Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

∃GUARANTEENASH for Boolean Games is NEXP-Hard

Egor Ianovski and **Luke Ong**
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, UK

Abstract

Boolean games are an expressive and natural formalism through which to investigate problems of strategic interaction in multiagent systems. Although they have been widely studied, almost all previous work on Nash equilibria in Boolean games has focused on the restricted setting of pure strategies. This is a shortcoming as finite games are guaranteed to have at least one equilibrium in mixed strategies, but many simple games fail to have pure strategy equilibria at all. We address this by showing that a natural decision problem about mixed equilibria: determining whether a Boolean game has a mixed strategy equilibrium that guarantees every player a given payoff, is NEXP-hard. Accordingly, the ϵ variety of the problem is NEXP-complete. The proof can be adapted to show coNEXP-hardness of a similar question: whether all Nash equilibria of a Boolean game guarantee every player at least the given payoff.

Introduction

A multiagent environment makes strategic considerations inevitable. Any attempt to explain the behaviour of a system consisting of self-interested agents cannot ignore the fact that agents' behaviour may be influenced or completely determined by the behaviour of other agents in the system. As the field of game theory concerns itself with precisely these issues, its concepts find fertile ground in the study of multiagent systems.

A shortcoming of game theoretical techniques is that games, being combinatorial objects, are liable to get very large very quickly. Any computational application of game theory would need alternative representations to the normal and extensive forms prominent in the economics literature. One such representation, based on propositional logic, is the Boolean game.

Boolean games were initially introduced as two player games which have an algebra isomorphic to the Lindenbaum algebra for propositional logic (Harrenstein et al. 2001). Since then Boolean games have garnered interest from the multiagent community as a simple yet expressive framework to model strategic interaction. This has led to the study of complexity issues involved in reasoning about these games. While many questions have been answered, the issue of mixed strategies remained open.

In this paper we address this lacuna and present the first complexity result about mixed equilibria in the Boolean games literature: the NEXP-hardness of ∃GUARANTEENASH, which asks whether a Boolean game has an equilibrium where each player attains at least $\boldsymbol{v}[i]$ utility, for some input vector $\boldsymbol{v}$.

Related Work

Complexity results for Boolean games were first studied in the two player case by Dunne and van der Hoek (2004) and the n-player case by Bonzon et al. (2006), where among other results the authors showed that determining the existence of a pure equilibrium is Σ_2^p-complete in the general case, and can be easier should some restrictions be placed on the goal formulae of the players. A later paper considered to what extent the interdependency of players contributed to complexity (Bonzon, Lagasquie-Schiex, and Lang 2009). Further enquiry into tractable fragments of Boolean games was carried out by Dunne and Wooldridge (2012).

Since their inception Boolean games have attracted the interest of the multiagent community, which led to increased interest about their cooperative side. The standard game theoretic notion of core was studied in a Boolean setting (Dunne et al. 2008; Bonzon, Lagasquie-Schiex, and Lang 2012), and a number of papers considered means of lining up player incentives with a central planner (Endriss et al. 2011; Kraus and Wooldridge 2012; Levit et al. 2013). in two recent papers (2013a; 2013b), Ågotnes et al. considered ways of combining Boolean games with the spiritually similar games in epistemic logic (Ågotnes et al. 2011; Ågotnes and van Ditmarsch 2011).

All the work mentioned has been in the setting of pure strategies only. Mixed strategies do arise in a cardinal extension to Boolean games dubbed weighted Boolean formula games (Mavronicolas, Monien, and Wagner 2007) or satisfiability games (Bilò 2007), but there the authors only consider a very restricted fragment of these games, and its relation to a subclass of the congestion games of Rosenthal (1973).

A very similar framework to Boolean games is that of Boolean circuit games (Schoenebeck and Vadhan 2012), building on earlier work by Feigenbaum, Koller, and Shor

(1995). There players are equipped with a Boolean circuit with k input gates and m output gates. The input gates are partitioned among the players, and a player's strategy is an assignment of values to the gates under his control. The output gates encode a binary representation of the player's utility on a given input. In this setting we do find a treatment of mixed strategies, including the complexity of ∃GUARANTEENASH.

Note that a Boolean game can be seen as a very specific type of Boolean circuit game: the players' circuits are restricted to NC^1, and the number of output gates to one. Thus easiness results for Boolean circuit games directly transfer to Boolean games, and hardness results transfer in the other direction. In particular, this means that the NEXP-completeness of ∃GUARANTEENASH for Boolean circuit games proved by Schoenebeck and Vadhan does not imply the result of this paper.

Preliminaries

While there are many breeds of games in the literature, we here restrict ourselves to what is perhaps the most widely studied class:

Definition 1. A *finite strategic game* consists of n players, each equipped with a finite set of pure strategies, S_i, and a utility function $u_i : S_1 \times \cdots \times S_n \to \mathbb{R}$.

An n-tuple of strategies is called a *strategy profile*: thus a utility function maps strategy profiles to the reals.

Example 1. In a game of *matching pennies* two players are given a coin each and may choose to display that coin heads or tails up. Player Two seeks to match the move of Player One, while player one seeks to avoid that. Hence we have $u_2(HH) = u_2(TT) = 1$, $u_1(HT) = u_1(TH) = 1$, and 0 otherwise.

Note that to represent a finite strategic game explicitly (the *normal form* of the game) we would need to list the players' utility on every possible profile. This would require on the order of $n|S_i|^n$ entries, taking S_i to mean the size of the "typical" strategy set. Such a representation is both exponential in the number of players and linear in the number of strategies - which in itself may be very large, e.g. in chess.

Ideally we would wish to avoid such a representation. If a game has some internal structure, it would be natural to ask if the game can be described in a more succinct way. In the case where the game can be interpreted as players holding propositional preferences over Boolean variables the Boolean game offers precisely that.

Definition 2. A *Boolean game* is a representation of a finite strategic game given by n disjoint sets of propositional variables, Φ_i, and n formulae of propositional logic, γ_i.

The intended interpretation is that player i controls the variables in Φ_i in an attempt to satisfy γ_i, which may depend on variables not in player i's control. Player i plays a truth assignment to Φ_i, i.e. his set of pure strategies is 2^{Φ_i}. A profile of strategies, ν, is thus a model of proposition logic and i's utility function is $\nu \mapsto 1$ if $\nu \vDash \gamma_i$, and $\nu \mapsto 0$ otherwise.

