*i.e.* $s_i = x_i$) such that $Pa(x_i) \cap In \neq \emptyset$. As $Pa(x_i) \neq \emptyset$ (because In is not the empty set – In contains x_i – and its intersection with $Pa(x_i)$ is not empty), we know from Algorithm 1 that $CPT(x_i) = \{\bigvee_{v \in Pa(x_i)} v : \bar{x}_i \succ x_i\} \cup \{\bigwedge_{v \in Pa(x_i)} \bar{v} : x_i \succ \bar{x}_i\}$. As $\exists v \in Pa(x_i)$ such that $v \in In$, we know that $\bar{x}_i \succ_i x_i$. So there exists $s'_i = \bar{x}_i$ such that $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, so s is not a SPNE; that is in contradiction with the initial assumption.
- $\exists x_i \in Out$ (*i.e.* $s_i = \bar{x}_i$) such that $Pa(x_i) \subseteq Out$. At this point, there are two possibilities:
  - $Pa(x_i) = \emptyset$; so we know from Algorithm 1 that $CPT(x_i) = x_i \succ \bar{x}_i$. So there exists $s'_i = x_i$ such that $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, so s is not a SPNE; that is in contradiction with the initial assumption.
  - $Pa(x_i) \neq \emptyset$; so we know from Algorithm 1 that $CPT(x_i) = \{\bigvee_{v \in Pa(x_i)} v : \bar{x}_i \succ x_i\} \cup \{\bigwedge_{v \in Pa(x_i)} \bar{v} : x_i \succ \bar{x}_i\}$. As $Pa(x_i) \subseteq Out$, we have $\forall v \in Pa(x_i)$, $v \in Out$, and we know that $x_i \succ_i \bar{x}_i$. So there exists $s'_i = x_i$ such that $(s'_i, s_{-i}) \succ_i (s_i, s_{-i})$, so s is not a SPNE; that is in contradiction with the initial assumption.

Thus, $s \in SP$. ■

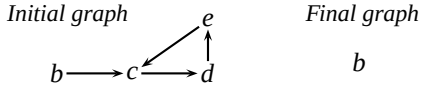
4.3 Managing minimal odd-length cycles

Of course, the removal of minimal 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, a minimal odd-length cycle may be considered as a paradox.

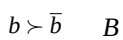
This initial assumption is also justified by the fact that some interesting argumentation systems make sense only without minimal odd-length cycles (see [16] and Section 5).

Moreover, the removal of this kind of cycles guarantees some important properties (see Prop. 1 and Conseq. 1).

Example 10 *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 a minimal odd-length cycle which will be removed and the final AF will contain only b.*

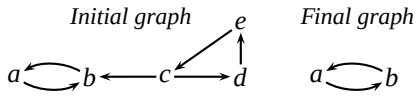


So, by applying Algorithm 1, we have $V = \{b\}$, $N = \{1\}$, *with* $\pi_1 = \{b\}$ *and the following CP-net which represents the preferences of all players:*



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 minimal 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).*

Example 11 *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 a minimal 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:*

So, G has two SPNEs $\{a\bar{b}, \bar{a}b\}$ 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 a minimal 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).*

5 An application in decision making: the omelet problem

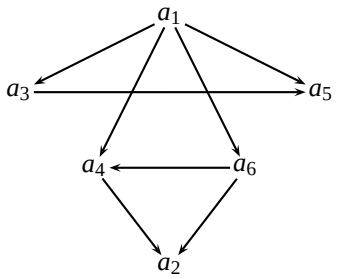
This application is a classical problem of decision making. In [16], Amgoud and al. have proposed to formalize it using a special argumentation system which does not contain minimal odd-length cycles.

In this example, an agent prepares an omelet and should decide whether or not to add an egg to a 5 eggs omelet knowing that the remaining egg may be rotten. The possible actions of this agents are: "to break the egg directly in the omelet", "to break the egg in a cup", "to throw away the egg". And the agent wants to realize different goals presented here in the order of their importance: "do not waste the omelet", "do not waste the 6th egg", "to have a 6 eggs omelet", "to avoid having a cup to wash".

