International Journal of Approximate Reasoning 50 (2009) 899–914

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Dependencies between players in Boolean games

Elise Bonzon [a,*], Marie-Christine Lagasquie-Schiex [b], Jérôme Lang [c]

[a] Université Paris Descartes, CRIP5, 45 rue des Saints-Pères, F-75270 Paris Cedex 06, France
[b] Université de Toulouse, UPS, IRIT, 118 route de Narbonne, F-31062 Toulouse Cedex 9, France
[c] Université Paris-Dauphine, LAMSADE, F-75775 Paris Cedex 16, France

ARTICLE INFO

Article history:
Received 21 April 2008
Received in revised form 10 October 2008
Accepted 12 February 2009
Available online 27 February 2009

Keywords:
Game theory
Compact preference representation
Problem decomposition

ABSTRACT

Boolean games are a logical setting for representing static games in a succinct way, taking advantage of the expressive power and succinctness of propositional logic. A Boolean game consists of a set of players, each of them controlling a set of propositional variables and having a specific goal expressed by a propositional formula, or more generally a specification of the player's preference relation in some logical language for compact preference representation, such as prioritized goals. There is a lot of graphical structure hidden in a Boolean game: the satisfaction of each player's goal depends on players whose actions have an influence on her goals. Exploiting this dependency structure facilitates the computation of pure Nash equilibria, by partly decomposing a game into several sub-games that are only loosely related.

1. Introduction

Computing solution concepts for games is a challenging problem, and has been addressed in various places under various assumptions. In particular, as soon as the number of players is not small, or as soon as the strategy set of some players is combinatorial (which is typically the case when players control several variables), not only the computation of solution concepts is hard, but the *representation* (or specification) of the game itself is problematic, since the explicit representation of the utility matrix would be exponentially large. Boolean games [1–4] precisely address this issue. In their basic version, they allow for expressing in a compact way static games with binary preferences: each player of a Boolean game controls a set of propositional variables, and each player's preferences are expressed by a propositional formula.[1]

Bonzon et al. [4,5] give a semantical characterization of Nash equilibria in Boolean games, and identify the computational complexity of several issues, such as the existence of pure-strategy Nash equilibria in a Boolean game (both in the case of dichotomous preferences and in the case of non-dichotomous preferences expressed by means of CP-nets or prioritized goals). The conclusions are rather pessimistic: in the case of dichotomous preferences expressed by plain propositional formulas, the existence of a pure-strategy Nash equilibrium is Σ_2^p-complete.

However, these pessimistic results have to be tempered by the fact that in practical situations, there is a limited degree of interactions between players. This assumption that the dependencies between players are limited is at the heart of several frameworks, including *local-effect games* [6,7], where players may share some actions, and where the utility of a player depends only on the number of players who chose each action; and *graphical games* [10,9,11], where the representation

* Corresponding author.
E-mail addresses: elise.bonzon@mi.parisdescartes.fr (E. Bonzon), lagasq@irit.fr (M.-C. Lagasquie-Schiex), lang@irit.fr (J. Lang).

[1] We refer here to the version of Boolean games defined in [4], which generalizes the initial proposal [1]. Boolean games can easily be extended to allow for non-dichotomous preferences, represented in some compact language for preference representation (see Chapter 8 of [2] and [5,4]).

0888-613X/$ - see front matter © 2009 Published by Elsevier Inc.
doi:10.1016/j.ijar.2009.02.008

of the players' utilities is based on a dependency relation between variables and players: the utility of player i is described by a table specifying a numerical value for each combination of values to each of the set of variables that are relevant to i (see the concluding Section for more details).

In this paper we address a similar issue for Boolean games: using the syntactical nature of goals, we can express the dependencies between players under the form of a graph: if the satisfaction of a player i depends on some variables controlled by a player j, then i may need some action of j to see her goal satisfied. This intuitive notion of dependency between players and its graphical representation allow us to exploit the structure of such graphs so as to decompose Boolean games, and make the computation of pure-strategy Nash equilibria all the easier that the dependency graph is sparse. On the other hand, our results are still somehow preliminary, because they only pave the way towards designing and implementing efficient algorithms for computing pure-strategy Nash equilibria in Boolean games. Moreover, we do not consider mixed strategies at all.

For the sake of simplicity and presentation, we focus first on the dichotomous preferences, although, as we show in Section 5, our notions and results apply much more generally. We give the necessary background on Boolean games in Section 2. In Section 3 we define the dependency graph between players induced by a Boolean game, and study a few of its properties. In Section 4 we show how using this dependency graph may make the computation of pure Nash equilibria easier. In Section 5, we show how our notions and results can be reformulated and stated much more generally for Boolean games with non-dichotomous preferences represented in some language for compact preference representation. Related work and further issues are discussed in Section 6. Proofs are given in Appendix.

2. *n*-Player Boolean games

For any finite set $V = \{a, b, \ldots\}$ of propositional variables, L_V denotes the propositional language built up from V, the Boolean constants $\top$ and $\bot$, and the usual connectives. Formulas of L_V are denoted by φ, ψ, etc. A *literal* is a variable x of V (positive literal) or the negation of a variable (negative literal). If $\varphi \in L_V$, then $Var(\varphi)$ denotes the set of propositional variables appearing in φ.

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 $V' \subseteq V$. A V'-interpretation, also known as a partial interpretation, is a truth assignment to each variable of V'. V'-interpretations are denoted by listing all variables of V', with a $^-$ symbol when the variable is set to false: for instance, let $V' = \{a, b, d\}$, then the V'-interpretation $M = \{a, d\}$ assigning a and d to true and b to false is denoted by $a\bar{b}d$. If $\{V_1, \ldots, V_p\}$ is a partition of V and $\{M_1, \ldots, M_p\}$ are partial interpretations, where $M_i \in 2^{V_i}$, $(M_1, \ldots, M_p)$ denotes the interpretation $M_1 \cup \cdots \cup M_p$.

The partial instantiation of a formula φ by an X-interpretation M_X is the formula $(\varphi)_{M_X}$ obtained from φ by instantiating all positively (resp. negatively) instantiated atoms in M_X by $\top$ (resp. $\bot$). For instance, if $\varphi = (a \wedge \neg b) \leftrightarrow (c \vee d))$, $X = \{a, d\}$ and $M_X = a\bar{d}$, then $(\varphi)_{M_X}$ is equivalent to $\neg b \leftrightarrow c$.

As usual, $\models$ denotes both satisfaction of a formula by an interpretation ($M \models \varphi$) and the classical consequence relation ($\varphi \models \psi$). If M is a partial interpretation of $Var(\varphi)$, we write $M \models \varphi$ if φ is satisfied by every interpretation for $Var(\varphi)$ which agrees with M; equivalently, $M \models \varphi$ if the conjunction of all literals assigned true by M logically entails φ. Due to this equivalence, we use the same notation for entailment and satisfaction, as it is standard in propositional logic.

Finally, given $M \in 2^V$, switch(M, x) denotes the interpretation obtained by switching the value of x in M, and leaving the values of other variables unchanged.

Given a set of propositional variables V, a Boolean game on V is an n-player game[2] where the actions available to each player consist in assigning a truth value to each variable in a given subset of V. The preferences, or goals, of each player i are represented by a propositional formula φ_i formed upon the variables in V. Thus, a player in a Boolean game has a dichotomous preference relation: either her goal is satisfied or it is not. This restriction is of course an important loss of generality, and may appear at first glance unreasonable. However, first note that many concrete situations can be modelled as games where agents have dichotomous preferences: see for instance the kidney exchange problem in [8]. Second (and more importantly), the results and notions we give in the paper hold for more general Boolean games where preferences are non-dichotomous (see Section 5). We choose to focus first on the case of dichotomous preferences for the sake of the exposition.

Definition 1. An *n-player Boolean game* is a 4-uple (N, V, π, Φ), where

- $N = \{1, 2, \ldots, n\}$ is a finite set of players (also called agents);
- V is a finite set of propositional variables;
- $\pi : N \mapsto 2^V$ is a control assignment function mapping each player to the variables she controls. For the ease of notation, the set of all the variables controlled by i is written π_i instead of $\pi(i)$. Each variable is controlled by one and only one agent, that is, $\{\pi_1, \ldots, \pi_n\}$ forms a partition of V;
- $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ is a set of goals, where each φ_i is a satisfiable formula of L_V.

[2] We refer here to the definition of Boolean games as in [4]. See this paper for the relationship to [1,2].

Definition 2. Let $G = (N, V, \pi, \Phi)$ be a Boolean game.

A *strategy* for player i in G is a π_i-interpretation. The set of strategies for player i in G is $S_i = 2^{\pi_i}$.

A *strategy profile* s for G is a n-uple $s = (s_1, s_2, \ldots, s_n)$ where for all $i \in N$, $s_i \in S_i$. $S = S_1 \times \cdots \times S_n$ is the set of all strategy profiles.

Note that since $\{\pi_1, \ldots, \pi_n\}$ forms a partition of V, a strategy profile s is an interpretation for V, i.e., $s \in 2^V$. The following notations are usual in game theory. A *coalition* is a subset of N. Let $s = (s_1, \ldots, s_n)$ be a strategy profile. For any non-empty coalition $I \subseteq N$, the projection of s on I is defined by $s_I = (s_i)_{i \in I}$ and $s_{-I} = s_{N \setminus I}$. If $I = \{i\}$, we denote the projection of s on $\{i\}$ by s_i instead of $s_{\{i\}}$; similarly, we note s_{-i} instead of $s_{-\{i\}}$. π_I denotes the set of the variables controlled by I, and $\pi_{-I} = \pi_{N \setminus I}$. The set of strategies for $I \subseteq N$ is $S_I = \times_{i \in I} S_i$. If s and s' are two strategy profiles, (s_{-I}, s'_I) denotes the strategy profile obtained from s by replacing s_i with s'_i for all $i \in I$.