Example 2. Matching pennies can be given a Boolean representation by setting $\Phi_1 = \{p\}$, $\Phi_2 = \{q\}$, $\gamma_1 = \neg(p \leftrightarrow q)$ and $\gamma_2 = p \leftrightarrow q$.

The size of a Boolean game is thus on the order of $n(\max_i(|\Phi_i|) + \max_i(|\gamma_i|))$. In the best case γ_i is small and the resulting representation is linear in the number of players and logarithmic in the number of strategies, giving greater succinctness on both fronts.

Having defined the game representation, we now turn to reasoning about such games. The most common solution concept is the Nash equilibrium, which we define below.

Definition 3. Given a strategy profile $\boldsymbol{s}$, we use $\boldsymbol{s}_{-i}(\sigma_i')$ to mean the profile obtained by replacing the strategy of i in $\boldsymbol{s}$ with σ_i'. A *best response* for i to $\boldsymbol{s}$ is some σ_i' that maximises $u_i(\boldsymbol{s}_{-i}(\sigma_i'))$.

A strategy profile $\boldsymbol{s} = (\sigma_1, \ldots, \sigma_n)$ where every σ_i is a best response to $\boldsymbol{s}$ is a *Nash equilibrium*.

Example 3. In a game of matching pennies, T is a best response for Player One to HH, and H is a best response for Player Two, so the profile HH is not in equilibrium. In fact, the game has no equilibria in pure strategies.

The fact that games as simple as matching pennies may fail to have a pure strategy equilibrium casts doubt on its suitability as a solution concept. Fortunately, a natural extension of the framework rectifies the matter.

Definition 4. A *mixed strategy* for player i in a finite strategic game is a probability distribution over S_i.

The utility player i obtains from a profile of mixed strategies S is $\sum p(S')u_i(S')$, where $p(S')$ is the probability assigned to the pure profile S' by the mixed strategies in S.

If the idea of playing a probability distribution seems strange, the reader may instead opt to view a mixed strategy as a randomised decision procedure that leads to an outcome in the set of pure strategies, S_i. In a game of rock-paper-scissors, a mixed strategy could consist of generating a number in $[0, 1]$ then playing rock if it falls between 0 and 0.4, paper if it falls between 0.4 and 0.9 and scissors if it is above 0.9. The payoff of a mixed strategy is the expected payoff - the sum of the possible outcomes weighted by their probability.

It is in this context that Nash proved his seminal result:

Theorem 1 (Nash, 1951). *Every finite strategic game has an equilibrium in mixed strategies.*

Example 4. The unique equilibrium of matching pennies involves both players randomising over their sets of strategies by assigning a weight of 0.5 to both H and T. In this equilibrium both players attain a utility of 0.5.

In the above example the reader will observe that 0.5 is also the utility Player One would attain by deviating to the pure strategy H or the pure strategy T. This is always the case - as the payoff of a mixed strategy is a weighted sum of pure strategies, it cannot yield more utility than any of its components individually. What this means for us is that if it is possible for a player to deviate from a profile, then it is possible to deviate with a pure strategy. If we want to verify that a mixed profile is in equilibrium, we need only

check that no pure strategy for a player will lead to a better outcome.

Since every game has an equilibrium, the algorithmic question of asking whether an equilibrium exists is not relevant. This motivates decision problems based on qualified notions of equilibria, such as the one that concerns us in this paper:

∃GUARANTEENASH: Given a Boolean game G and a rational-valued vector $\boldsymbol{v} \in [0,1]^n$, does G have an equilibrium $\boldsymbol{s}$ such that $u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$ for each player i?

For representation purposes, as the entries of $\boldsymbol{v}$ are rationals we will assume they are given as a pair of integers representing the numerator and denominator.

It is natural to also consider a problem closely related to the dual:

∀GUARANTEENASH: Given a Boolean game G and a rational-valued vector $\boldsymbol{v} \in [0,1]^n$, does every equilibrium of G, $\boldsymbol{s}$, satisfy $u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$ for each player i?

Main Result

Our reduction will be from the following NEXP-complete problem:

NEXPTM: Given a non-deterministic Turing machine M, an integer in binary K and a string w, does M accept w in at most K steps?

Proposition 1. NEXPTM *is NEXP-complete.*

Proof. For membership in NEXP, we need only simulate the computation of M on w for K steps. Each step can be simulated in non-deterministic polynomial time, and the number of steps is exponential in $|K|$.

For hardness, let N be a non-deterministic Turing machine with an exponential time clock f. Let M be a Turing machine with an identical transition relation to N, but with no internal clock. Clearly, N accepts w if and only if M accepts w in at most $f(w)$ steps. That is, $(M, f(w), w)$ is a positive instance of NEXPTM. Moreover, the triple $(M, f(w), w)$ is polynomial in the size of N and w: $|M| \leq |N|$, $|w| = |w|$ and as $f(w) \in O(2^{p(|w|)})$, when written in binary it is of size $O(p(|w|))$. This gives us the desired reduction. □

We can now prove the hardness of ∃GUARANTEENASH. For questions of NEXP-membership, see the discussion below.

Theorem 2. ∃GUARANTEENASH *for Boolean games is NEXP-hard.*

Proof. We will give a reduction from NEXPTM. Given a triple (M, K, w) we shall construct, in polynomial time, a Boolean game G and a utility vector $\boldsymbol{v}$, such that G has an equilibrium where player i's utility is at least $\boldsymbol{v}[i]$ if and only if M accepts w in K steps or less.

For convenience, we augment M with a "do nothing" transition: if M is at an accepting state, then we allow it to move to the next computation step without moving the head, changing state, or writing anything to the tape. It is clear that augmenting M in such a fashion does not change the language accepted by M, but it ensures that the machine state is defined at all computation steps; we do not need to worry about the case where the machine accepts in under K steps, as if it does, it will still accept at step K.

Let $k = |K|$, and q be the number of states of M.

For intuition, a computation history of M on w could be seen as a $K \times K$ table, or for simplicity $2^k \times 2^k$, padding as needed. The ith row, jth column contains the contents of the jth tape cell at the ith computation step, whether or not the head is over that cell at that step and, if so, which state the machine is in. As the size of the table is 2^k by 2^k, we can index an entry by using two k-bit integers.