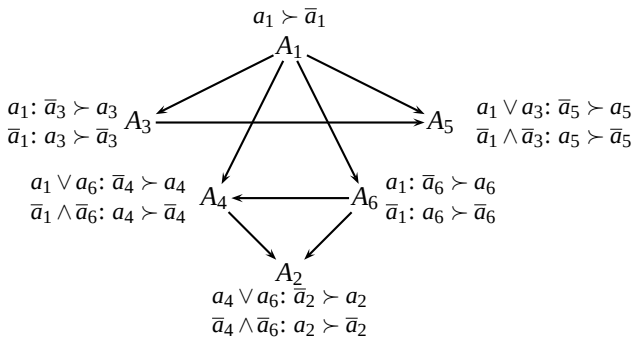
This problem can be described in argumentation by the following system proposed in [16]:

- the set of arguments $\mathcal{A}$ is composed with six arguments:
  - argument a_1 in favor of "to break the egg directly in the omelet" when the egg is not rotten
  - argument a_2 in favor of "to break the egg directly in the omelet" when the egg is rotten
  - argument a_3 in favor of "to break the egg in a cup" when the egg is not rotten
  - argument a_4 in favor of "to break the egg in a cup" when the egg is rotten
  - argument a_5 in favor of "to throw away the egg" when the egg is not rotten
  - argument a_6 in favor of "to throw away the egg" when the egg is rotten
- the attack relation on $\mathcal{A}$ is built using two constraints:
  - an argument in favor of an action x is in conflict with an argument in favor of another action y
  - there is a preference relation between the arguments obtained by the preference relation on the goals

Considering that the agent is quite sure that the egg is good, Amgoud and al proposes in [16] the following representation of this system:



This argumentation system does not have minimal odd-length cycles and it can be translated into the following CP-net following Algorithm 1:



And, using Algorithms 2 and 3, the SPNE of this CP-net is: $\{a_1 a_2 \overline{a}_3 \overline{a}_4 \overline{a}_5 \overline{a}_6\}$ which corresponds to the preferred extension $\{a_1, a_2\}$ which supports the action "to break the egg directly in the omelet". Of course, this result is consistent with that proposed by [16].

6 Some related works

The main related works for our paper concern 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). Dung 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 [21] 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; note that this use of dynamic games in argumentation is always a highly studied topic (see for instance [22, 23]). The main differences with our work are first the static nature of our game, and secondly the number and the role of the players.

Another proposition which mixes argumentation and game theory is the work made by Rahwan and Larson (see [24]). They transpose the notion of mechanism design of a game in argumentation; thus, for a given semantics, they propose a special mechanism for designing argumentation protocols adapted to this semantics. It is still a dynamic

and strategic view of argumentation which is very distant from ours: unlike their point of view, we are not interested by the way that the agents exchange their arguments but by the interactions between these arguments themselves.

Mentioned in the introduction of this paper, another important related works are about 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, but give another way to compute those extensions, and allow to use classical game theory concepts to do so.

Another related works refer to the use of an argument as a literal in a propositional formula. This idea can also be found in [25, 26, 27] (for instance, in [26], 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 [28]), or these cycles can be accepted and treated as even-length cycles for computing the extensions (see for instance [29]). Our approach corresponds to a particular subcase of the first case: *minimal* odd-length cycles can appear in the AF but they will be removed before the translation and the computation of the extensions (see Algorithms 1 and 4). Moreover, our algorithms allow to treat every argumentation framework corresponding to the proposition made by Amgoud and al ([16]), as, by construction, these argumentation systems contain no minimal odd-length cycle.

7 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 four 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 minimal odd-cycles are removed from AF, if the resulting argumentation framework 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 minimal odd-length cycle. We explained this choice by the fact that such argumentation frameworks have some important properties. However it would be interesting to study these argumentation frameworks, because some minimal odd-length cycles may make sense. So a future work will be to modify our algorithms for allowing the translation of any cyclic argumentation framework. Nevertheless, note that, in the current state, our work can already take into account the most frequently used argumentation frameworks (those that are coherent, as the one proposed by [16]).

References

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

[2] H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. In *Handbook of Philosophical Logic*, volume 4, pages 218–319. Kluwer Academic, 2002.

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

[4] L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In *Proc. of ICMAS*, pages 31–38, 2000.

[5] C. Chesñevar, A. Maguitman, and R. Loui. Logical models of argument. *ACM Computing surveys*, 32(4):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*, 77:321–357, 1995.

[7] P. Harrenstein, W. van der Hoek, JJ. Meyer, and C. Witteveen. Boolean Games. In *Proc. of TARK*, pages 287–298, 2001.

[8] P. Harrenstein. *Logic in Conflict*. PhD thesis, Utrecht University, 2004.

[9] E. Bonzon, MC. Lagasquie-Schiex, J. Lang, and B. Zanuttini. Compact preference representation and boolean games. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(1):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*, 13:377–403, 2003.