The goal φ_i of player i is a compact representation of a dichotomous preference relation, or equivalently, of a binary utility function $u_i : S \to \{0, 1\}$ defined by $u_i(s) = 0$ if $s \models \neg\varphi_i$ and $u_i(s) = 1$ if $s \models \varphi_i$. s is at least as good as s' for i, denoted by $s \succeq_i s'$, if $u_i(s) \geqslant u_i(s')$, or equivalently, if $s \models \neg\varphi_i$ implies $s' \models \neg\varphi_i$; s is strictly better than s' for i, denoted by $s \succ_i s'$, if $u_i(s) > u_i(s')$, or, equivalently, $s \models \varphi_i$ and $s' \models \neg\varphi_i$.

3. Dependencies between players

The syntactical expression of goals suggests to associate with each player the set of propositional variables that may have an influence on the satisfaction of her goal, which in turn allows for defining the set of players her goal depends on. Obviously, if the goal φ_i of player i does not mention any variable controlled by player j, then the satisfaction of i does not depend directly on j. This is only a sufficient condition: it may be the case that the syntactical expression of φ_i mentions a variable controlled by j, but that this variable plays no role whatsoever in the satisfaction of φ_i, as variable y in $\varphi_i = x \wedge (y \vee \neg y)$. We therefore use the notion of formula-variable independency from [12]:

Definition 3 [12]. A propositional formula φ is *independent from* a propositional variable x if there exists a formula ψ logically equivalent to φ and in which x does not appear.[3] The set of all variables on which φ depends is denoted by $DepVar(\varphi)$. A *normalization* of a propositional formula φ is a propositional formula ψ which does not contain any redundant variable, that is, such that (a) φ and ψ are logically equivalent and (b) $Var(\psi) = DepVar(\varphi)$.

Definition 4. Let $G = (N, V, \pi, \Phi)$ be a Boolean game. The set of *relevant variables* for a player i, denoted by $RV_G(i)$, is the set of all variables $v \in V$ such that φ_i is not independent from v.

For the sake of notation, the set of relevant variables for a player i in a given Boolean game G will be denoted by RV_i instead of $RV_G(i)$. We are now in position to define the *relevant players* for a given player i as the set of players controlling at least one variable of RV_i.

Definition 5. Let $G = (N, V, \pi, \Phi)$ be a Boolean game. The set of *relevant players* for a player i, denoted by RP_i,[4] is the set of agents $j \in N$ such that j controls at least one relevant variable of i: $RP_i = \bigcup_{v \in RV_i} \pi^{-1}(v)$.

Example 1. Three friends (1, 2 and 3) are invited at a party. 1 wants to go to this party. 2 wants to go to the party if and only if 1 does, whereas 3 wants to go there, wants 2 to goes and 1 not to. This situation can be modelled by the following Boolean game $G = (N, V, \pi, \Phi)$, defined as follows:

- $V = \{a, b, c\}$, with a (resp. b, c) meaning "1 (resp. 2, 3) goes to the party";
- $N = \{1, 2, 3\}$,
- $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$,
- $\varphi_1 = a$, $\varphi_2 = a \leftrightarrow b$ and $\varphi_3 = \neg a \wedge b \wedge c$.

We can see that 1's satisfaction depends only on herself, 2's depends on 1 and herself, whereas 3's depends on 1, 2 and herself. So, we have: $RV_1 = \{a\}$, $RV_2 = \{a, b\}$, $RV_3 = \{a, b, c\}$, $RP_1 = \{1\}$, $RP_2 = \{1, 2\}$, $RP_3 = \{1, 2, 3\}$.

This relation between players can be seen as a directed graph containing a vertex for each player, and an edge from i to j whenever $j \in RP_i$, i.e., if j is a relevant player of i.

Definition 6. Let $G = (N, V, \pi, \Phi)$ be a Boolean game. The *dependency graph of a Boolean game* G is the directed graph $\mathscr{P} = \langle N, R \rangle$, with $\forall i, j \in N$, $(i, j) \in R$ (denoted by $R(i, j)$) if $j \in RP_i$.

[3] We have this equivalent semantical characterization of formula-variable independency [12]: φ is dependent from x if there exists an interpretation s such that $s \models \varphi$ and $\text{switch}(s, x) \models \neg\varphi$.

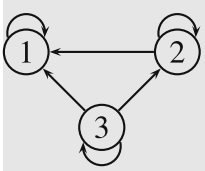
[4] Again, the set of relevant players for a Boolean game G should be denoted by $RP_G(i)$: for the ease of notation we simply write RP_i.

Note that $\mathscr{P}$ is not necessarily acyclic.

$R(i)$ is the set of players who may, by performing some action, influence the satisfaction of i's goal: $j \in R(i)$ if and only if $j \in RP_i$. Remark however that this is a weak notion of dependency: there are some cases where $j \in R(i)$ and yet i does not need any action by j to see to it that her goal satisfied. For instance, let $\pi_1 = \{a\}$, $\pi_2 = \{b\}$ and $\varphi_1 = a \vee b$. We have $2 \in R(1)$, yet 1 has a strategy to see her goal satisfied (namely, setting a to true) and therefore does not need an action by 2. Our notion of independency is too weak to take this into account: 1 depends on 2 just because there is no equivalent formula of φ_1 in which b does not appear, and in spite of that 1 does not *need* 2.[5]

We denote by R^* the transitive closure of R. $R^*(i,j)$ means that there exists a path from i to j in R. Then, $R^*(i)$ can be interpreted as the set of all players who have a direct or indirect influence on i, and $R^{*-1}(i)$ as the set of all players on which i has a direct or indirect influence.

Example 1 (*continued*). The dependency graph $\mathscr{P}$ induced by G is depicted as follows:



- $R(1) = \{1\}$, $R(2) = \{1,2\}$, $R(3) = \{1,2,3\}$.
- $R^{-1}(1) = \{1,2,3\}$, $R^{-1}(2) = \{2,3\}$, $R^{-1}(3) = \{3\}$.
- $R^*(1) = \{1\}$, $R^*(2) = \{1,2\}$ and $R^*(3) = \{1,2,3\}$.
- $R^{*-1}(1) = \{1,2,3\}$, $R^{*-1}(2) = \{2,3\}$ and $R^{*-1}(3) = \{3\}$.

We remark that every directed graph on set of players N is the graph induced by some Boolean game. Indeed, for every dependency graph $\mathscr{P} = \langle N, R\rangle$, we can construct the Boolean game $G = (N, V, \pi, \Phi)$, where $V = \{v_1, \ldots, v_n\}$, $\forall i \in N$, $\pi_i = \{v_i\}$, and $\forall i \in N$, $\forall j$ such that $j \in R(i)$, $\varphi_i = \bigwedge_j v_j$. If $\nexists j$ such that $j \in R(i)$, then $\varphi_i = \top$.

We now introduce the notion of stable set. A stable set is a subset B of players whose goal does not depend on players outside it.[6]

Definition 7. Let $G = (N, V, \pi, \Phi)$ be a Boolean game. $B \subseteq N$ is *stable* for R if and only if $R(B) \subseteq B$, i.e., $\forall j \in B$, $\forall i$ such that $i \in R(j)$, then $i \in B$.

Clearly, $\varnothing$ and N are stable, and the set of stable sets for a Boolean game is closed under union and intersection. These four properties actually fully characterize the set of coalitions that correspond to the set of stable sets for a Boolean game. This result is not crucial for the rest of the paper but it sheds some light on the meaning of stable sets.

Proposition 1. *Let $\mathscr{C} \subset 2^N$. There exists a Boolean game G such that $\mathscr{C}$ is the set of stable sets for G if and only if $\mathscr{C}$ satisfies the following four properties:*

(1) $\varnothing \in \mathscr{C}$;
(2) $N \in \mathscr{C}$;
(3) If $B, B' \in \mathscr{C}$ then $B \cup B' \in \mathscr{C}$;
(4) If $B, B' \in \mathscr{C}$ then $B \cap B' \in \mathscr{C}$.

We now define the projection of a Boolean game G on the set of players $B \subseteq N$ in order to decompose a Boolean game into several sub-games:

Definition 8. Let $G = (N, V, \pi, \Phi)$ be a Boolean game, and let $B \subseteq N$ be a stable set for R. The *projection* of G on B is defined by $G_B = (B, V_B, \pi_B, \Phi_B)$, where $V_B = \cup_{i \in B} \pi_i$, $\pi_B : B \to V_B$ such that $\pi_B(i) = \{v | v \in \pi_i\}$, and $\Phi_B = \{\psi_i \mid i \in B\}$, where for every $i \in B$, ψ_i is a normalization of φ_i.

The projection of a Boolean game on a stable set of players is a Boolean game:

Proposition 2. *If B is a stable set, then $G_B = (B, V_B, \pi_B, \Phi_B)$ is a Boolean game.*

As shown on the following example, this proposition allows us to decompose a Boolean game into several smaller Boolean games.

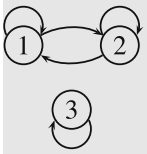
Example 2. Let $G = (N, V, \pi, \Phi)$ be the Boolean game defined by

[5] We could work out a stronger notion of dependency, which would be closer to a notion of "i needs j", in which 1 would not depend on 2 in our current example. Note that this stronger notion of dependency, which has an abductive flavour, is much harder to compute than the one developed in this paper. This is a very interesting topic for further research.

[6] Note that the notion of stable set defined here is different from the usual notion of stable set in graph theory.

- $V = \{a, b, c\}$,
- $N = \{1, 2, 3\}$,
- $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$,
- $\varphi_1 = a \leftrightarrow b$, $\varphi_2 = a \leftrightarrow \neg b$ and $\varphi_3 = \neg c$.