A way to visualise the proof is that in our game G, which consists of six players, Player One is equipped with variables that allow him to describe a single entry of this table. Player Four plays a partial matching pennies game against Player One, thereby forcing Player One to play a mixed strategy randomising over all entries of the table, and thus specifying an entire computation history with his mixed strategy. Player Two then verifies that the mixed strategy provided by Player One contains a consistent description of the head location at each computation step, and Player Three checks that every two consecutive steps are linked by exactly one transition rule. Players Five and Six play matching pennies with Players Two and Three to force them to randomise across all table entries.

To this end, let:

$$\Phi_1 = \{Zero_1, One_1, Head_1, Left_1, Right_1\} \cup \{Time_1^i\}_{1\leq i\leq k} \cup \{Tape_1^i\}_{1\leq i\leq k} \cup \{State_1^i\}_{1\leq i\leq q}.$$

The intended meaning of $Time_1^i$ (respectively $Tape_1^i$) is the value of the ith most significant bit of the integer denoting the index of the computation step (respectively tape cell) in question, given the standard convention of interpreting "true" as 1 and "false" as 0. A truth assignment by Player One can therefore be read as: at the computation step specified by $Time_1^1, \ldots, Time_1^k$ the tape cell specified by $Tape_1^1, \ldots, Tape_1^k$ contains 0 if $Zero_1$, 1 if One_1 and is blank if neither. The machine head is hovering over the cell in question if $Head_1$, and is located to the left or right of that cell respectively if $Left_1$ or $Right_1$. If the head is over the cell in question, the machine is in state i if $State_1^i$ (if the head is not over the cell, $State_1^i$ is a junk variable that has no meaning).

Player One's goal formula is a conjunction of four subformulae:

$$\gamma_1 = Init \wedge Final \wedge Cons_1 \wedge \neg\gamma_4.$$

Intuitively, $Init$ means that if the player plays the first computation step, his description of the machine must agree with the initial configuration of M on w. $Final$ means that if the player plays the last (Kth) computation step, the machine must be in an accepting state. $Cons_1$ states the description of the machine must be internally consistent. The final conjunct is to force the player to randomise across all

computation steps and tape cells, to which we will return later.

Init requires that at time zero the configuration of the machine is faithfully represented by Player One's play. This takes the form of an implication where the antecedent states that we are at step zero:

$$Init = (\bigwedge_{1 \leq i \leq k} \neg Time_1^i) \rightarrow Consequent.$$

The consequent itself is a conjunction of three further subformulae, for the head, the state and the tape.

$$Consequent = InitHead \wedge InitState \wedge InitTape.$$

The head requirement states that the head is at the leftmost cell. That is, at cell zero $Head_1$ is true, and at every other cell $Left_1$ is true:

$$InitHead = \Big((\bigwedge_{1 \leq i \leq k} \neg Tape_1^i) \rightarrow Head_1\Big) \wedge \Big(\neg(\bigwedge_{1 \leq i \leq k} \neg Tape_1^i) \rightarrow Left_1\Big).$$

The state requirement is simply M's initial state:

$$InitState = State_1^{initial}.$$

The tape requirement is a conjunction of $|w| + 1$ implications. The first $|w|$ implications state that if the tape cell chosen is within the first $|w|$ cells, then its contents must agree with w. If we use i as shorthand for the conjunction of tape variables expressing i, and $w[i]$ for $Zero_1$ or One_1 depending on the ith bit of w, this has the following form:

$$InitTape = \bigwedge_{0 \leq i < |w|} (i \rightarrow w[i]) \wedge BlankCells.$$

Note that this formula is linear in $|w|$, so the construction so far was polynomial.

The last formula in *InitTape* states that all other cells are blank.

$$BlankCells = \neg(\bigvee_{0 \leq i < |w|} i) \rightarrow (\neg Zero_1 \wedge \neg One_1).$$

Final states that at computation step K, the machine accepts. If we use K as shorthand for the appropriate conjunction of time variables, we get the following implication:

$$Final = K \rightarrow State_1^{accepting}.$$

$Cons_1$ requires that the player's description of a given computation step and cell is internally consistent. This means the cell cannot have both 0 and 1 on it, the head must be either over the cell or to one direction and the machine must be in exactly one state. It is worth noting that this says nothing about whether Player One's descriptions of different steps and cells are consistent with each other: this is the task of Players Two and Three.

For $Cons_1$, we introduce a generalised XOR symbol, which we denote $\boldsymbol{OneOf}$, with the interpretation that $\boldsymbol{OneOf}(\varphi_1, \ldots, \varphi_n)$ is true if and only if exactly one φ_i is. Such a symbol could be replaced by a propositional logic formula polynomial in the size of $\varphi_1, \ldots, \varphi_n$ - simply take the disjunction of all n admissible possibilities. This gives us the desired formula:

$$\begin{aligned} Cons_1 = {} & \neg(Zero_1 \wedge One_1) \\ & \wedge \boldsymbol{OneOf}(Head_1, Left_1, Right_1) \\ & \wedge \boldsymbol{OneOf}(\overline{State_1^i}). \end{aligned}$$

By $\overline{State_1^i}$ we mean $State_1^1, \ldots, State_1^q$.

To finish the description of γ_1, we turn to Player Four. Player Four is playing a partial matching pennies game with Player One over the time and tape variables. We thus equip her with the following:

$$\Phi_4 = \{Time_4^i\}_{1 \leq i \leq k} \cup \{Tape_4^i\}_{1 \leq i \leq k}.$$

The objective is to guess the same computation step and cell index as player one:

$$\gamma_4 = \Big(\bigwedge_{1 \leq i \leq k} (Time_1^i \leftrightarrow Time_4^i)\Big) \wedge \Big(\bigwedge_{1 \leq i \leq k} (Tape_1^i \leftrightarrow Tape_4^i)\Big).$$

Since the payoffs of Player One and Four depend only on each other, we can already describe how they will behave in any equilibrium. Player One has the power to satisfy *Init*, *Final* and $Cons_1$ unilaterally, so all that remains is $\neg\gamma_4$. To maximise the chances of this happening, Player One's best bet is to play every step/cell with probability $1/2^{2k}$ - this will ensure that Player Four will guess the same step with probability $1/2^{2k}$, whereas if Player One were to play any (i, j) with a greater probability than $1/2^{2k}$, then Player Four could play (i, j) with probability 1 and thus guess Player One's choice more often.