[11] S. Doutre and J. Mengin. Preferred Extensions of Argumentation Frameworks: Computation and Query Answering. In A. Leitsch R. Goré and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 272–288. Springer-Verlag, 2001.

[12] P. Dunne and T. Bench-Capon. Complexity and combinatorial properties of argument systems. Tech. report, U.L.C.S., 2001.

[13] P. Dunne and T. Bench-Capon. Coherence in finite argument system. *Artificial Intelligence*, 141(1-2):187–203, 2002.

[14] L. Savage. *The foundations of statistics*. Dover, New-York, 1972.

[15] E. Bonzon, C. Devred, and MC. Lagasquie-Schiex. Translation of an argumentation framework into a CP-Boolean game. In *Proc. of ICTAI*, pages 522–529. IEEE Computer Society, 2009.

[16] L. Amgoud, Y. Dimopoulos, and P. Moraïtis. Making decisions through preference-based argumentation. In *Proc. of KR*, pages 113–124, 2008.

[17] C. Berge. *Graphs and Hypergraphs*. North-Holland Mathematical Library, 1973.

[18] Sylvie Doutre. *Autour de la sémantique préférée des systèmes d'argumentation*. Thèse, Université Paul Sabatier, IRIT, 2002.

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

[20] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. Preference-Based Constrained Optimization with CP-nets. *Computational Intelligence*, 20(2):137–157, 2004.

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

[22] Phan Minh Dung and Phan Minh Thang. A unified framework for representation and development of dialectical proof procedures in argumentation. In *Proceedings of International Joint conference on Artificial Intelligence (IJCAI)*, Pasadena, California, USA, 2009. Springer Verlag.

[23] Phan Minh Thang, Phan Minh Dung, and Nguyen Duy Hung. Toward a common framework for dialectical proof procedure in abstract argumentation. *Journal of Logic and Computation*, 19(6):1071–1109, 2009.

[24] Iyad Rahwan and Kate Larson. Argumentation and game theory. In Iyad Rahwan and Guillermo Simari, editors, *Argumentation in Artificial Intelligence*, pages 321–339. Springer, 2009.

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

[26] P. Besnard and S. Doutre. Characterization of semantics for argument systems. In *Proc. of KR*, pages 183–193, 2004.

[27] S. Coste-Marquis, C. Devred, and P. Marquis. Constrained argumentation frameworks. In *Proc. of KR*, pages 112–122, 2006.

[28] S. Coste-Marquis, C. Devred, and P. Marquis. Prudent semantics for argumentation frameworks. In *Proc. of ICTAI*, pages 568–572, 2005.

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

Algorithm 4: Removal of minimal cycles in an argumentation system

```
begin
    /* INPUTS: AF = ⟨𝒜, ℛ⟩ an argumentation system */
    /* OUTPUTS: AF after removal of its minimal odd-length cycles */
    /* LOCAL VARIABLES:
        ℳ = Boolean adjacency matrix representing the transitive closure of ℛ,
        SC = set of minimal cycles of AF,
        C, C' = cycles (sequences of arguments),
        AF' = AF reduced to its cycles */
    /* USED SUBFUNCTIONS:
        TRANSITIVECLOSURE(AF) = function computing the transitive closure of ℛ
            and returning the corresponding Boolean adjacency matrix,
        REDUCTIONTOCYCLES(AF,ℳ) = function which returns the argumentation
            system corresponding to AF reduced to its cycles,
        DIJKSTRA(AF,x,y) = function returning the shortest non-empty path between
            two vertices x and y in AF,
        ADDCYCLETOSETOFCYCLES(SC,C) = function adding a cycle C to the set
            of cycles SC
        REMSUBAF(AF,C) = function which removes a cycle C to AF

    ℳ = TRANSITIVECLOSURE(AF)
    SC = ∅
    AF' = REDUCTIONTOCYCLES(AF,ℳ) /* AF' contains only the cycles of AF */
    for each vertex x of AF' do
        /* by construction of AF', x belongs at least to an cycle */
        C = DIJKSTRA(AF',x,x)          /* C is the shortest non-empty cycle containing
        x */
        if there exists no cycle C' ∈ SC such that C' ⊂ C then
            /* C is a minimal cycle */
            ADDCYCLETOSETOFCYCLES(SC, C)
    for each cycle of SC do
        if C is an odd-length cycle then
            /* C must be removed to AF /*
            AF = REMSUBAF(AF,C)
    return AF
end
```