We have: $RV_1 = \{a, b\}$, $RV_2 = \{a, b\}$, $RV_3 = \{c\}$, $RP_1 = \{1, 2\}$, $RP_2 = \{1, 2\}$, $RP_3 = \{3\}$. The dependency graph $\mathscr{P}$ of G follows. The sets of players $B = \{1, 2\}$ and $C = \{3\}$ are stable. We can decompose G in 2 independent Boolean subgames:



- $G_B = (B, V_B, \pi_B, \Phi_B)$, with $B = \{1, 2\}$, $V_B = \{a, b\}$, $\pi_1 = a$, $\pi_2 = b$, $\varphi_1 = a \leftrightarrow b$, $\varphi_2 = a \leftrightarrow \neg b$.
- $G_C = (C, V_C, \pi_C, \Phi_C)$, with $C = \{3\}$, $V_C = \{c\}$, $\pi_3 = c$, $\varphi_3 = \neg c$.

Note that Proposition 2 no longer holds when B is not stable. In Example 2, take $B = \{1, 3\}$, then $G_{\{1,3\}} = \langle \{1, 3\}, \{a, c\}, (\pi_1, \pi_3), (a \leftrightarrow b, \neg c) \rangle$ is not a Boolean game, because φ_3 uses a variable (b) which is not in $V_{\{1,3\}} = \{a, c\}$.

4. Computing Nash equilibria

Pure-strategy Nash equilibria (PNE) for n-player Boolean games are defined exactly as usual in game theory (see for instance [13]), having in mind that utility functions are induced from players' goals $\varphi_1, \ldots, \varphi_n$. A PNE is a strategy profile such that each player's strategy is an optimal response to the other players' strategies.

Definition 9. Let $G = (N, V, \pi, \Phi)$ be a Boolean game with $N = \{1, \ldots, n\}$. $s = \{s_1, \ldots, s_n\}$ is a *pure-strategy Nash equilibrium* (PNE) if and only if $\forall i \in \{1, \ldots, n\}$, $\forall s'_i \in S_i$, $(s_i, s_{-i}) \succeq_i (s'_i, s_{-i})$.

The following simple characterization of PNEs is straightforward from this definition

Proposition 3 ([4], Proposition 2). *A strategy profile s is a pure-strategy Nash equilibrium for G iff for all $i \in N$, either $s \models \varphi_i$ or $s_{-i} \models \neg\varphi_i$ holds.*

These definitions lead to some obvious properties of pure-strategy Nash equilibria. If a player does not control any of her relevant variables, she has no influence on her own goal, and thus has no preference over her strategies. And if all players are in the same case, all strategy profiles are PNEs.

Proposition 4. *Let G be a Boolean game. If $i \notin RP_i$ holds for every $i \in N$, then every $s \in S$ is a PNE.*

If each player of a Boolean game depends only on a single player, then players such that $RP_i = \{i\}$ will be the only ones having an influence on their own goals. A strategy profile s will be a PNE if and only if it satisfies the preferences of these players.

Proposition 5. *Let G be a Boolean game such that $\forall i \in N$, $|RP_i| = 1$. s is a PNE if and only if $\forall i \in N$ such that $RP_i = \{i\}$, $s \models \varphi_i$.*

If the irreflexive part of the players' dependency graph $\mathscr{P}$ of a game G is acyclic, (i.e., if there is no cycle of length $\geqslant 2$), then we can use a procedure inspired by the "forward sweep procedure" [14] to find the pure-strategy Nash equilibria. Let us see this on an example.

Example 1 (*continued*). The irreflexive part of the dependency graph $\mathscr{P}$ of G is acyclic.

$RP_1 = \{1\}$, so a strategy profile $s = (s_1, s_2, s_3)$ is a PNE only if 1's goal is satisfied, i.e., $s_1 = a$.

Given 1's strategy $s_1 = a$, 2 has a best response (namely, $s_2 = b$), because her goal depends only on the variables controlled by 1 and herself. Finally, given the strategies of 1 and 2, 3's goal will not be satisfied whatever she does, therefore 3 has two best responses, namely $s_3 = c$ and $s_3 = \neg c$. Therefore, G has 2 PNEs: $\{abc, ab\overline{c}\}$.

Proposition 6. *Let G be a Boolean game such that the irreflexive part of the dependency graph $\mathscr{P}$ of G is acyclic.*

Then, G has at least one PNE. Moreover, s is a PNE of G if and only if for every $i \in N$, either $(s_i, s_{R^*(i)\setminus\{i\}}) \models \varphi_i$ or $s_{R^*(i)\setminus\{i\}} \models \neg\varphi_i$.

The computation of the set of PNEs of a Boolean game G such that the irreflexive part of its dependency graph is acyclic is done by Algorithms 1–3.

Algorithm 1: CompPNEAcycl: Computation of PNEs of a Boolean game such that the irreflexive part of the dependency graph is acyclic

begin

/* INPUT: $G = (N, V, \pi, \Phi)$ a Boolean game, $\mathcal{P} = \langle N, R \rangle$ the dependency graph associated (the irreflexive part of $\mathcal{P}$ is acyclic)*/
/* OUTPUTS: a set of PNEs S */
/* LOCAL VARIABLES: $I =$ set of players controlling variables already instantiated, $T =$ set of players controlling variables remaining to be instantiated */

/* Initialization */
$I = \varnothing$, $T = N$, $S = \{\top\}$
return CompPNEAcyclRec$(G, \mathcal{P}, I, T, S)$

end

Algorithm 2: CompPNEAcyclRec: Recursive computation of PNEs of a Boolean game such that the irreflexive part of the dependency graph is acyclic

begin

/* INPUT: $G = (N, V, \pi, \Phi)$ a Boolean game, $\mathcal{P} = \langle N, R \rangle$ the dependency graph associated (the irreflexive part of $\mathcal{P}$ is acyclic), I the set of players set of players controlling variables already instantiated, T the set of players controlling variables remaining to be instantiated, S the set of partial PNEs already computed*/
/* OUTPUTS: a set of PNEs S */
/* LOCAL VARIABLES: $PI =$ set of players we can satisfied, $i =$ current player*/

/* Initialization */
$PI = \varnothing$
for $i \in T$ **do**
  if $R(i) \subseteq I \cup \{i\}$ **then** $PI = PI \cup \{i\}$
    /* Variables controlled by players in RP_i are already instantiated */
for $i \in PI$ **do**
  /* Instantiation of variables controlled by i */
  $S =$ CompStratPlay(G, i, S)
  $I = I \cup \{i\}$
  $T = T \setminus \{i\}$
if $T = \varnothing$ **then**
return S /* All variables are instantiated */
else return CompPNEAcyclRec$(G, \mathcal{P}, I, T, S)$

end

Algorithm 3: CompStratPlay: Computation of strategies of a player allowing to satisfy her goal

```
begin
    /* INPUT: G = (N, V, π, Φ) a Boolean game, i a player, S the set of partial
    PNEs already computed*/
    /* OUTPUTS: a set of partial PNEs SP */
    /* LOCAL VARIABLES: s_i = instantiation of variables controlled by i, s =
    current partial PNE of S*/

    /* Initialization */
    SP = ∅
    for s ∈ S do
        if s ⊨ ¬φ_i then
            /* i cannot satisfy her goal, each one of her strategy belongs to a
            PNE*/
            for s_i ∈ 2^{π_i} do SP = SP ∪ {s ∪ s_i}
        else
            for s_i ∈ 2^{π_i} do
                /* i can satisfy her goal, and each one of her strategy satisfying it
                belongs to a PNE*/
                if s_i ⊨ (φ_i)_s then
                    SP = SP ∪ {s ∪ s_i}
    return SP
end
```

Example 1 (*continued*). In this example, G has 2 PNEs: $\{abc, ab\bar{c}\}$, and we have $R^*(1) = \{1\}$, $R^*(2) = \{1,2\}$, $R^*(3) = \{1,2,3\}$. For $s = abc$, we have

- $(s_{R^*(1)\setminus\{1\}}, s_1) = s_1 \models \varphi_1 = a$,
- $(s_{R^*(2)\setminus\{2\}}, s_2) = (s_1, s_2) \models \varphi_2 = a \leftrightarrow b$,
- $s_{R^*(3)\setminus\{3\}} = (s_1, s_2) \models \neg\varphi_3 = a \vee \neg b \vee \neg c$.

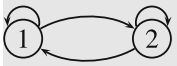
A similar line of reasoning holds for $s = ab\bar{c}$.

However, when the irreflexive part of the dependency graph is not acyclic, the existence of PNE is no longer guaranteed, as we can see on the following example.

Example 3. Let $G = (N, V, \pi, \Phi)$ be the Boolean game defined by $V = \{a, b\}$, $N = \{1, 2\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\varphi_1 = a \leftrightarrow b$ and $\varphi_2 = (a \leftrightarrow \neg b)$.

We have: $RV_1 = \{a, b\}$, $RV_2 = \{a, b\}$, $RP_1 = \{1, 2\}$, $RP_2 = \{1, 2\}$.

The dependency graph $\mathscr{P}$ of G is the following:

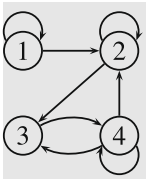


This game has no PNE

However, as shown in Example 4, a game with a cyclic dependency graph may have a PNE.

Example 4. Let $G = (N, V, \pi, \Phi)$ be the Boolean game defined by $V = \{a, b, c, d\}$, $N = \{1, 2, 3, 4\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$, $\pi_4 = \{d\}$, $\varphi_1 = a \leftrightarrow b$, $\varphi_2 = b \leftrightarrow c$, $\varphi_3 = \neg d$, and $\varphi_4 = d \leftrightarrow (b \wedge c)$. We have: $RP_1 = \{1, 2\}$, $RP_2 = \{2, 3\}$, $RP_3 = \{4\}$, $RP_4 = \{2, 3, 4\}$.