As such in any equilibrium play Player One will describe entry (i, j) of the computation table with probability $1/2^{2k}$.

Player Two's purpose is to verify the consistency of Player One's description of the head. This involves verifying that at a given computation step the $Head_1$ variable is true in exactly one cell, $Left_1$ is true in every cell to the right and $Right_1$ is true in every cell to the left.

Now of course, she cannot verify the entire row in one go - there are 2^k cells to check. Instead a pure strategy for her will check only a pair of consecutive cells. Player Five will then force her to randomise over *all* pairs of consecutive cells - and if Player One's description is inconsistent, there must be a witnessing pair.

Player Two controls the following variables:

$$\begin{aligned} \Phi_2 = {} & \\ & \{Head_2, sHead_2, Left_2, sLeft_2, Right_2, sRight_2\} \\ & \cup \{Tape_2^i\}_{1 \leq i \leq k} \cup \{sTape_2^i\}_{1 \leq i \leq k} \cup \{Time_2^i\}_{1 \leq i \leq k}. \end{aligned}$$

The lowercase "s" can be read as "successor". The intended meaning of these variables is that $Tape_2^1, \ldots, Tape_2^k$ name a cell and $sTape_2^1, \ldots, sTape_2^k$ the cell directly to the right of it. The other variables state the location of the head

in relation to these two cells at the computation step specified by the time variables.

Player Two's goal formula is a conjunction of four subformulae:

$$\gamma_2 = \mathit{MatchOne}_2 \wedge \mathit{Cons}_2 \wedge \boldsymbol{Succ}_2 \wedge \neg\gamma_5.$$

Intuitively, $\mathit{MatchOne}_2$ states that Player Two ought to play the same head configuration as dictated by Player One. Cons_2 requires that this configuration be internally consistent. $\boldsymbol{Succ}_2$ is to ensure that the two cells chosen are indeed consecutive.

Before we state $\mathit{MatchOne}_2$ we ought to first ask what we mean by saying that players one and two play the same head configuration. As in any given (pure) strategy profile, either player will be describing a single computation step and at most two cells; if it turns out that they are speaking about different step/cell configurations we should not be concerned about whatever claims they make - we have no way to prove they are not in agreement, so we may as well assume that they are. Only in the instance where they happen to refer to the same step/cell should we expect accord. Since Player Two is referring to two cells in any play, we require that if either of the cells she references coincides with that referenced by Player One, they must agree.

The desired formula is thus of the following form:

$$\begin{aligned}\mathit{MatchOne}_2 = \\ &\mathit{AgreeTime} \rightarrow \Big((\mathit{AgreeCell} \rightarrow \mathit{AgreeHead}) \\ &\wedge (\mathit{sAgreeCell} \rightarrow \mathit{sAgreeHead})\Big).\end{aligned}$$

The subformulae are as follows:

$$\begin{aligned}\mathit{AgreeTime} &= \bigwedge_{1\le i\le k} (\mathit{Time}_1^i \leftrightarrow \mathit{Time}_2^i).\\ \mathit{AgreeCell} &= \bigwedge_{1\le i\le k} (\mathit{Tape}_1^i \leftrightarrow \mathit{Tape}_2^i).\\ \mathit{sAgreeCell} &= \bigwedge_{1\le i\le k} (\mathit{Tape}_1^i \leftrightarrow \mathit{sTape}_2^i).\\ \mathit{AgreeHead} &= (\mathit{Head}_1 \leftrightarrow \mathit{Head}_2)\\ &\quad \wedge (\mathit{Left}_1 \leftrightarrow \mathit{Left}_2)\\ &\quad \wedge (\mathit{Right}_1 \leftrightarrow \mathit{Right}_2).\\ \mathit{sAgreeHead} &= (\mathit{Head}_1 \leftrightarrow \mathit{sHead}_2)\\ &\quad \wedge (\mathit{Left}_1 \leftrightarrow \mathit{sLeft}_2)\\ &\quad \wedge (\mathit{Right}_1 \leftrightarrow \mathit{sRight}_2).\end{aligned}$$

Internal consistency amounts simply to the conjunction of the valid combinations of claims about the head:

$$\begin{aligned}\mathit{Cons}_2 = &(\mathit{Right}_2 \wedge \mathit{sRight}_2) \vee (\mathit{Right}_2 \wedge \mathit{sHead}_2)\\ &\vee (\mathit{Head}_2 \wedge \mathit{sLeft}_2) \vee (\mathit{Left}_2 \wedge \mathit{sLeft}_2)\end{aligned}$$

Succ_2 states that the two tape locations are, in fact, consecutive. We will prove a lemma to show that this is concisely expressible in propositional logic.

Lemma 1. *Let $\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n)$ be a formula that is true if and only if the binary integer encoded by $q_1,\ldots,q_n$ is the successor of the binary integer encoded by $p_1,\ldots,p_n$. As a convention, 2^n-1 has no successor.*

$\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n)$ can be replaced by a propositional formula of size polynomial in $p_1,\ldots,p_n$ and $q_1,\ldots,q_n$.

Proof. We take advantage of the fact that to increment a binary integer we only need to modify the rightmost consecutive block of 1s, and there are only n such possible blocks.

Since we have a boundary condition to consider, we require that the first integer is not 2^n-1:

$$\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n) = \neg(\bigwedge_{1\le i\le n} p_i) \wedge \mathit{Succ}'.$$

Succ' is then:

$$\begin{aligned}&\Big(\neg p_1 \rightarrow \big(q_1 \wedge \bigwedge_{i=2}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\wedge\Big((p_1 \wedge \neg p_2) \rightarrow \big(\neg q_1 \wedge q_2 \wedge \bigwedge_{i=3}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\wedge\Big((p_1 \wedge p_2 \wedge \neg p_3) \rightarrow \big(\neg q_1 \wedge \neg q_2 \wedge q_3 \wedge \bigwedge_{i=4}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\ \vdots\\ &\wedge\Big((\neg p_n \wedge \bigwedge_{i=1}^{n-1} p_i) \rightarrow \big((\bigwedge_{i=1}^{n-1} \neg q_i) \wedge q_{i+1}\big)\Big).\end{aligned}$$

This is quadratic in the number of variables, giving us the desired result. □

Succ_2 can then be stated simply:

$$\mathit{Succ}_2 = \boldsymbol{Succ}(\overline{\mathit{Tape}_2^i}; \overline{\mathit{sTape}_2^i}).$$

Finally, Player Five is trying to guess Player Two's choice of cell and computation step.

$$\begin{aligned}&\Phi_5 = \{\mathit{Time}_5^i\}_{1\le i\le k} \cup \{\mathit{Tape}_5^i\}_{1\le i\le k}.\\ &\gamma_5 = \bigwedge_{i=1}^{k}(\mathit{Tape}_2^i \leftrightarrow \mathit{Tape}_5^i) \wedge \bigwedge_{i=1}^{k}(\mathit{Time}_2^i \leftrightarrow \mathit{Time}_5^i).\end{aligned}$$

Player Three's purpose is to verify that the tape contents in successive computation steps respect the transition rules of M. Like Player Two, he does this for a small, local part of the table, and is forced to randomise over the whole table by Player Six. To do this he specifies a total of six cells and two computation steps: consecutive triples in consecutive steps. Then he verifies that the tape contents, head position

and machine state are in agreement with some rule of M.

$$\begin{aligned}\Phi_3 = \{&pHead_3, Head_3, sHead_3, npHead_3, nHead_3,\\ &nsHead_3, pZero_3, Zero_3, sZero_3, npZero_3,\\ &nZero_3, nsZero_3, pOne_3, One_3, sOne_3,\\ &npOne_3, nOne_3, nsOne_3\}\\ \cup&\{pState_3^i\}_{1\le i\le q} \cup \{State_3^i\}_{1\le i\le q}\\ \cup&\{sState_3^i\}_{1\le i\le q} \cup \{npState_3^i\}_{1\le i\le q}\\ \cup&\{nState_3^i\}_{1\le i\le q} \cup \{nsState_3^i\}_{1\le i\le q}\\ \cup&\{pTape_3^i\}_{1\le i\le k} \cup \{Tape_3^i\}_{1\le i\le k}\\ \cup&\{sTape_3^i\}_{1\le i\le k} \cup npTape_3^i\}_{1\le i\le k}\\ \cup&\{nTape_3^i\}_{1\le i\le k} \cup \{nsTape_3^i\}_{1\le i\le k}\\ \cup&\{Time_3^i\}_{1\le i\le k} \cup \{nTime_3^i\}_{1\le i\le k}.\end{aligned}$$

The "p" can be read as "predecessor", referring to the cell to the left, and "n" as "next computation step". The intended meaning is simply the state and tape contents in each of the six cells, as well as whether the head is over that cell.

Player Three's goal formula is a conjunction of five subformulae:

$$\gamma_3 = MatchOne_3 \wedge Triple \wedge Succ_3 \wedge Rules \wedge \neg\gamma_6.$$

$MatchOne_3$ states that if any of the step/cell pairs named by Player Three coincide with the one named by Player One, Player Three must agree with Player One. $Triple$ requires that the three cells named in either computation step should be a consecutive triple, and the triple at either step must be the same. $Succ_3$ requires the two computation steps named to be consecutive. $Rules$ is to verify that the configuration thus described is consistent with a rule of M.

$MatchOne_3$ is a conjunction of a total of six statements, depending on which step/cell pair coincides with that played by Player One. We will only give one such statement below, in the case that Player One named the same step as $Time_3^1, \ldots, Time_3^k$ and the same cell as $pTape_3^1, \ldots, pTape_3^k$. The other five statements are obtained in the obvious manner.

$$\begin{aligned}&\Big(\bigwedge_{i=1}^{k}(Time_1^i \leftrightarrow Time_3^i) \wedge \bigwedge_{i=1}^{k}(Tape_1^i \leftrightarrow pTape_3^i)\Big) \rightarrow\\ &\Big((Zero_1 \leftrightarrow pZero_3) \wedge (One_1 \leftrightarrow pOne_3)\\ &\quad\wedge (Head_1 \leftrightarrow pHead_3) \wedge \bigwedge_{i=1}^{q}(State_1^i \leftrightarrow pState_3^i)\Big).\end{aligned}$$

$Triple$ states that the tape cells selected are consecutive triples, and that the same triple is chosen in both steps. It is worth noting that given our previous definition of successor, if Player Three is to satisfy this conjunct then the middle cell cannot be 0 or $2^k - 1$.

$$\begin{aligned}Triple = &\boldsymbol{Succ}(\overline{pTape_3^i}; \overline{Tape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{Tape_3^i}; \overline{sTape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{npTape_3^i}; \overline{nTape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{nTape_3^i}; \overline{nsTape_3^i})\\ &\wedge \bigwedge_{1\le i\le k}(Tape_3^i \leftrightarrow nTape_3^i).\end{aligned}$$

$Succ_3$ requires that the computation steps be consecutive:

$$Succ_3 = \boldsymbol{Succ}(\overline{Time_3^i}; \overline{nTime_3^i}).$$

$Rules$ is a conjunction of four formulae: three of the formulae are conjunctions containing an implication for each $(r, s) \in Q \times \{0, 1, \perp\}$, representing the machine's behaviour if it reads s in state r and the head is over the left, centre or right cell respectively. The fourth term is $NoHead$, to handle the case where the head is not over any cell in the triple:

$$Rules = Left \wedge Centre \wedge Right \wedge NoHead.$$

We will examine $Left$ and $NoHead$, understanding that $Centre$ and $Right$ are handled in similar fashion.

$$\begin{aligned}Left = \Big(\bigwedge_{(r,s)\in Q\times\{0,1,\perp\}} &\big((pState_3^r \wedge s) \rightarrow\\ &\boldsymbol{OneOf}(\overline{Rule[(r,s) \rightarrow (r', s', D)]})\big)\Big).\end{aligned}$$

The s in the antecedent is meant to be replaced by $pZero_3$, $pOne_3$, or $\neg(pZero_3 \vee pOne_3)$ as appropriate. The intuition of the $Rules$ term is that should the machine read s in state r it should pick exactly one of the rules available to it, and if the head is not present then the tape contents should not change.

The subformula to deal with a specific rule can be broken up as follows:

$$Rule[(r, s) \rightarrow (r', s', D)] = L \wedge B.$$

L describes the behaviour of the machine if the left cell is not the leftmost cell on the tape, B deals with the boundary case where it is.