The dependency graph $\mathscr{P}$ of G is the following:

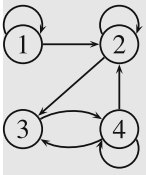


G has 2 PNEs: $\{abcd, \overline{a}\overline{b}\overline{c}\overline{d}\}$

The following proposition shows that if a strategy profile s is a PNE of a Boolean game G, and if B is a stable set, then the restriction of s to the variables controlled by players in B is a PNE of the projection of G on B.

Proposition 7. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game, let $B \subseteq N$ be a stable set for R, and let G_B be the projection of G on B. If s is a PNE for G, then s_B is a PNE for G_B.*

Example 4 (*continued*). Let us recall the dependency graph $\mathscr{D}$ of G:



The set of players $B = \{2, 3, 4\}$ is stable. $G_B = (B, V_B, \pi_B, \Phi_B)$ is a Boolean game, with $V_B = \{b, c, d\}$, $\pi_2 = b$, $\pi_3 = c$, $\pi_4 = d$, $\varphi_2 = b \leftrightarrow c$, $\varphi_3 = \neg d$, and $\varphi_4 = d \leftrightarrow (b \wedge c)$.
G has 2 PNEs: $\{abcd, \overline{a}\overline{b}\overline{c}\overline{d}\}$.
$\{bcd, \overline{b}\overline{c}\overline{d}\}$ are 2 PNEs of G_B (and in this case, G_B has no other PNEs).

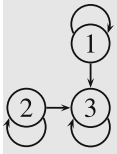
As we can see on Example 2, the converse is not always true: $C = \{3\}$ is stable, and the Boolean game $G_C = (C, V_C, \pi_C, \Phi_C)$ has a PNE: $\{\overline{c}\}$, but the game G has no PNE.

However, there exist simple cases for which the converse is true, and for which it will be easier to compute pure-strategy Nash equilibrium by decomposing the initial Boolean game.

Proposition 8. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game. Let B and C be two stable sets of players, and let G_B and G_C be the two associated Boolean games.*

Suppose than s_B is a PNE for G_B and s_C is a PNE for G_C such that $\forall i \in B \cap C$, $s_{B,i} = s_{C,i}$, where $s_{B,i}$ (resp. $s_{C,i}$) represents the strategy of player i for the game G_B (resp. G_C). Then, $s_{B\cup C}$ is a PNE for $G_{B\cup C}$.

Example 5. Let $G = (N, V, \pi, \Phi)$ be the Boolean game defined by $V = \{a, b, c\}$, $N = \{1, 2, 3\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$, $\varphi_1 = a \leftrightarrow c$, $\varphi_2 = b \leftrightarrow \neg c$, and $\varphi_3 = c$. We have: $RP_1 = \{1, 3\}$, $RP_2 = \{2, 3\}$, $RP_3 = \{3\}$. The dependency graph $\mathscr{D}$ of G is drawn below. The sets of players $B = \{1, 3\}$ and $C = \{2, 3\}$ are stable. We have two new Boolean games.



- $G_B = (B, V_B, \pi_B, \Phi_B)$, with $B = \{1, 3\}$, $V_B = \{a, c\}$, $\pi_1 = a$, $\pi_3 = c$, $\varphi_1 = a \leftrightarrow c$ and $\varphi_3 = c$. G_B has one PNE: $\{ac\}$ (denoted by $s_B = (s_{B,1}, s_{B,3})$).
- $G_C = (C, V_C, \pi_C, \Phi_C)$, with $C = \{2, 3\}$, $V_C = \{b, c\}$, $\pi_2 = b$, $\pi_3 = c$, $\varphi_2 = b \leftrightarrow \neg c$, $\varphi_3 = c$. G_C has one PNE: $\{\overline{b}c\}$ (denoted by $s_C = (s_{C,2}, s_{C,3})$).

$B \cap C = \{3\}$ and we have $s_{B,3} = s_{C,3} = c$, so $G_{B\cup C}$ has one PNE: $\{a\overline{b}c\}$.

We can easily generalize Proposition 8, with p stable sets covering the set of players:

Proposition 9. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game, and let $B_1 \cdots B_p$ be p stable sets of players, such that $B_1 \cup \cdots \cup B_p = N$. Let $G_{B_1}, \ldots, G_{B_p}$ be the p Boolean games associated.*

If $\exists s_{B_1} \cdots s_{B_p}$ PNEs of $G_{B_1}, \ldots, G_{B_p}$ such that $\forall i, j \in \{1, \ldots, p\}$, $\forall k \in B_i \cap B_j$, $s_{B_i,k} = s_{B_j,k}$, then $s = (s_{B_1}, \ldots, s_{B_p})$ is a PNE of G.

As shown in Example 5, splitting a Boolean game makes the computation of Nash equilibria easier. If we try to compute Nash equilibria in the original game, we have to check if either $s \models \varphi_i$ or $s_{-i} \models \neg\varphi_i$ for each of the 8 strategy profiles s and for each of the three players. So, we have to make 12 verifications for each player (8 for each strategy profile in order to verify $s \models \varphi_i$, and 4 for each s_{-i} to verify $s_{-i} \models \neg\varphi_i$), then 36 for the game in the worst case. Meanwhile, the computation of PNEs once the game is split is much easier: for G_B, from Proposition 6, we have to make 6 verifications for player 1 (4 to compute $(s_1, s_3) \models \varphi_1$, and 2 to compute $s_3 \models \neg\varphi_1$); and only 2 for player 3 (because $R^*(3) \setminus \{3\} = \varnothing$). So, we only have to do 8 verifications in the worst case to find the PNEs of G_B, and the same for G_C, which has an equivalent configuration. As we have to check if the instantiation of player 3's variables are the same for PNEs of the 2 games, we have to make 17 verifications to compute PNEs of the game G.

5. Generalization to non-dichotomous preferences

The choice of dichotomous utilities (where agents can only express plain satisfaction or plain dissatisfaction, with no intermediate levels) is an important loss of generality. Fortunately, this restriction can easily be relaxed: generalizing the definition of a Boolean game so as to allow non-dichotomous preferences is easy, as it suffices to replace the preference component of a Boolean game by an input expressed in a (propositional) language for *compact preference representation* (see [5,4]). In the following, for the sake of the exposition, we focus on compact representation languages for *ordinal* preferences.

A *preference relation* $\succeq$ is a reflexive and transitive binary relation (not necessarily complete) on S. The strict preference $\succ$ associated with $\succeq$ is defined as usual by $s \succ s'$ if and only if $s \succeq s'$ and not $s' \succeq s$, and the indifference relation associated with $\succeq$ by $s \sim s'$ if and only if $s \succeq s'$ and $s' \succeq s$.

Let L be a propositional language for compact representation for ordinal preferences, equipped with a function $Induce_L$ that maps any input of L to a preference relation $\succeq$ on 2^V. If $\Phi \in L$, then Φ is called a *preference specification* and $Induce_L(\Phi)$, generally denoted $\succeq_\Phi$, is the preference relation induced by Φ. If two preference specifications Φ and Ψ of L induce the same preference relation, i.e., $Induce_L(\Phi) = Induce_L(\Psi)$, then Φ and Ψ are said to be *L-equivalent*. The set of variables $Var(\Phi)$ on which a preference specification Φ depends is a straightfoward generalization of the set of variables on which a propositional formula φ depends. We denote by $Var(\Phi)$ the set of propositional variables appearing in Φ.

Definition 10. An *L-Boolean game* is defined to be a 4-uple $G = (N, V, \pi, \Phi)$, where $N = \{1, \ldots, n\}$, V and π are as before and $\Phi = \langle \Phi_1, \ldots, \Phi_n \rangle$, where for each i, Φ_i is a compact representation in L of the preference relation $\succeq_i$ of agent i on S. $Pref_G = \langle \succeq_1, \ldots, \succeq_n \rangle$ denotes the collection of preference specifications of all players.

Φ_i is the preference specification of i. The preference relation of a player i in G is thus $Induce(\Phi_i)$, and will often be denoted $\succeq_i$.

Remark that if L_P is the purely propositional preference representation language, where a (dichotomous) preference is represented by a propositional formula, then L_P-Boolean games are just standard Boolean games as defined in Section 2. See [5,4] for several families of L-Boolean games.

For the sake of illustration we give an example in which preferences are represented with prioritized goals (see [5]); however, we insist that similar results would hold for other languages for compact preference representation, including CP-nets and other graphical languages.

Definition 11. A *prioritized goal base* Σ is a collection $\langle \Sigma^1; \ldots; \Sigma^p \rangle$ of sets of propositional formulas. Σ^j represents the set of goals of priority j, with the convention that the smaller j, the higher priority the formulas in Σ^j.

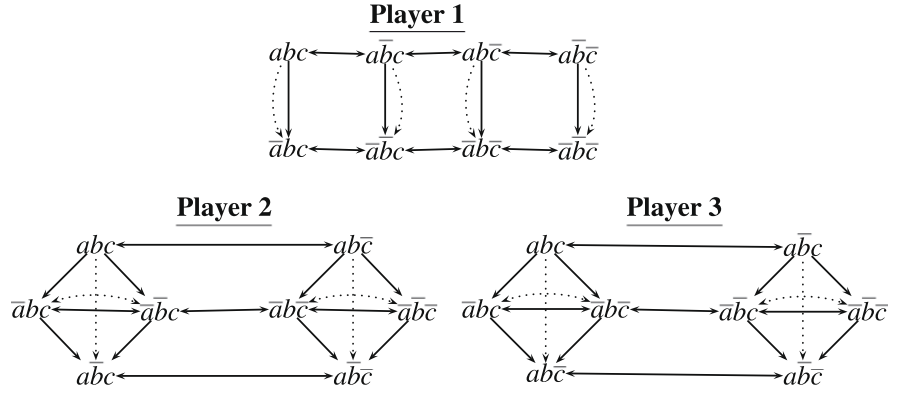
In this context, several criteria can be used in order to generate a preference relation $\succeq$ from Σ. We choose here to stick to the leximin criterion (see [16–18]). In the following, if s is an interpretation of 2^V then we let $Sat(s, \Sigma^j) = \{\varphi \in \Sigma^j \mid s \models \varphi\}$.