We will give an example of how $Rule[(q_3, 0) \rightarrow (q_4, 1, \mathit{left})]$ would be handled. That is, the rule that says that if the machine reads a 0 in state q_3 it may write a 1, transition to state q_4 and move the head left. All rules except "do nothing" can be handled similarly, and "do nothing" would merely assert that if the machine reads an accepting state, then nothing changes.

The L part triggers if the head is over the leftmost cell in the triple, and the leftmost cell is not cell 0. It then ensures that in the next computation step the leftmost cell contains 1 and the other cells are unchanged. Since the head leaves the

monitored triples we need no terms to account for it.

$$L = \Big(\neg(\bigwedge_{1\leq i\leq k} \neg pTape_3^i) \wedge pHead_3\Big) \rightarrow \Big(npOne_3 \wedge (Zero_3 \leftrightarrow nZero_3) \wedge (sZero_3 \leftrightarrow nsZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (sOne_3 \leftrightarrow nsOne_3)\Big).$$

In the boundary case the head is over the leftmost cell of the tape, so when it attempts to move left it instead stands still.

$$B = \Big((\bigwedge_{1\leq i\leq k} \neg pTape_3^i) \wedge pHead_3\Big) \rightarrow \Big(npState_3^4 \wedge npOne_3 \wedge npHead_3 (Zero_3 \leftrightarrow nZero_3) \wedge (sZero_3 \leftrightarrow nsZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (sOne_3 \leftrightarrow nsOne_3)\Big).$$

Finally, the $NoHead$ term asserts in the absence of a head the tape contents do not change.

$$NoHead = (\neg pHead_3 \wedge \neg Head_3 \wedge \neg sHead_3) \rightarrow \Big((pOne_3 \leftrightarrow npOne_3) \wedge (nZero_3 \leftrightarrow npZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (Zero_3 \leftrightarrow nZero_3) \wedge (sOne_3 \leftrightarrow nsOne_3) \wedge (sZero_3 \leftrightarrow nsZero_3)\Big).$$

This brings us to the last player, who is trying to guess the first step and central cell chosen by Player Three:

$$\Phi_6 = \{Time_6^i\}_{1\leq i\leq k} \cup \{Tape_6^i\}_{1\leq i\leq k}.$$
$$\gamma_6 = \bigwedge_{i=1}^{k}(Tape_3^i \leftrightarrow Tape_6^i) \wedge \bigwedge_{i=1}^{k}(Time_3^i \leftrightarrow Time_6^i).$$

The construction so far has been polynomial. We now claim that M having an accepting run on w in at most K steps is equivalent to the constructed game having a Nash equilibrium where Players One, Two and Three have the following guaranteed payoffs:

$$\boldsymbol{v}[1] = \frac{2^{2k}-1}{2^{2k}}.$$
$$\boldsymbol{v}[2] = \frac{2^k(2^k-1)-1}{2^k(2^k-1)}.$$
$$\boldsymbol{v}[3] = \frac{(2^k-2)(2^k-1)-1}{(2^k-2)(2^k-1)}.$$

Strictly speaking, only the payoffs of Two and Three are necessary - Player One will be able to attain $\boldsymbol{v}[1]$ utility even if M does not accept w. However, we include One's utility above as it will help us to reason about the profile being in equilibrium.

First, suppose M has an accepting run on w in at most K steps. Consider the profile where Player One randomises over all step/cell combinations with equal weight, and at each step/cell combination plays his variables in accordance to the accepting run. Player Four also randomises over all step/cell combinations with equal weight. Player Two randomises over all computation steps and the first $2^k - 1$ cells. Her other variables she plays in accordance to the run. Player Five likewise randomises over all steps and the first $2^k - 1$ cells. Player Three randomises over the first $2^k - 1$ steps and the $2^k - 2$ cells between the first and last. His other variables he plays in accordance to the run. Player Six randomises over the same $2^k - 1$ steps and the $2^k - 2$ cells.

In such a profile, Players One, Two and Three will satisfy their goals unless their step/cell combination is guessed by their opponent. Given our setup, this will happen with probabilities $1/2^{2k}$, $1/2^k(2^k-1)$ and $1/(2^k-2)(2^k-1)$ respectively, giving us the payoffs $\boldsymbol{v}[1]$, $\boldsymbol{v}[2]$ and $\boldsymbol{v}[3]$. It remains to see that this profile is in equilibrium. Recall that this is equivalent to checking that no player has a pure strategy that will grant them more utility than what they are earning already.

Let us first consider Players Four through Six. Any pure strategy by Player Four is a step/cell pair, and hence, given the play of Player One, has a $1/2^{2k}$ chance of satisfying γ_4. Player Four is thus indifferent between the current situation and any deviation. For Player Five any pure strategy using the first $2^k - 1$ cells will have a $1/2^k(2^k - 1)$ chance of satisfying γ_5, and any other pure strategy 0. Player Five thus likewise has no incentive to deviate. In the same fashion, any pure strategy for Player Six will satisfy γ_6 with probability $1/(2^k - 2)(2^k - 1)$ or 0, so she is also indifferent.

In the case of Player One, observe that no matter what pure strategy he picks, there is a $1/2^{2k}$ chance of Player Four guessing the cell/step component and thus making γ_1 false. It follows that any such strategy will yield at most a $\boldsymbol{v}[1]$ chance of satisfying γ_1. For Player Two, if she picks a pure strategy using the first $2^k - 1$ cells there will likewise be a $1/2^k(2^k-1)$ chance of her step/cell combination being guessed. If she picks a pure strategy using the last cell, she will be unable to satisfy the $Succ_2$ component of γ_2, yielding a utility of 0. For Player Three, any pure strategy using the $2^k - 1$ steps and the $2^k - 2$ cells randomised over by six will have a $1/(2^k-2)(2^k-1)$ chance of being guessed, and any other choice of pure strategy will violate either $Triple$ or $Succ_3$. This establishes that the described profile is in equilibrium.

Next, suppose that no accepting run exists. We claim that in any equilibrium Player One will still obtain a utility of $\boldsymbol{v}[1]$, but either Player Two or three will be unable to secure a payoff of $\boldsymbol{v}[2]$, $\boldsymbol{v}[3]$. For the first part, note that for any choice of step/cell by Player One, the remaining variables can be set to satisfy $Init$, $Final$ and $Cons_1$ unilaterally. It is sufficient to simply respect the initial configuration of the machine at step zero, play an accepting state at step K, and any internally consistent description elsewhere. Any strategy that does not satisfy $Init$, $Final$ and $Cons_1$ is thus dominated and can be excluded from consideration. All that remains is the choice of cell/step and it is easy to see that the only equilibrium play would involve giving every pair equal weight.

Player One's play will thus describe a sequence of 2^k con-

figurations of M, with the initial configuration at step zero and an accepting state at step K. However, as M has no accepting run on w in K steps, this sequence cannot represent a valid computation and a violation must occur somewhere.

If this violation involves the assertion of the presence of more than one head or the $Left_1$, $Right_1$ variables incorrectly specifying the location of the head, we claim that Player Two cannot obtain a utility of $\boldsymbol{v}[2]$.

Observe that in this case there must exist two consecutive cells at some time step where Player One plays one of the following combinations:

Cell i	Cell $i+1$
$Left_1$	$Right_1$
$Left_1$	$Head_1$
$Head_1$	$Head_1$
$Right_1$	$Left_1$
$Head_1$	$Right_1$

In this case, should Player Two play a strategy involving cell i, since she is committed to playing a legal head assignment she will have to disagree with Player One on either cell i or cell $i+1$. This means she will suffer a $1/2^{2k}$ chance of having $MatchOne$ falsified if Player One plays the cell in question. As there is still at least a $1/2^k(2^k-1)$ chance of having the cell/step combination guessed by Player Five, this means the maximum utility Player Two can obtain in this case is $\boldsymbol{v}[2] - 1/2^{2k} + 1/2^{2k}2^k(2^k-1)$. (The last term is to avoid double counting the case where both Player One and Player Five name the same cell/step combination.)

Of course, Player Two may opt in this case not to play any strategies involving cell i. This will however mean that she is randomising over at most (2^k-2) cells, and Player Five will randomise accordingly, meaning the highest utility she can obtain is $\frac{2^k(2^k-2)-1}{2^k(2^k-2)}$.

Suppose now that Player One does not make such a violation. The remaining possibilities for an incorrect run are:

1. The head makes an illegal transition.
2. The tape contents undergo an illegal change.
3. The state undergoes an illegal change.

Let us deal with case 1. Suppose between step t and $t+1$ the head, which is at cell i at t, performs an illegal transition. This could mean moving more than one cell in a direction, moving off the edge of the tape, staying still in a non-accepting state or moving one cell left or right without a justifying transition rule. Observe that neither of these possibilities is consistent with the $Rules$ requirement. As such, should Player Three pick step t and cell i, he will have to disagree with Player One on the movement of the head, thereby running a risk of falsifying his formula should Player One play t and i. This will prevent Player Three from obtaining v_3 utility for the same reasoning as with Player Two.

In case 2, there would exist steps t and $t+1$, and a cell i the contents of which would change without a justifying rule. This, too, would violate $Rules$. For case 3, we note that by the machine state we mean the state variable that occurs in the same cell as the head: the value of the other state variables is of no account. As such, $Rules$ again would be violated as it requires the correct state to be propagated to cell hosting the head. This completes the proof. □

We can adapt this proof to show that ∀GUARANTEENASH is coNEXP-hard. Note that this does not follow immediately: ∀GUARANTEENASH is not simply the complement of ∃GUARANTEENASH. Letting $\boldsymbol{s}$ range over equilibrium profiles, ∀GUARANTEENASH is the question whether:

$$\forall \boldsymbol{s} \,.\, \forall i \,.\, u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$$

the complement of ∀GUARANTEENASH is then:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) < \boldsymbol{v}[i].$$

To show that ∀GUARANTEENASH is coNEXP-hard we need only show that the latter problem is NEXP-hard.

Corollary 1. ∀GUARANTEENASH *is coNEXP-hard.*

Proof. We argue that the proof of Theorem 2 can be adapted to show this. Note that the utilities of Players One, Four, Five and Six did not play a rôle in the proof. Those of Four, Five and Six were omitted entirely, whereas Player One has been seen to achieve $\boldsymbol{v}[1]$ utility in every equilibrium. What remains are Two and Three, and we will argue that those players could be collapsed into a single player.

Introduce a new player into the game constructed in the proof of Theorem 2, Player Seven, with $\gamma_7 = \gamma_2 \wedge \gamma_3$ and $\Phi_7 = \emptyset$. We argue that the Turing machine M accepts w in at most K steps if and only if there exists an $\boldsymbol{s}$ for which:

$$u_7(\boldsymbol{s}) \geq 1 - \frac{(2^k-2)(2^k-1) + 2^k(2^k-1) - 1}{2^k(2^k-1)(2^k-2)(2^k-1)}.$$

This can be seen by replicating the argument in the proof: in the presence of an accepting run, the only way Player Seven can lose utility is if Player Five or Six guesses the same cell/step, which happens with probabilities $\frac{1}{2^k(2^k-1)}$ and $\frac{1}{(2^k-2)(2^k-1)}$ respectively. Adding a term for double counting and simplifying yields the quantity above.

If we call that quantity $\boldsymbol{v}[7]$, we have established that $u_7(\boldsymbol{s}) \geq \boldsymbol{v}[7]$ if and only if M accepts w in at most K steps. If we set the other entries of $\boldsymbol{v}$ to 1 to make sure they do not interfere, we demonstrate that the following question is NEXP-hard:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i].$$

For the next step, add Player Eight with $\gamma_8 = \neg\gamma_7$ and $\Phi_8 = \emptyset$. As $u_8 = 1 - u_7$ the following question is NEXP-hard as well, letting $\boldsymbol{v}[8] = 1 - \boldsymbol{v}[7]$ and the other entries of $\boldsymbol{v}$ be 0:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) \leq \boldsymbol{v}[i].$$

It remains to show that the inequality can be made strict.

First, observe that we can increase Player Seven's score, and hence decrease Player Eight's, by an arbitrarily small ϵ of a certain form: let $\gamma_7' = \gamma_7 \vee Pennies$ where $Pennies$ is a matching pennies game over a new set of variables Φ_7' against some new player. This will give Player Seven $1/2^{|\Phi_7'|}$ additional utility, minus a double counting term.