Definition 12. Let $\Sigma = \langle \Sigma^1; \ldots; \Sigma^p \rangle$, and let s and s' be two interpretations of 2^V. The leximin preference relation induced by Σ is defined by: $s \succ^{lex}_\Sigma s'$ if and only if $\exists k \in \{1, \ldots, p\}$ such that: $|Sat(s, \Sigma^k)| > |Sat(s', \Sigma^k)|$ and $\forall j < k$, $|Sat(s, \Sigma^j)| = |Sat(s', \Sigma^j)|$. Finally, $s \succeq^{lex}_\Sigma s'$ if and only if not $(s \prec^{lex}_\Sigma s')$.

Note that $\succeq^{lex}_\Sigma$ is a complete preference relation. Here is now an example within this preference representation language:

Example 6. $G = (N, V, \pi, \Phi)$ where $N = \{1, 2, 3\}$, $V = \{a, b, c\}$, $\pi_1 = \{a\}$, $\pi_2 = \{b\}$, $\pi_3 = \{c\}$, $\Sigma_1 = \langle \{a\} \rangle$, $\Sigma_2 = \langle \{b \vee \neg a\}; \{a\} \rangle$ and $\Sigma_3 = \langle \{c \vee \neg a\}; \{a\} \rangle$.

We draw below the preference relations[7] $Pref^{lex}_G = \langle \succeq^{lex}_{\Sigma_1}, \succeq^{lex}_{\Sigma_2}, \succeq^{lex}_{\Sigma_3} \rangle$.



[7] Arrows are oriented from more preferred to less preferred strategy profiles (s_1 is preferred to s_2 is denoted by $s_1 \to s_2$). To make the figures clearer, we do not draw edges that are obtained from others by transitivity. The dotted arrows indicate the links taken into account in order to compute Nash equilibria. For example, player 2 prefers abc to $\overline{a}bc$ because $|Sat(ab, \Sigma^1_2)| = 1$, $|Sat(ab, \Sigma^2_2)| = 1$ (both strata of Σ^2 are satisfied), and $|Sat(\overline{a}b, \Sigma^1_2)| = 1$, $|Sat(ab, \Sigma^2_2)| = 0$ (only the first stratum of Σ^2 is satisfied).

We now have to generalize the dependency graph between players from Boolean games to L-Boolean games, for an arbitrary language L. We choose here to stick with complete preorders for the sake of simplicity (our notions and results would extend to partially ordered preference relations, but this would require quite a lot of additional notations and definitions). Recall that, in Section 3, a player i was dependent on a player j if her propositional goal φ_i was dependent of one of the variables that j controls. Therefore, what we have to start with is generalizing formula-variable dependency to a dependency notion between a preference relation (or a syntactical input in a compact representation language from which this preference relation can be induced) and a variable. We stick here to this very natural, syntactical definition of dependency between a preference specification and a propositional variable (see [15] for semantical definitions).

Definition 13. Let Φ be a preference specification of a preference relation in some language L, and $x \in V$. Φ is *independent* from x if and only if there exists a preference specification Ψ in L such that

(1) Φ and Ψ are L-equivalent;
(2) $x \notin Var(\Psi)$.

A preference specification Φ is *irredundant* if and only if for all $x \in Var(\Phi)$, Φ depends on x.[8] Ψ is a *normalization of* Φ if and only if Φ and Ψ are L-equivalent and Ψ is irredundant.

Note that this definition depends on the language L chosen for representing preferences.

For instance, consider the prioritized goal base $\Sigma = \langle \Sigma_1 \rangle$, where $\Sigma_1 = \{p \wedge q, p \wedge \neg q\}$. $\Sigma' = \langle \{p\} \rangle$ induces exactly the same preference relation, thus Σ and Σ' are L-equivalent. Since $q \notin Var(\Sigma')$, Σ' does not depend on q, so Σ is independent from q. Moreover, it is clearly not possible to find a Σ'' L-equivalent to Σ' in which p does not occur, therefore Σ' is irredundant, and Σ' is a normalization of Σ.

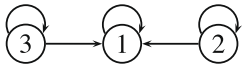
We are now in position of defining the notions used for building the dependency graph for a L-Boolean game:

Definition 14. Let $G = (N, V, \pi, \Phi)$ a L-Boolean game. The set of *relevant variables* for a player i, denoted by RV_i, is the set of all variables $v \in V$ such that Φ_i is not independent from v. The set of *relevant players* for a player i, denoted by RP_i, is the set of agents $j \in N$ such that j controls at least one relevant variable of i: $RP_i = \bigcup_{v \in RV_i} \pi^{-1}(v)$.

The dependency graph of a L-Boolean game is defined exactly as in Section 3. As we consider only complete preference relations, the definition of pure Nash equilibria is also the same as previously.

These definitions work for all languages. However, for the sake of illustration, in the following we stick to the preference representation language based on prioritized goals, from which the preference relation is induced by the *leximin* criterion.

Example 6 (*continued*). The dependency graph $\mathscr{P}$ of G is the following: $RV_1 = \{a\}$, $RV_2 = \{a, b\}$, $RV_3 = \{a, c\}$, $RP_1 = \{1\}$, $RP_2 = \{1, 2\}$, $RP_3 = \{1, 3\}$.



This game has one PNE: $\{abc\}$.

Definition 8 applies here, and allows us to introduce the notion of projection of a L-Boolean game G on a stable set B, defined exactly as in Section 3:

Definition 15. Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, and let $B \subseteq N$ be a stable set for R. The *projection* of G on B is defined by $G_B = (B, V_B, \pi_B, \Phi_B)$, where $V_B = \cup_{i \in B} \pi_i$, $\pi_B : B \to V_B$ such that $\pi_B(i) = \{v | v \in \pi_i\}$, and $\Phi_B = \{\Psi_i \mid i \in B\}$, where for every $i \in B$, Ψ_i is a normalization of Φ_i.

We can now generalize some properties previously established for non-dichotomous preferences. We start with the following, which is a generalization of Proposition 2.

Proposition 10. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, and B a stable set for G. Then G_B is a L-Boolean game.*

The first part of Proposition 6 can also be generalized in this framework.

Proposition 11. *Let G be a L-Boolean game such that the players' dependency graph $\mathscr{P}$ of G is acyclic. Then, G has at least one PNE.*

We now give a generalization of Proposition 7.

Proposition 12. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, let $B \subseteq N$ be a stable set for R, and let $G_B = (B, V_B, \pi_B, \Phi_B)$ be the projection of G on B.*

If s is a PNE for G, then s_B is a PNE for G_B.

[8] Saying Φ depends on x is the same than saying Φ is not independent from x.

The following is a generalization of Proposition 8.

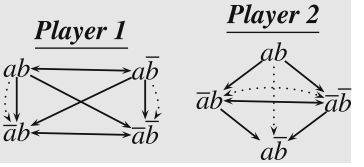
Proposition 13. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game. Let B and C be two stable sets of players, and let G_B and G_C be the two associated L-Boolean games. Suppose than s_B is a PNE for G_B and s_C is a PNE for G_C such that $\forall i \in B \cap C$, $s_{B,i} = s_{C,i}$. Then, $s_{B\cup C}$ is a PNE for $G_{B\cup C}$.*

We can then generalize Proposition 9 exactly in the same way than in Section 4.

Proposition 14. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, and let $B_1 \cdots B_p$ be p stable sets of players, such that $B_1 \cup \cdots \cup B_p = N$. Let $G_{B_1}, \ldots, G_{B_p}$ be the p associated L-Boolean games. If there exist $s_{B_1} \ldots s_{B_p}$ PNEs of $G_{B_1}, \ldots, G_{B_p}$ such that for all $i, j \in \{1, \ldots, p\}$ and $k \in B_i \cap B_j$, $s_{B_i,k} = s_{B_j,k}$, then $s = (s_{B_1}, \ldots, s_{B_p})$ is a PNE of G.*

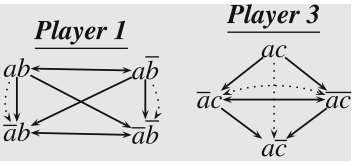
Example 6 (*continued*). The sets of players $B = \{1, 2\}$ and $C = \{1, 3\}$ are stable. We have two new Boolean games:

$G_B = (B, V_B, \pi_B, \Phi_B)$, with $B = \{1, 2\}$, $V_B = \{a, b\}$, $\pi_1 = a$, $\pi_2 = b$, $\Sigma_1 = \langle a \rangle$, and $\Sigma_2 = \langle (b \vee \neg a); a \rangle$. The preference relations $Pref_G^{lex} = \langle \succeq_1^{lex}, \succeq_2^{lex} \rangle$ are drawn on the right.



G_B has one PNE: $\{ab\}$ (denoted by $s_B = (s_{B,1}, s_{B,2})$).

$G_C = (C, V_C, \pi_C, \Phi_C)$, with $C = \{1, 3\}$, $V_C = \{a, c\}$, $\pi_1 = a$, $\pi_3 = c$, $\Sigma_1 = \langle a \rangle$ and $\Sigma_3 = \langle (c \vee \neg a); a \rangle$. The preference relations $Pref_G^{lex} = \langle \succeq_1^{lex}, \succeq_3^{lex} \rangle$ are drawn on the right.



G_C has one PNE: $\{ac\}$ (denoted by $s_C = (s_{C,1}, s_{C,3})$).
$B \cap C = \{1\}$. But we have $s_{B,1} = s_{C,1} = a$, so Proposition 14 can be applied: $G_{B\cup C}$ has one PNE: $\{abc\}$.