All that remains is to show that we can identify a "sufficiently small" ϵ. By this we mean an ϵ satisfying the following:

$$\exists \boldsymbol{s} \,.\, u_8(\boldsymbol{s}) - \epsilon < \boldsymbol{v}[8] \iff \exists \boldsymbol{s} \,.\, u_8(\boldsymbol{s}) \leq \boldsymbol{v}[8].$$

To see that this is possible, recall that if M does not accept w in K steps, then Player One necessarily specifies an incorrect computation history of the machine. As we have seen in the proof of Theorem 2, such a violation decreases the maximum attainable score of Player Two or Three by a fixed amount. It is thus possible to calculate the maximum attainable utility of Player Seven in the presence of such a violation, which will give us the bounds within which ϵ may reside.

This completes the proof. □

Discussion

The preceding proof raises two related questions. To begin with, one may ask whether six players are necessary. The answer is no, as we have hinted in the proof of the corollary. The reader may convince themselves that one may reduce the number to three in a straightforward fashion by collapsing Players Two and Three, and Four, Five and Six onto each other, in a similar fashion to the proof of the corollary. We used six players to simplify the exposition of the proof. Whether it is further possible to reduce the number to two is a different matter - and an interesting one, as we will see shortly

Second: whether there is a membership result to go with the hardness. Strictly speaking, there is not. As there exist games where every equilibrium requires irrational weights on the strategies chosen (Nash 1951; Bilò and Mavronicolas 2012) we cannot rely on the intuitive approach of guessing a strategy profile and checking whether it is in equilibrium.

One way this problem is addressed in the literature is to restrict attention to two player games, where a rational equilibrium *is* guaranteed to exist (Cottle and Dantzig 1968), which brings us back to the first question. The second way is to consider the notion of an ϵ-equilibrium: a profile of strategies where no player can gain more than ϵ utility by deviating. This problem, ϵ-$\exists$GUARANTEENASH, clearly does belong to NEXP, and the reader can convince themselves that by inserting a sufficiently small ϵ into the proof above we can establish that it is NEXP-complete.

Conclusion

We have shown that the problem of determining whether a Boolean game has a Nash equilibrium which guarantees each player a certain payoff is NEXP-hard. This is the first complexity result about mixed equilibria in the Boolean games framework, and demonstrates that in this instance Boolean games are as difficult as the more general class of Boolean circuit games.

The complexity of many other natural problems remains open, most significantly that of NASH: the task of computing a mixed equilibrium. However, given the difficulty in obtaining this result for normal form games (Daskalakis, Goldberg, and Papadimitriou 2006) one could posit that it is unlikely that this can be achieved with the current tools of complexity theory. It would be interesting to see whether there is an exponential time analogue of PPAD that could lead to a solution to this problem.

Acknowledgements

Egor Ianovski is supported by a scholarship, and Luke Ong is partially supported by a grant, from the Oxford-Man Institute of Quantitative Finance.

References

Ågotnes, T.; van Benthem, J.; van Ditmarsch, H. P.; and Minica, S. 2011. Question-answer games. *Journal of Applied Non-Classical Logics* 21(3-4):265–288.

Ågotnes, T.; Harrenstein, P.; van der Hoek, W.; and Wooldridge, M. 2013a. Boolean games with epistemic goals. In *LORI*, 1–14.

Ågotnes, T.; Harrenstein, P.; van der Hoek, W.; and Wooldridge, M. 2013b. Verifiable equilibria in boolean games. In *IJCAI*.

Bilò, V., and Mavronicolas, M. 2012. The complexity of decision problems about Nash equilibria in win-lose games. In Serna, M., ed., *Algorithmic Game Theory*, Lecture Notes in Computer Science. Springer Berlin Heidelberg. 37–48.

Bilò, V. 2007. On satisfiability games and the power of congestion games. In Kao, M.-Y., and Li, X.-Y., eds., *Algorithmic Aspects in Information and Management*, volume 4508 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 231–240.

Bonzon, E.; Lagasquie-Schiex, M.-C.; Lang, J.; and Zanuttini, B. 2006. Boolean games revisited. In *ECAI*, 265–269.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2009. Dependencies between players in Boolean games. *Int. J. Approx. Reasoning* 50(6):899–914.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2012. Effectivity functions and efficient coalitions in Boolean games. *Synthese* 187(1):73–103.

Cottle, R. W., and Dantzig, G. B. 1968. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications* 1:103–125.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2006. The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, 71–78. New York, NY, USA: ACM.

Dunne, P. E., and van der Hoek, W. 2004. Representation and complexity in Boolean games. In Alferes, J. J., and Leite, J. a., eds., *Logics in Artificial Intelligence*, volume 3229 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 347–359.

Dunne, P. E., and Wooldridge, M. 2012. Towards tractable Boolean games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, 939–946. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Dunne, P. E.; van der Hoek, W.; Kraus, S.; and Wooldridge, M. 2008. Cooperative Boolean games. In *AAMAS'08*, 1015–1022.

Endriss, U.; Kraus, S.; Lang, J.; and Wooldridge, M. 2011. Designing incentives for Boolean games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, 79–86.

Feigenbaum, J.; Koller, D.; and Shor, P. 1995. A game-theoretic classification of interactive complexity classes (extended abstract). In *Proceedings of the tenth annual IEEE conference on computational complexity*, 227–237.

Harrenstein, P.; van der Hoek, W.; Meyer, J.-J.; and Witteveen, C. 2001. Boolean games. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, TARK '01, 287–298. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kraus, S., and Wooldridge, M. 2012. Delegating decisions in strategic settings. In *ECAI 2012*, 468–473.

Levit, V.; Grinshpoun, T.; Meisels, A.; and Bazzan, A. L. C. 2013. Taxation search in boolean games. In *AAMAS*, 183–190.

Mavronicolas, M.; Monien, B.; and Wagner, K. W. 2007. Weighted Boolean formula games. In Deng, X., and Graham, F., eds., *Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 469–481.

Nash, J. 1951. Non-cooperative games. *Annals of Mathematics* 54(2):286–295.

Ågotnes, T., and van Ditmarsch, H. 2011. What will they say?public announcement games. *Synthese* 179(1):57–85.

Rosenthal, R. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory* 2(1):65–67.

Schoenebeck, G. R., and Vadhan, S. 2012. The computational complexity of Nash equilibria in concisely represented games. *ACM Trans. Comput. Theory* 4(2):4:1–4:50.