6. Discussion

We have shown how the intuitive notion of dependency between players in a Boolean game can be exploited so as to facilitate the computation of pure-strategy Nash equilibria. Moreover, our properties not only hold for the standard version of Boolean game (with propositional goals and dichotomous preferences) but also for generalized Boolean games, where players' preferences are expressed in a compact representation language (prioritized goals bases, CP-nets, etc. cf. [5,4]).

Another class of games with dichotomous preferences shares a lot with Boolean games: Qualitative Coalitional Games (QCG), introduced by [19]. In a QCG, each agent has a set of goals, and is satisfied if one of her goals is achieved, but is indifferent on which goal is, and on the number of goals achieved.[9] Thus agents have dichotomous preferences (as in the standard version of Boolean games – cf. Sections 2–4). A characteristic function associates with each agent, or set of agents, the set of goals they can achieve. See [4] for more details.

Boolean games take place in a larger stream of work, that we may gather under the generic name of *compactly represented games*. All frameworks for compactly represented games make use of notions of dependencies between players and/or actions that have a lot in common with ours. Most of these frameworks, including [9,10,21], share the following mode of representation of players' utilities: the utility of a player i is described by a table specifying a numerical value for each combination of values to each of the set of variables that are relevant to i.[10] The representation of games with such utility tables is called *graphical normal form* (GNF) in [11]. Dependencies between players and variables in such games naturally induces a dependency relation between players, in the same way as we do (i depends on j if i's utility table refers to a variable that is controlled by j).

Boolean games are very similar to these graphical games, except that the form chosen for expressing compactly players' preferences is *logical*. The logical form is sometimes exponentially more compact than the graphical form: consider for instance the dichotomous preference relation corresponding to the goal $\varphi = x_1 \oplus \cdots \oplus x_p$, where $\oplus$ is exclusive or. While the logical representation of u_φ is linear in p, its representation by utility tables would be exponential in p, since each of the p variables is relevant to player's utility. In the general case of non-dichotomous utility functions or preference relations,

[9] In [20], QCGs are extended by allowing agents to have preferences over goals.
[10] In multi-agent influence diagrams [9], a player's utility is actually expressed in a more compact way as the sum of local utilities, each corresponding to a smaller set of variables.

the Boolean game framework, by allowing some flexibility on the choice of the language for preference representation, is more general than the one of graphical games, where the format for expressing preferences is fixed. Moreover, solving games in logical form may benefit from the huge literature on SAT and related algorithms for propositional logic (see Section 6.3 of [4] for more details).

The notion of dependency between players and variables in graphical games is used for the very same purpose as our dependency graph, namely, to split up a game into a set of interacting smaller games, which can be solved more or less independently. In [11], specific restrictions on graphical games are studied, either by bounding the size of players' neighbourhoods (the neighbourhood of a player i in a graphical game is the set of players who potentially influence the utility of i), or by imposing that the dependency relation between players should be acyclic. Then [11] studies the impact of such restrictions on the complexity of checking the existence of a Nash equilibrium (or their computation). Clearly, similar structural restrictions on Boolean games would probably allow for a complexity fall with respect to the complexity results for the general case in [4]. This is left for further study.

The work reported here is still preliminary and can be pursued in many other directions.

First, apart of the *structural* restrictions mentioned just above, we may study the impact of *syntactical* restrictions on propositional goals on the computation of Nash equilibria and on the construction of the dependency graph. In [22], Sichman and Conte introduced dependency graphs which can represent and/or dependencies[11] on actions needed to achieve an agent's goal and on the agents who control these actions. In the first case, this is similar to our set of relevant variables, and in the second case this corresponds to our set of relevant players. Sichman and Conte's ideas can be used for introducing and/or dependencies in our framework, but using the syntactical form of the goals. In [23], three notions of dependency are defined: the weak one is the same than our (an agent i is dependent from a set of agents C if C can achieve i's goal). The second one, called normal dependency, adds to weak dependency the condition that i cannot achieve her goal by herself. Finally, the third one adds the fact that agents in C are the only ones able to achieve i's goal. Following Sichman and Conte [22], Boella et al. [23] use an and-graph to represent weak/strong dependency: for every coalition C, there is an and-edge from agent i, $i \in C$, to agent $j \in N$ if the agents in C can achieve the goal desired by the agent j. This notion of dependency is the basis of their computation of admissible coalition under the do-ut-des criterion (see [24]).

Second, while our Section 5 does not focus on particular language (prioritized goals are used in an example just for the sake of illustration), we may want to study in further detail the computation of Nash equilibria (using the structural properties of the game) for some specific languages for preference representation (see [5,4] for the case of CP-nets and prioritized goals). A particularly appealing language consists in specifying preferences by *weighted goals*, where a player's utility function is represented using several propositional formulas, each being attached with a numerical value (see [25]). This is especially interesting because this language generalizes the representation by utility tables in graphical games.

So far, Boolean games allow only for expressing *static* games (with simultaneous moves by the players) and with *complete information*. Enriching Boolean games with dynamicity and nature-driven uncertainty, as in multi-agent influence diagrams, is not as simple as it looks at first glance, and is a challenging issue.

In this paper we focused only on pure-strategy Nash equilibria (as does more generally the literature on succinctly represented games), which is a strong limitation. Computing *mixed strategy* Nash equilibria in Boolean games is a challenging issue. Finding a mixed equilibrium in a succinctly represented game would probably require to solve a linear program with an exponential number of variables. However, our decomposition techniques could work as well, to a certain extent. This is a very interesting (and difficult) topic for further research.

Finally, let us mention an informal relationship to social network analysis.[12] A social network is a graph whose nodes are individuals and edges represent some type of interdependency. Our dependency graph between players (and more generally, similar dependency graph in graphical games such as in [10,9,11]), can therefore be viewed as a specific case of social network, where the dependency relation between i and j expresses that the satisfaction of i may rely on the action taken by j. Technically, of course, both lines of work significantly diverge: social network analysis studies the properties of large graphs found in the real world (friendship or trading relations, co-authorship etc.). Our paper does not study the properties of such graphs but uses their structure to make the computation of Nash equilibria easier. Some of the properties we use are relevant in social network analysis. For instance, the decomposition techniques at work in Propositions 7 and 8 will be more efficient if the dependency graph is composed of weakly interconnected (but possibly strongly intra-connected) clusters of agents, which may correspond (to some extent) to the notion of small world well-known in social networks.

Acknowledgement

We would like to thank the anonymous referees, as well as those of the ECSQARU'07 preliminary version of this article, for many useful comments and suggestions. This work has been partly supported by the ANR project ANR-05-BLAN-0304 "Preference Handling and Aggregation in Combinatorial Domains".

[11] The or-dependency means that several actions allow an agent to achieve a goal in several ways, and the and-dependency means that this agent needs all these actions to achieve her goal.

[12] We are indebted to one of the reviewers for this idea.

A. Proofs

Proposition 1. *Let $\mathscr{C} \subset 2^N$. There exists a Boolean game G such that $\mathscr{C}$ is the set of stable sets for G if and only if $\mathscr{C}$ satisfies the following four properties:*

(1) $\varnothing \in \mathscr{C}$;
(2) $N \in \mathscr{C}$;
(3) If $B, B' \in \mathscr{C}$ then $B \cup B' \in \mathscr{C}$;
(4) If $B, B' \in \mathscr{C}$ then $B \cap B' \in \mathscr{C}$.

Proof. Since every dependency graph corresponds to some Boolean game, it is sufficient to show that there exists a relation R on N such that $\mathscr{C}$ is the set of stable sets for R if and only if $\mathscr{C}$ satisfies (1)–(4).

$\Rightarrow$ $\varnothing$ and N are obviously stable for R.
If B and B' are stable sets for R, then $R(B) \subseteq B$ and $R(B') \subseteq B'$. So, $R(B) \cup R(B') \subseteq B \cup B'$. Now, $R(B) = \{j | \exists i \in B : j \in RP_i\}$ and $R(B') = \{j | \exists i \in B' : j \in RP_i\}$. Thus, $R(B) \cup R(B') = \{j | \exists i \in B \cup B' : j \in RP_i\} = R(B \cup B')$. Then, $R(B \cup B') \subseteq B \cup B'$, and $B \cup B'$ is a stable set.
If B and B' are stable, then $R(B) \subseteq B$ and $R(B') \subseteq B'$. So, $R(B) \cap R(B') \subseteq B \cap B'$. We know than, $R(B) = \{j | \exists i \in B : j \in RP_i\}$ and $R(B') = \{j | \exists i \in B' : j \in RP_i\}$. Thus, $R(B) \cap R(B') = \{j | \exists i \in B \cap B' : j \in RP_i\} = R(B \cap B')$. Then, $R(B \cap B') \subseteq B \cap B'$, and $B \cap B'$ is a stable set.

$\Leftarrow$ Let $\mathscr{C}$ be a set of coalitions satisfying (1)–(4).
For every $i \in N$, there exists a smallest set X_i in $\mathscr{C}$ containing i: $X_i = \bigcap\{B \in \mathscr{C} | i \in B\}$ (since $\mathscr{C}$ is closed for $\cap$, we have $X_i \in \mathscr{C}$). Now, we construct R such that for all $i, j \in N$, $(i, j) \in R$ if and only if $j \in X_i$.
For every $B \in \mathscr{C}$ and every $i \in B$, if $(i, j) \in R$, then by construction of R, $j \in B$, therefore B is stable for R.
It remains to be shown that every subset B stable for R is in $\mathscr{C}$. Assume that $R(B) \subseteq B$. Then for every $i \in B$, $R(i) \subseteq B$. Now, by construction of R, $R(B) = X_i$. Since $i \in X_i$, we have $B \subseteq \cup_{i \in B} X_i$. Now, for every $i \in B$, $R(i) \subseteq B$ by hypothesis, therefore $\cup_{i \in B} X_i = B$. Now, every X_i is in $\mathscr{C}$, therefore, using (3), we have $B \in \mathscr{C}$. □

Proposition 3. *If B is a stable set, then $G_B = (B, V_B, \pi_B, \Phi_B)$ is a Boolean game.*

Proof. Let $G_B = (B, V_B, \pi_B, \Phi_B)$. We have to check that every goal φ_i for $i \in B$ is a formula of L_{V_B}, or can be rewritten equivalently as a formula of L_{V_B}.

Suppose than $\exists i \in B$, $\exists v \in Var(\varphi_i)$ such that $v \notin V_B$. So, $\forall j \in B$, $v \notin \pi_j$. Let $k \in N \setminus B$ such that $v \in \pi_k$. We know that $v \in Var(\varphi_i)$, so either φ_i is independent from v, and then is logically equivalent to a formula in which v does not appear; or φ_i is not independent from v, and in this case $v \in RV_i$ and by definition $k \in RP_i$. So, $k \in R(i)$, but $k \notin B$: this is in contradiction with the fact that B is stable. □

Proposition 4. *Let G be a Boolean game. If $\forall i \in N$, $i \notin RP_i$ then every $s \in S$ is a PNE.*

Proof. Since $\forall i \in N$, $i \notin RP_i$, i has no influence on her own goal: $\forall i \in N$, either $s_{-i} \models \varphi_i$ or $s_{-i} \models \neg\varphi_i$. So, we have $\forall s_i \in S_i$, $(s_i, s_{-i}) \models \varphi_i$ or $s_{-i} \models \neg\varphi_i$. s is thus always a PNE. □

Proposition 5. *Let G be a Boolean game such that $\forall i \in N$, $|RP_i| = 1$.*
s is a PNE if and only if $\forall i \in N$ such that $RP_i = \{i\}$, $s \models \varphi_i$.

Proof

$\Rightarrow$ Let s be a PNE and let i such that $RP_i = \{i\}$. Suppose that $s \models \neg\varphi_i$ (1). $RP_i = \{i\}$ implies $DepVar(\varphi_i) \subseteq \pi_i$, therefore (1) implies $s_i \models \neg\varphi_i$ (2). Now, s is a PNE, therefore by Proposition 3, either (a) $s \models \varphi_i$ or (b) $s_{-i} \models \neg\varphi_i$. (a) is excluded because (2) holds, therefore we have $s_{-i} \models \neg\varphi_i$ (3). Now, because $\neg\varphi_i$ does not depend on the variables controlled by players other than i, $\neg\varphi_i$ must be a tautology, that is, $\varphi_i \equiv \bot$, which is excluded by definition (all goals are satisfiable). Thus we have a contradiction.

$\Leftarrow$ Suppose now that (1) $s \models \varphi_i$ holds for every i such that $RP_i = \{i\}$. Let $i \in N$. By Proposition 3, s is a PNE if and only if either (a) $s \models \varphi_i$ or (b) $s_{-i} \models \neg\varphi_i$. If $RP_i = \{i\}$ then $s \models \varphi_i$, and (a) is satisfied. If $RP_i = \{j\}$ with $j \neq i$, then $s \models \neg\varphi_i$ if and only if $s_{-i} \models \neg\varphi_i$, therefore either (a) or (b) is satisfied. We conclude that s is a PNE. □

Proposition 6. *Let G be a Boolean game such that the irreflexive part of the dependency graph $\mathscr{D}$ of G is acyclic.*
Then, G has at least one PNE. Moreover, s is a PNE of G if and only if for every $i \in N$, either $(s_i, s_{R^*(i)\setminus\{i\}}) \models \varphi_i$ or $s_{R^*(i)\setminus\{i\}} \models \neg\varphi_i$.

Proof

- Assume without loss of generality that the players are numbered in such a way that for every $i \in N$, if i depends on j then $j < i$. Let $s = (s_1, \ldots, s_n)$ be defined inductively as follows: for $i = 1, \ldots, n$, if $(\varphi_i)_{s_1,\ldots,s_{i-1}}$ is satisfiable then take s_i such that $(s_1, \ldots, s_i) \models \varphi_i$. Such an s_i exists because i does not depend on k for all $k > i$. If $(\varphi_i)_{s_1,\ldots,s_{i-1}}$ is unsatisfiable then take any s_i. It is easily checked that s is a PNE.
- s is a PNE of G if and only if for every $i \in N$, either $(s_i, s_{R^*(i)\setminus\{i\}}) \models \varphi_i$ or $s_{R^*(i)\setminus\{i\}} \models \neg\varphi_i$.
  - $\Rightarrow$ Assume that s is a PNE and that exists a player $i \in N$ such that $(s_i, s_{R^*(i)\setminus\{i\}}) \not\models \varphi_i$ and $s_{R^*(i)\setminus\{i\}} \not\models \neg\varphi_i$.
    Then, we have $s \models \neg\varphi_i$, and, as $\forall k \notin R^*(i)$, $(\varphi_i)_{s_k} = \varphi_i$, $s_{-i} \not\models \neg\varphi_i$. Thus, s is not a PNE.
  - $\Leftarrow$ Assume now that $(s_i, s_{R^*(i)\setminus\{i\}}) \models \varphi_i$ or $s_{R^*(i)\setminus\{i\}} \models \neg\varphi_i$.
    If i is a player such that $(s_i, s_{R^*(i)\setminus\{i\}}) \models \varphi_i$, we obviously have $s \models \varphi_i$.If $s_{R^*(i)\setminus\{i\}} \models \neg\varphi_i$, then, as $\forall k \notin RP_i, (\varphi_i)_{s_k} = \varphi_i$, we have $s_{-i} \models \neg\varphi_i$. As $\forall i \in N$, either $s \models \varphi_i$, or $s_{-i} \models \neg\varphi_i$, s is a PNE. □

Proposition 7. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game, let $B \subseteq N$ be a stable set for R, and let G_B be the projection of G on B. If s is a PNE for G, then s_B is a PNE for G_B.*

Proof. Let $s = (s_B, s_{-B})$ be a PNE of G, which is equivalent to: for every $i \in N$, either $s \models \varphi_i$ or $s_{-i} \models \neg\varphi_i$ [4]. Let us show that s_B is a PNE of G_B. Let $i \in B$.

If $s \models \varphi_i$, then because $i \in B$ and B is stable, s_{-B} has no influence on φ_i and we have $s_B \models \varphi_i$.

If $s_{-i} \models \neg\varphi_i$ then only players in B have an influence on φ_i. So, we have $s_{(B\setminus\{i\})} \models \neg\varphi_i$, which corresponds to $s_{B-i} \models \neg\varphi_i$. □

Proposition 8. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game. Let B and C be two stable sets of players, and let G_B and G_C be the two associated Boolean games.*

Suppose than s_B is a PNE for G_B and s_C is a PNE for G_C such that $\forall i \in B \cap C$, $s_{B,i} = s_{C,i}$, where $s_{B,i}$ (resp. $s_{C,i}$) represents the strategy of player i for the game G_B (resp. G_C).

Then, $s_{B\cup C}$ is a PNE for $G_{B\cup C}$.

Proof. B and C are stable, so from Proposition 1, $B \cup C$ is a stable set, and from Proposition 2, $G_{B\cup C}$ is a Boolean game. We know that $s_B = (s_{B\setminus B\cap C}, s_{B\cap C}) = (s_{B\setminus C}, s_{B\cap C})$ and that $s_C = (s_{C\setminus B}, s_{B\cap C})$. As $\forall i \in B \cap C$, $s_{B,i} = s_{C,i}$, we have $s_{B\cap C} = (s_B, s_C) = (s_{B\setminus C}, s_{C\setminus B}, s_{B\cap C})$. Let $i \in B \cup C$.

- $i \in B \setminus C$ (or $i \in C \setminus B$). As B is stable, we know that $\forall j \in R(i)$, $j \notin C \setminus B$. Thus, $s_{C\setminus B}$ has no influence on the satisfaction of φ_i, and so this satisfaction is only determined by $s_B = (s_{B\setminus C}, s_{B\cap C})$. In this case, if $s_B \models \varphi_i$, then $s_{B\cup C} \models \varphi_i$, and if $s_{B-i} \models \neg\varphi_i$, then $s_{B\cup C-i} \models \neg\varphi_i$.
- $i \in B \cap C$. As B and C are stable, we know that $B \cap C$ is stable, and then that $\forall j \in R(i), j \in R(B \cap C) \subseteq B \cap C$. Thus, the satisfaction of φ_i is only determined by $s_{B\cap C}$. As s_B is a PNE for G_B and s_C is a PNE for G_C, we have either $s_{B\cap C} \models \varphi_i$, or $s_{B\cap C-i} \models \neg\varphi_i$. Then, we have either $s_{B\cup C} \models \varphi_i$, or $s_{B\cup C-i} \models \neg\varphi_i$.

So, we have $\forall i \in N$ either $s_{B\cup C} \models \varphi_i$, or $s_{B\cup C-i} \models \neg\varphi_i$. $s_{B\cup C}$ is PNE of G. □

Proposition 9. *Let $G = (N, V, \pi, \Phi)$ be a Boolean game, and let $B_1 \cdots B_p$ be p stable sets of players, such that $B_1 \cup \cdots \cup B_p = N$. Let $G_{B_1}, \ldots, G_{B_p}$ be the p Boolean games associated.*

If $\exists s_{B_1} \cdots s_{B_p}$ PNEs of $G_{B_1}, \ldots, G_{B_p}$ such that $\forall i, j \in \{1, \ldots, p\}$, $\forall k \in B_i \cap B_j$, $s_{B_i,k} = s_{B_j,k}$, then $s = (s_{B_1}, \ldots, s_{B_p})$ is a PNE of G.

Proof. Let B_1 and B_2 two stable sets, and G_{B_1} and G_{B_2} the two Boolean games associated. We can apply Proposition 8, and show that $B_1 \cup B_2$ is a stable set, that $G_{B_1\cup B_2}$ is a Boolean game and that $s_{B_1\cup B_2}$ is a PNE of $G_{B_1\cup B_2}$.

We can do the same for $B_1 \cup B_2$ and B_3, and so on until the final result. □

Proposition 10. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, and B a stable set for G. Then G_B is a L-Boolean game.*

Proof. Let $G_B = (B, V_B, \pi_B, \Phi_B)$. B is stable, therefore for every $i \in B$, Φ_i depends only on variables controlled by players of B, that is, depends only on V_B. Therefore, if $i \in B$ and Ψ_i is a normalization of Φ_i, then $Var(\Psi_i) \subseteq V_B$. This entails that $G_B = (B, V_B, \pi_B, \Phi_B)$ is a well-defined L-Boolean game. □

Proposition 11. *Let G be a L-Boolean game such that the players' dependency graph $\mathscr{P}$ of G is acyclic. Then, G has at least one PNE.*

Proof. Suppose without loss of generality that players are numbered on the following way: for all $i \in N$, if i depends on j, then $j < i$. Let $s = (s_1, \ldots, s_n)$ defined inductively as follows: for $i = 1, \ldots, n$, we take s_i such that for all s'_i, $(s_1, \ldots, s_i) \succeq_i (s_1, \ldots, s'_i)$. Such a s_i exists because i does not depend on k for all $k > i$. We have built a strategy profile s such that for all i, for all s'_i, $(s_i, s_{-i}) \succeq_i (s'_i, s_{-i})$. s is a PNE. □

Proposition 12. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, let $B \subseteq N$ be a stable set for R, and let $G_B = (B, V_B, \pi_B, \Phi_B)$ be the projection of G on B. If s is a PNE for G, then s_B is a PNE for G_B.*

Proof. Let s be a PNE for G: $\forall i \in N$, $\forall s'_i \in S_i$, $(s'_i, s_{-i}) \preceq_i (s_i, s_{-i})$. Let $s = (s_B, s_{-B})$. We want to check that s_B is a PNE for G_B.

Let $i \in B$. Suppose that s_B is not a PNE for G_B, then there is a $s'_i \in S_i$ such that $(s'_i, s_{B-i}) \succ_i (s_i, s_{B-i})$. As $i \in B$ and B is stable, we know that the only players having an influence on i are in B. So, s_{-B} has no influence on i's preferences. Thus we have $(s'_i, s_{B-i}, s_{-B}) \succ_i s = (s_i, s_{B-i}, s_{-B})$, that contradicts the fact than s is a PNE for G. □

Proposition 13. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game. Let B and C be two stable sets of players, and let G_B and G_C be the two associated L-Boolean games. Suppose than s_B is a PNE for G_B and s_C is a PNE for G_C such that $\forall i \in B \cap C$, $s_{B,i} = s_{C,i}$. Then, $s_{B\cup C}$ is a PNE for $G_{B\cup C}$.*

Proof. B and C are stable, so from Proposition 1, $B \cup C$ is a stable set and from Proposition 10, $G_{B\cup C}$ is a Boolean game. Let $i \in B \cup C$.

- $i \in B \setminus C$ (or $i \in C \setminus B$). s_B is PNE for G_B, so $\forall s'_i \in S_i, (s'_i, s_{B-i}) \preceq_i (s_i, s_{B-i})$. As $i \notin C$, we can write $\forall s'_i \in S_i$, $(s'_i, s_{B\cup C-i}) \preceq_i (s_i, s_{B\cup C-i})$.
- $i \in B \cap C$. We have $\forall k \in B \cap C$, $s_{B,k} = s_{C,k}$. s_B is PNE for G_B, so $\forall s'_i \in S_i, (s'_i, s_{B-i}) \preceq_i (s_i, s_{B-i})$; s_C is a PNE for G_C, so $\forall s'_i \in S_i, (s'_i, s_{C-i}) \preceq_i (s_i, s_{C-i})$. As $\forall k \in B \cap C$, $s_{B,k} = s_{C,k}$, we have $\forall s'_i \in S_i, (s'_i, s_{B-i}, s_{C-i}) \preceq_i (s_i, s_{B-i}, s_{C-i})$, that is $\forall s'_i \in S_i, (s'_i, s_{B\cup C-i}) \preceq_i (s_i, s_{B\cup C-i})$. □

Proposition 14. *Let $G = (N, V, \pi, \Phi)$ be a L-Boolean game, and let $B_1 \dots B_p$ be p stable sets of players, such that $B_1 \cup \dots \cup B_p = N$. Let $G_{B_1}, \dots, G_{B_p}$ be the p associated L-Boolean games. If there exist $s_{B_1} \dots s_{B_p}$ PNEs of $G_{B_1}, \dots, G_{B_p}$ such that for all $i, j \in \{1, \dots, p\}$ and $k \in B_i \cap B_j$, $s_{B_i,k} = s_{B_j,k}$, then $s = (s_{B_1}, \dots, s_{B_p})$ is a PNE of G.*

Proof. Similar to the proof of Proposition 9. □

References

[1] P. Harrenstein, W. van der Hoek, J.-J. Meyer, C. Witteveen, Boolean games, in: Proceedings of the 8th International Conference on Theoretical Aspects of Rationality and Knowledge (TARK'01), in: J. van Benthem (Ed.), Theoretical Aspects of Rationality and Knowledge, Morgan Kaufmann, San Francisco, 2001, pp. 287–298.
[2] P. Harrenstein, Logic in Conflict, Ph.D. Thesis, Utrecht University, 2004.
[3] P.E. Dunne, W. van der Hoek, Representation and complexity in boolean games, in: Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04), LNCS, vol. 3229, José Júlio Alferes et João Alexandre Leite (eds), 2004, pp. 347–359.
[4] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, B. Zanuttini, Compact preference representation and Boolean games, Journal of Autonomous Agents and Multi-Agent Systems (2008).
[5] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, Compact preference representation for Boolean games, Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence (PRICAI'06), vol. 4099, Springer-Verlag, 2006, pp. 41–50.
[6] K. Leyton-Brown, M. Tennenholtz, Local-effect games, in: International Joint Conferences on Artificial Intelligence (IJCAI'03), 2003, pp. 772–777.
[7] N. Bhat, K. Leyton-Brown, Computing Nash equilibria of action-graph games, in: Conference on Uncertainty in Artificial Intelligence (UAI'04), 2004, pp. 35–42.
[8] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, Efficient coalitions in Boolean games, in: K. Apt, R. Rooij (Eds.), New Perspectives on Games and Interaction, Texts in Logic and Games, 2008.
[9] D. Koller, B. Milch, Multi-Agent Influence Diagrams for Representing and Solving Games, Games and Economic Behavior 45 (1) (2003) 181–221. full version of paper in IJCAI '01.
[10] M. Kearns, M.L. Littman, S. Singh, Graphical models for game theory, in: Proceedings of Uncertainty in Artificial Intelligence (UAI'01), 2001, pp. 253–260.
[11] G. Gottlob, G. Greco, F. Scarcello, Pure Nash equilibria: hard and easy games, Journal of Artificial Intelligence Research 24 (2005) 357–406.
[12] J. Lang, P. Liberatore, P. Marquis, Propositional independence – formula-variable independence and forgetting, Journal of Artificial Intelligence Research 18 (2003) 391–443.
[13] M.J. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, 1994.
[14] C. Boutilier, R.I. Brafman, H.H. Hoos, D. Poole, Reasoning with conditional Ceteris Paribus preference statements, in: Proceedings of Uncertainty in Artificial Intelligence (UAI'99), 1999, pp. 71–80.
[15] P. Besnard, J. Lang, P. Marquis, Variable forgetting in preference relations over propositional domains, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06), Springer-Verlag, 2006, pp. 763–764.
[16] D. Dubois, J. Lang, H. Prade, Inconsistency in possibilistic knowledge bases: to live with it or not live with it, in: L. Zadeh, J. Kacprzyk (Eds.), Fuzzy Logic for the Management of Uncertainty, Wiley, New York, 1992, pp. 335–351.
[17] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, H. Prade, Inconsistency management and prioritized syntax-based entailment, in: Bajcsy (Ed.), Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93), Chambéry, France, 1993, pp. 640–645.
[18] D.J. Lehmann, Another perspective on default reasoning, Annals of Mathematics and Artificial Intelligence 15 (1995) 61–82.
[19] M. Wooldridge, P.E. Dunne, On the computational complexity of qualitative coalitional games, Artificial Intelligence 158/1 (2004) 27–73.
[20] P.E. Dunne, M. Wooldridge, Preferences in qualitative coalitional games, in: Proceedings of the 6th Workshop on Game Theoretic and Decision Theoretic Agents (GTDT'04), New York, 2004, pp. 29–38.
[21] P. La Mura, Game networks, in: Proceedings of Uncertainty in Artificial Intelligence (UAI'00), 2000, pp. 335–342.
[22] J.S. Sichman, R. Conte, Multi-agent dependence by dependence graphs, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), 2002, pp. 483 – 490.

[23] G. Boella, L. Sauro, L. van der Torre, From social power to social importance, Web Intelligence and Agent Systems Journal 5 (5) (2007) 393–404.
[24] G. Boella, L. Sauro, L. van der Torre, Admissible agreements among goal-directed agents, in: Proceedings of the International Conference on Intelligent Agent Technology (IAT'05), IEEE, 2005, pp. 543–554.
[25] Y. Chevaleyre, U. Endriss, J. Lang, Expressive power of weighted propositional formulas for cardinal preference modelling, in: P. Doherty, J. Mylopoulos, C. Welty (Eds.), Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06), AAAI Press, 2006, pp. 145–152.